# A Technique for the Direct Manipulation of Spline Curves

Richard H. Bartels
John C. Beatty

Computer Graphics Laboratory
Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
(519) 888-4534

**Abstract**

A frequent objection to the use of Bézier and B-spline curves is that the manipulation of control vertices lying off the curve is non-intuitive. Whether true or not, they incontestably increase the clutter on-screen. We introduce a simple technique by which the user may reshape a Bézier or B-spline curve by picking any point on the curve and moving it to a new location, through which an appropriately altered curve then passes. Computation of the new curve can be done in real-time, and the technique can be generalized to simultaneously satisfy a useful class of constraints.

## 1. Introduction

Consider the *Bézier curve* shown in Figure 1. It is defined by a sequence of *control vertices* positioned in the plane, shown in Figure 2 by small squares. The curve is actually an approximation of the piecewise-linear *control polygon* that results from connecting the control vertices in order. It is generally reshaped by moving one of the control vertices, thus resulting in a new approximation. Hence the control vertices are customarily shown when the curve is to be reshaped, though of course they need only be shown when the user wishes to select one so as to alter the curve. Because the order in which the control vertices are approximated is part of the definition, it is frequently necessary to show the control polygon as well. Significant clutter often results, especially for small curves overlying other objects.

Figure 1. A Bézier curve.

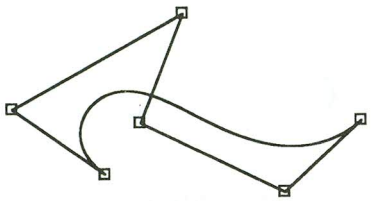

Figure 2. The Bézier curve of Figure 1, together with its *control vertices* (shown as small squares), and its *control polygon* (the sequence of line segments joining the control vertices).

Points on a Bézier curve are defined as a function of the positions of the control vertices. It seems reasonable to suppose that given such a point, together with a new location through which we would like the curve to pass instead, we might compute new positions for the control vertices that cause this to happen. This would enable us to suppress display of the control vertices and control polygon, of which the user then need not be aware.

In fact, there is more than enough freedom to accomplish this: we are given the $x$ and $y$ coordinates of the new position through which the curve is to pass — two constraints — while we have at our disposal the $x$ and $y$ coordinates of all the control vertices. How to select from among the infinite number of possible control vertex positions that cause the curve to interpolate a new location those that also "cause a suitable change in the curve's shape," and how one may select a solution that satisfies other useful constraints, is the subject of this paper.

In what follows, we provide only the background and definitions necessary to fix our notation. For details the reader is referred to [Bartels], [Farin], [Faux], or [Mortenson].

## 2. The Technique

Consider a Bézier curve $C(u) = (X(u),Y(u))$. Without loss of generality, the curve is traced out from beginning to end as $u$ varies from 0 to 1. Let us number the control vertices 0 through $d$, and represent the position of the $i^{th}$ control vertex by $(x_i,y_i)$. A cubic Bézier curve is then given by the equations

$$X(u) = x_0B_{0,3}(u) + x_1B_{1,3}(u) + x_2B_{2,3}(u) + x_3B_{3,3}(u) \quad \text{(e 1)}$$

$$Y(u) = y_0B_{0,3}(u) + y_1B_{1,3}(u) + y_2B_{2,3}(u) + y_3B_{3,3}(u)$$

where

$$B_{i,d}(u) = \binom{d}{i}(1-u)^{d-i}\,u^i$$

is the $i^{th}$ *Bézier basis function* or *Bernstein polynomial* of degree $d$, which for clarity we shall abbreviate as $B_i$ when $d$ is clear from the context. Let us suppose that the user has selected some point $(x,y)$ along the curve, thus fixing the value of $u$, and moved the cursor to a new location $(x',y')$ through which the curve is to pass for the same value of $u$. If we write $x'$ as $x+\Delta x$ and $y'$ as $y+\Delta y$, then for a cubic Bézier curve we may write

$$x + \Delta x = (x_0+\Delta x_0)B_0 + (x_1+\Delta x_1)B_1 + (x_2+\Delta x_2)B_2 + (x_3+\Delta x_3)B_3 \quad \text{(e 2)}$$

$$y + \Delta y = (y_0+\Delta y_0)B_0 + (y_1+\Delta y_1)B_1 + (y_2+\Delta y_2)B_2 + (y_3+\Delta y_3)B_3$$

whence

$$\Delta x = \Delta x_0 B_0 + \Delta x_1 B_1 + \Delta x_2 B_2 + \Delta x_3 B_3 \quad \text{(e 3)}$$

$$\Delta y = \Delta y_0 B_0 + \Delta y_1 B_1 + \Delta y_2 B_2 + \Delta y_3 B_3$$

We have two equations in the eight unknowns $\Delta x_i$ and $\Delta y_i$, so there are many ways in which we could choose the $\Delta x_i$ and $\Delta y_i$.

Setting $\Delta x_i = \Delta x$, and $\Delta y_i = \Delta y$ certainly works — it is well known that the Bernstein polynomials sum to one, so for this choice of the $\Delta x_i$ and $\Delta y_i$ (e 3) becomes

$$\Delta x = \Delta x\,(B_0 + B_1 + B_2 + B_3) = \Delta x$$

$$\Delta y = \Delta y\,(B_0 + B_1 + B_2 + B_3) = \Delta y$$

But this merely translates the curve – useful, but it does not result in a change of shape.

Let us instead chose *weights* $w_i$, and use them to compute the $\Delta x_i$ and $\Delta y_i$, according to the following formula:

$$\Delta x_i = \Delta x\, w_i = \Delta x \frac{B_i}{B_0^2 + B_1^2 + B_2^2 + B_3^2} \quad \text{(e 4)}$$

$$\Delta y_i = \Delta y\, w_i = \Delta y \frac{B_i}{B_0^2 + B_1^2 + B_2^2 + B_3^2}$$

We shall see in Section 3 that there is a sound theoretical justification for this choice. In the meantime, we note that the amount by which each control vertex moves is proportional to its influence on the point being moved – surely an intuitive choice.

It is straightforward to verify by substitution that equations (e 4) satisfy equations (e 3). They are generalized to Bézier curves of degree $d$ in the obvious way – the denominator is simply the sum of the squares of the $d$ basis functions. (Note that while the Bernstein polynomials sum to one, their squares do not.)

Figure 3 shows the effect of moving a point on a Bézier curve by computing new control vertex positions using the fifth degree version of equations (e 4). The new curve is a perfectly reasonable alteration of the old, given that it is now required to go through the indicated position.

Figure 3. Direct manipulation of a fifth degree Bézier curve. The original curve is shown in gray and the altered curve in black. The black square indicates the point moved.

For illustrative purposes, Figure 4 shows the corresponding movement in the control vertices as the indicated point on the curve is repositioned.

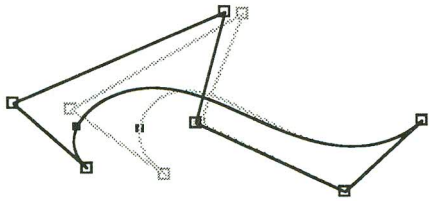

Figure 4. The curve of Figure 3, but with the control vertices and polygon shown. The control vertices that move the most are those having the most influence over the point moved, which is to say those whose corresponding basis functions are largest, so that the control vertex at the right end of the curve moves hardly at all.

One of the classical advantages of Bézier curves is that they pass through their first and last control vertices. This allows the user to precisely position a curve's endpoints, which do not move as any of the curve's remaining control vertices are manipulated to change its shape. Our technique appears to have the disadvantage that it will move these endpoints.

This is an easy thing to fix, however. Equations (e 3) work because the denominator is "the sum of the squares of the basis functions of the control vertices being moved." We can restrict movement to a subset of the potentially available control vertices by letting the denominator be the sum of the squares of the basis functions associated with exactly those control vertices, and computing weights with the form of (e 3) only for those vertices. So if we wish to leave the endpoints of a cubic Bézier curve fixed in place, we need only alter (e 3) to

$$\Delta x = \Delta x_1 B_1 + \Delta x_2 B_2 \quad \text{(e 5)}$$

$$\Delta y = \Delta y_1 B_1 + \Delta y_2 B_2$$

where

$$\Delta x_i = \Delta x\, w_i = \Delta x \frac{B_i}{B_1^2 + B_2^2} \quad \text{(e 6)}$$

$$\Delta y_i = \Delta y\, w_i = \Delta y \frac{B_i}{B_1^2 + B_2^2}$$

It is straightforward to verify by substitution that these also satisfy equations (e 3). Figure 5 and Figure 6 are analogous to Figure 3 and Figure 4, respectively, except that we have "locked down" the endpoints of the curve by this means. Of course, one need not lock down both endpoints.

Figure 5. Compare this example to Figure 3. In this case we have forced the endpoints of the curve to remain fixed in place.

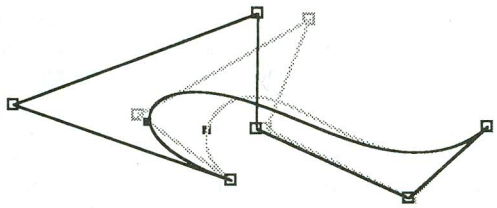

Figure 6. A re-rendering of the curve shown in Figure 5, except that the control vertices and control polygon are shown in addition to the curve. The initial state of the curve is shown in gray, and its final state in black.

Sometimes, of course, it *is* desirable to move the endpoints, so we adopt the convention that selecting a point on the curve within a small, fixed distance of the actual endpoint indicates that the endpoint itself is to be displaced by tracker movement, rather than a point on the interior of the curve.

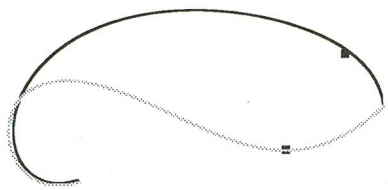

Figure 7. The initial fifth degree Bézier curve is shown in gray. The point marked with a black box is moved to a new position by computing new positions for the interior four control vertices using the fifth degree version of equations (e 6).

Equations (e 6) further suggest that we may, to some extent, control the locality of the changes produced by choosing how many control vertices we wish moved so that the curve will go through the new point. It seems most reasonable to select the control vertex with the largest influence on the point in question, namely the vertex scaling the basis function with the largest value at that point, and some number of its neighbors. Figure 7 and Figure 8 illustrate this for a fifth degree Bézier curve.

Figure 8. The gray curves here and in Figure 7 are identical, as are the initial and final positions of the black square. However, in this figure we compute a new position for only one control vertex to produce interpolation of the target position, while in Figure 7 four control vertices are moved.

Let us briefly consider multiple-segment Bézier curves. If the joints are fixed and we wish to maintain existing first or second derivative continuity, then we may use equations (e 4) or (e 5) to alter the segment containing the point being moved, and then from the usual continuity equations (see pages 214-215 in [Bartels], for example) compute the necessary movement in control vertices for the neighboring segment or segments. In Section 3 we will present a generalization of our technique that would enable us to compute altered coordinates for neighboring vertices in such a way as to maintain first or second degree continuity while moving joints as in Figure 3. However, it seems more natural in such circumstances to use a B-spline curve with an appropriate knot sequence, since for a cubic B-spline curve, second derivative continuity is preserved if there are no multiple knots, and knots of multiplicity two may be used to specify that only first derivative continuity be maintained at the corresponding joint.

*B-Splines*

Given control vertices $V_0$ through $V_m$, the resulting cubic B-spline curve $C(u) = (X(u),Y(u))$ of $m-2$ segments for the non-decreasing knot sequence $u_0, \ldots, u_{m+4}$ and $u_3 \le u < u_{m+1}$ is given by

$$X(u) = \sum_{i=0}^{m} x_i N_{i,4}(u)$$

$$Y(u) = \sum_{i=0}^{m} y_i N_{i,4}(u)$$

where the $i^{th}$ B-spline $N_{i,k}(u)$ of order $k$ is defined recursively by

$$N_{i,1}(u) = \begin{cases} 1 & u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+1} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u)$$

for $k \ge 2$. For cubics, in fact, each $N_{i,4}(u)$ is non-zero only for $u_i \le u < u_{i+4}$, so segment $i$ of the cubic B-spline curve is given by

$$X_i(u) = x_{i-3}N_{i-3,4}(u) + x_{i-2}N_{i-2,4}(u) + x_{i-1}N_{i-1,4}(u) + x_i N_{i,4}(u)$$

$$Y_i(u) = y_{i-3}N_{i-3,4}(u) + y_{i-2}N_{i-2,4}(u) + y_{i-1}N_{i-1,4}(u) + y_i N_{i,4}(u)$$

(It is convenient that the first segment of a cubic B-spline curve have index 3, as it is defined for $u_3 \le u < u_4$.)

Applying our technique to B-splines is straightforward: one uses exactly the same equations to compute new positions for the control vertices, and then generates the altered curve from the altered control vertices in the usual way. The four $B_i(u)$ in equations (e 1) and (e 4) are replaced by the four $N_{i,4}(u)$ that are non-zero for the segment of the B-spline curve containing the point being moved, new positions are computed for the corresponding control vertices, and the curve is redrawn. Figure 10[1] is a nine segment (twelve control vertex) uniform cubic B-spline curve; Figure 11 shows the effect of moving a point near the right hand end of this curve, using equations (e 4). It illustrates a potential problem, however:

[1] There is no Figure 9...

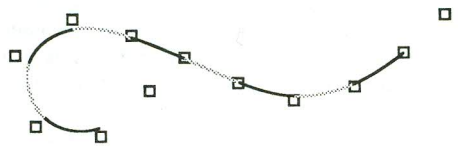

Figure 10. A nine segment, twelve control vertex uniform cubic B-spline curve. The control vertices are shown as squares, and the line pattern alternates from one segment of the curve to the next.

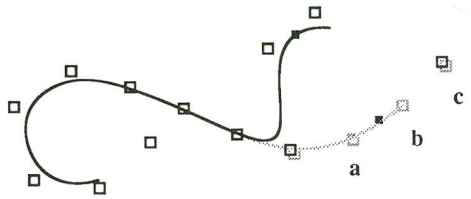


Figure 11. Starting from the curve of Figure 10 (shown in gray), the indicated point has been moved northwest. Only the rightmost four control vertices have moved.

the curve still points towards the very last control vertex, rightmost in Figure 11. It is easy to see why this is necessarily the case. Let **a**, **b** and **c** be the last three control vertices. Then the final point on the curve (see page 38 in [Bartels]) is found at

$$\mathbf{P}_e = \mathbf{a}\,N_9(u_e) + \mathbf{b}\,N_{10}(u_e) + \mathbf{c}\,N_9(u_e)$$

$$= \frac{1}{6}(\mathbf{a} + 4\mathbf{b} + \mathbf{c})$$

and the derivative $\mathbf{D}_e$ at $\mathbf{P}_e$ is

$$\mathbf{D}_e = 3\,(\mathbf{c} - \mathbf{a})$$

Moving a point in the last segment by applying equation (e 4) moves primarily **a** and **b** because the basis functions scaling them are largest. As the point moved approaches the left end of the final segment, **a** is moved more than **b**, but in both cases **c** remains nearly stationary because the B-spline weighting it is relatively small. We will return to this problem later, but in any case we want to be able to fix the endpoints of a cubic B-spline curve, and one of the mechanisms for doing so also avoids this problem.

Instead of uniform knot spacing, let us suppose that the first and final knots have multiplicity four, while the remaining breakpoints have multiplicity one and are uniformly spaced. Figure 12 shows such a curve, which for convenience we will refer to informally as an "fmek curve" (for "**f**ull **m**ultiplicity **e**nd **k**not curve").

Figure 12. A nine segment, twelve control vertex cubic B-spline fmek curve for the knot vector (0,0,0,0,1,2,3,4,5,6,7,8,9,9,9,9).

B-splines based on such knot vectors have two especially useful characteristics: they interpolate the first and last control vertices, and the derivatives at the beginning and end of the curve lie along the first and last legs of the control polygon. From the first property it follows that if we use variations of equations (e 4) which do not involve the initial or final vertex then the endpoints will remain fixed. By selecting a point near the end of the curve and rotating it about the end we can control the direction of the curve at the endpoint (illustrated in Figure 13). As was mentioned earlier, it is convenient to adopt the convention that selecting a point sufficiently close to an endpoint results in direct translation of the control vertex at that endpoint.

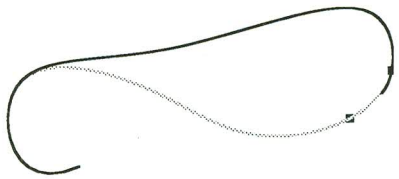

Figure 13. This is a four segment, seven control vertex cubic B-spline fmek curve. A three-vertex version of equations (e 5) not involving the last control vertex has been used to move the point represented by the black square, which lies on the last segment of the curve. It follows that the last vertex is not altered, so that the endpoint of the curve remains fixed, and the curve rotates around that endpoint.

Of course, we may choose not to lock down the endpoints of an fmek curve, and if we move a point sufficiently close to the end of the curve, the end will move also. Figure 14 illustrates this (and avoids the difficulty illustrated in Figure 11).

Figure 14. Here we have selected the same point, and using equations (e 4), moved it to the same position as we did in Figure 13. The last control vertex contributes substantially to the position moved, and so an entirely different alteration in the curve results.

Just as for Bézier curves, we may use a form of equations (e 4) involving anywhere between one and $k$ control vertices, where $k$ is the order of the B-spline. If we move one control vertex, then $k$ segments of the B-spline will be altered in order to interpolate the desired point, while if $k$ control vertices are moved then $k+(k-1)$ segments will be altered, so that to some extent we may control the locality of the change by selecting the number of control vertices involved.

Figure 15. A particular point on the curve of Figure 12 has been moved twice to the same new position, once by altering a single control vertex (resulting in the middle curve), and once by altering four control vertices (resulting in the upper curve). In the latter case, one additional segment to the left and two additional segments to the right change shape, although the alteration in the rightmost segment is not visible (partly because the point being moved is very close to a joint).

Movement of a single control vertex can result in "unbalanced" changes in shape, so it is probably a better idea to move two or more (see Figure 16): a decrease in the influence of one control vertex is balanced by an increase in the influence of another.

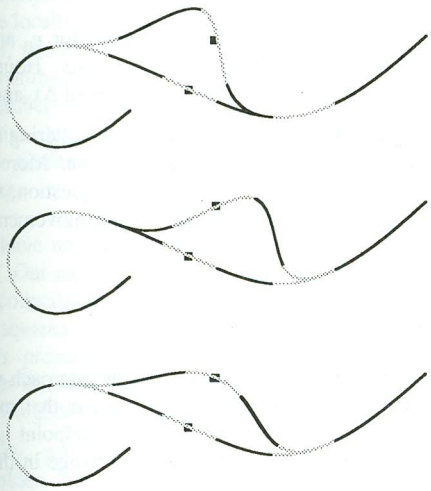

Figure 16. The points being moved in the top two curves lie ever so slightly left and right, respectively, of the parameter value for which the influence of the two closest control vertices are balanced. A single control vertex is being altered, and curves of distinctly different shape result. The bottom curve shows overlain the two curves that result from movement of the two points specified in the top two curves (as well as the initial curve), but with the alteration of two control vertices. In this case the two curves are nearly indistinguishable.

### 3. A Theoretical Justification

We will now provide a theoretical justification for equations (e 4)

$$\Delta x_i = \Delta x\, w_i = \Delta x \frac{B_i}{B_0^2+B_1^2+B_2^2+B_3^2}$$

$$\Delta y_i = \Delta y\, w_i = \Delta y \frac{B_i}{B_0^2+B_1^2+B_2^2+B_3^2}$$

and (e 6) by showing that they cause the curve to go through the new position by making a "minimal" change in the positions of the control vertices, in a sense that will become clear shortly.

Let us rewrite equations (e 3) as:

$$\begin{bmatrix} B_0 & B_1 & B_2 & B_3 \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \Delta x \qquad \text{(e 7)}$$

(We omit explicit discussion of $\Delta y$, the treatment of which is exactly analogous.) We make use of the *Householder transformation* [Lawson]

$$\mathbf{H} = \mathbf{I} - \tfrac{1}{\pi}\omega^T\omega$$

where

$$\sigma = \text{sign}(B_0)\sqrt{B_0^2+B_1^2+B_2^2+B_3^2}$$

$$\omega^T = \begin{bmatrix} (B_0+\sigma) \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

$$\omega = [\,(B_0+\sigma)\ B_1\ B_2\ B_3\,]$$

$$\pi = \sigma(B_0+\sigma)$$

As may be verified directly, $\mathbf{H}$ is an orthogonal matrix with the property that

$$[B_0\ B_1\ B_2\ B_3]\mathbf{H} = [-\sigma\ 0\ 0\ 0]$$

Because $\mathbf{H}$ is orthogonal, $\mathbf{H}^{-1} = \mathbf{H}^T$ so that $\mathbf{H}\mathbf{H}^T$ is the identity matrix. Hence equation (e 7) may be written as

$$[B_0\ B_1\ B_2\ B_3]\,\mathbf{H}\,\mathbf{H}^T \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \Delta x$$

Let $[e_0,e_1,e_2,e_3]^T = \mathbf{H}^T[\Delta x_0,\Delta x_1,\Delta x_2,\Delta x_3]^T$. Then we may write this equation as

$$[-\sigma\ 0\ 0\ 0] \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \Delta x$$

Clearly $e_0 = -\Delta x/\sigma$, but $e_1$, $e_2$, and $e_3$ may have any values whatsoever. Because $\mathbf{H}$ is an orthogonal matrix, we may think of it as accomplishing a change of coordinate systems that preserves Euclidian distance, so that $[e_0,e_1,e_2,e_3]^T$ and $[\Delta x_0,\Delta x_1,\Delta x_2,\Delta x_3]^T$ have the same length. Both are vectors representing the change made to the control vertices. So as to minimize this change, we let $e_1 = e_2 = e_3 = 0$. Finally, we undo the Householder transformation to compute the $\Delta x_i$. Thus

$$\begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \mathbf{H} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \left( \mathbf{I} - \frac{1}{\sigma (B_0+\sigma)} \begin{bmatrix} (B_0+\sigma) \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \begin{bmatrix} (B_0+\sigma) & B_1 & B_2 & B_3 \end{bmatrix} \right) \cdot \begin{bmatrix} -\Delta x/\sigma \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Straightforward algebra then yields

$$\begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \frac{\Delta x}{B_0^2 + B_1^2 + B_2^2 + B_3^2} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

which is simply another representation of equations (e 4).

Moreover, the above argument in no way depended on our having involved all the control vertices contributing to the position of the point being moved, so that modifications such as that of (e 6) also minimize the total change of the control vertices being allowed to move.

## 4. A Generalization

Repeated Householder transformations can be used to find a minimum length solution to a system of several underdetermined equations. For example, let us suppose that we wish to move a point $C(u)$ in the first segment of a floating B-spline curve over a uniform knot vector (such as the curve of Figure 10), *without alteration to the initial point of the curve* (see Figure 17 for an example).

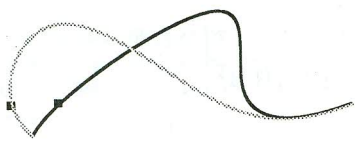

Figure 17. This is a six control vertex uniform cubic B-spline curve. There were two equations involved in moving the point shown as a black square: one yielded its new position, and the other enforced the constraint that the change accomplishing the first goal left the initial, leftmost point of the curve unchanged.

The appropriate equations are as follows:

$$\begin{bmatrix} B_0(u) & B_1(u) & B_2(u) & B_3(u) \\ B_0(u_3) & B_1(u_3) & B_2(u_3) & B_3(u_3) \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \end{bmatrix} = \begin{bmatrix} \Delta x \\ 0 \end{bmatrix}$$

Insertion of the first Householder transformation leaves us with

$$\begin{bmatrix} -\sigma & 0 & 0 & 0 \\ \beta_0 & \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \Delta x \\ 0 \end{bmatrix}$$

where $[\ \beta_0\ \beta_1\ \beta_2\ \beta_3\ ] = [\ B_0(u_3)\ B_1(u_3)\ B_2(u_3)\ B_3(u_3)\ ]\mathbf{H}$. We then insert a 3x3 Householder transformation $\mathbf{H}$ into the system

$$[\ \beta_1 \quad \beta_2 \quad \beta_3\ ] \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = 0$$

to obtain the system

$$[\ \beta_1 \quad \beta_2 \quad \beta_3\ ] \widehat{\mathbf{H}}\, \widehat{\mathbf{H}}^{\mathrm{T}} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = [-\tau\ 0\ 0\ ] \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = 0$$

Next embed $\widehat{\mathbf{H}}$ into a 4x4 matrix as

$$\widehat{\mathbf{H}}_4 = \begin{bmatrix} 1 & 0 \\ 0 & \widehat{\mathbf{H}} \end{bmatrix}$$

and combine the two transformations into $(\mathbf{H}\widehat{\mathbf{H}})(\widehat{\mathbf{H}}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})$ to obtain the system

$$\begin{bmatrix} -\sigma & 0 & 0 & 0 \\ \beta_0 & -\tau & 0 & 0 \end{bmatrix} \begin{bmatrix} e_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} \Delta x \\ 0 \end{bmatrix} \qquad \text{(e 8)}$$

This time we set $f_2 = f_3 = 0$ and solve for $e_0$ and $f_1$ by forward substitution. Inverting the two Householder transformations then yields the desired values of $\Delta x_i$ and $\Delta y_i$.

After solving for $\Delta x$, we may obtain $\Delta y$ by altering the right hand side and repeating the forward substitution. Moreover, so long as we do not change the value of $u$ in question, repeated forward substitution suffices for subsequent movement of the tracker.

## 5. Discussion

While promising, and indeed useful, the approach we have presented has its limitations. One problem is that moving a point close to the end of a curve when the endpoint is locked down can produce an unexpectedly large change in the shape of the curve (see Figure 18).

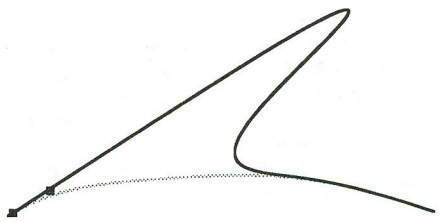

Figure 18. The leftmost black square records a position selected on the original, gray curve that is very close to the endpoint, which is not allowed to move. The black square to the right and above it marks the position to which the tracker was moved in producing the solid black curve.

What's happening is that the new curve is required to interpolate the new location for the same, fixed value of $u$. Because the point selected is very close to the beginning of the curve, which does not move, the new curve must cover a relatively large Euclidian distance while undergoing a relatively small change in $u$, resulting in the large overshoot shown.

A related phenomenon has to do with the movement of a selected point along a curve, which changes the relationship between the independent parameter $u$, arc length, the length of individual spline segments, and the portion of the curve that is altered when interaction occurs — we would prefer that the locality of movement be independent of the point being moved, but this may not be the case. Of course, the same changes can occur when moving control vertices directly, although the control vertex positions provide some indication of what is happening. For direct manipulation, one might use color or gray scale to distinguish consecutive segments, or space small beads uniformly in parameter space, to provide a visual indication of segment length and the resulting locality. A more interesting possibility we are exploring is to use the component of tracker motion along the curve to alter the parameter value $u$ for which the curve is required to interpolate the new location.

Another interesting possibility is to combine knot insertion, hierarchical B-spline refinement (introduced by [Forsey]), and direct manipulation so that a user may either cycle among available localities or specify exactly the subcurve that is to be reshaped, retaining the ability to subsequently perform more global manipulations. One might also alter the positions of some number of control vertices neighboring those moved to accomplish the interpolation, along the lines of the "warp" operator discussed in [Cobb], to widen the extent of the reshaping.

We have not yet discussed how the curve should first be entered. Our method of choice is to make an initial free-hand sketch, collecting data points every few pixels and performing a least squares fit in which the parametric spacing of data points is proportional to the Euclidian distance separating them. In the resulting curves the independent parameter u is generally a good approximation of arc length.

Finally, we have only begun to explore the utility of Householder transformations in reshaping curves while enforcing auxiliary constraints. Note also that equations (e 4) suggest other non-minimal, though perhaps useful, weighting schemes.

## 6. Acknowledgments

This work was supported by grants from the National Science and Engineering Council of Canada, by Digital Equipment of Canada, and by the Information Technology Research Centre.

## 7. References

Bartels — Richard H. Bartels, John C. Beatty and Brian A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling,* Morgan Kaufman Publishers, Inc., Los Altos, California, 1987.

Cobb — Elizabeth Susan Cobb, *Design of Sculptured Surfaces Using the B-Spline Representation,* PhD Dissertation, Department of Computer Science, University of Utah, June 1984.

Farin — Gerald Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1988.

Faux — Ivor D. Faux and Michael J. Pratt, *Computational Geometry for Design and Manufacture,* Ellis Horwood Limited, Chichester, England 1979.

Forsey — David R. Forsey and Richard H. Bartels, "Hierarchical B-Spline Refinement," *Computer Graphics*, 22:4 (August 1988), pp. 205-212 (proceedings of the Siggraph 88 Conference).

Lawson — Charles L. Lawson and Richard J. Hanson, *Solving Least Squares Problems,* Prentice Hall, 1974.

Mortenson — Michael E. Mortenson, *Geometric Modeling,* John Wiley and Sons, New York, 1985.