CHAPTER
ELEVEN

Shape Modification Tools

11.1 Introduction

The purpose of this chapter is to develop *tools* which allow a designer to *interactively* make *local* modifications to an *existing* NURBS curve or surface, in a way that is *natural* and *intuitive*. A NURBS curve or surface is defined by its control points, weights, and knots; modifying any of these changes the shape of the curve or surface. Generally designers do not want to work with such concepts; they prefer to specify constraints and shapes which apply directly to the curve or surface. In this chapter we develop tools which take such constraints and shape specifications and convert them into modifications in control point locations or weight values. Specifically, Sections 11.2 and 11.3 are based on specifying point constraints (i.e., the curve or surface is to change locally so that it passes through a given point); multiple point and derivative constraints are handled in Section 11.5, allowing control of tangent and curvature; and the methods of Section 11.4 allow a designer to specify the "shape" that a curve or surface is to assume in a selected region. These geometrically intuitive inputs are converted into control point relocations in Sections 11.2, 11.4, and 11.5, and into weight modifications in Section 11.3. The influence of control points and weights on shape is geometrically clear and intuitive. This was already shown, qualitatively, in previous figures, e.g., Figures 3.7, 3.23, 4.4, 4.6, 4.11, and 4.12; this chapter quantifies this influence. Although knot locations also affect shape, we know of no geometrically intuitive or mathematically simple interpretation of this effect, hence we present no techniques for moving knot locations.

The methods of this chapter are local, due to the local support property of B-splines. However, the extent of the curve or surface region which is to change can be made smaller by knot refinement, or larger by knot removal. Interactivity requires fast solutions, which in turn implies either simple equations, small linear systems, or a multistage solution in which the critical interactive stage is fast. The methods of Sections 11.2 and 11.3 are simple and very fast. The shape operators of Section 11.4 require knot refinement and knot removal as pre- and postprocessing steps, respectively. The technique in Section 11.5 requires inversion of a (usually small) matrix as a preprocessing step. We develop the mathematics of this chapter in detail, but we give only sketches of possible user interaction and algorithm organization. User interface and graphical interaction are beyond the scope of this book, and algorithm organization can depend on these as well as on the application area.

Many interesting topics which fall under the broad category of "shape control and modification" have been left out of this chapter – one has to draw the line somewhere. The body of literature is growing in the area known as "fairing" or "smoothing". There is no universal agreement on definitions for these concepts, but they generally involve properties such as geometric continuity, inflection points, convexity, and curvature. Many solution techniques in this area lead to constrained nonlinear minimization problems. Consequently, it is difficult to solve them at interactive speeds. We refer the reader to books by Su and Liu [Su89] and Hoschek and Lasser [Hosc93] and the many references therein. Welch and Witkin [Welc92] describe a method of surface design in which they minimize the integral of the weighted sum of the first and second fundamental forms over the surface. These forms measure how the surface is stretched and bent [DoCa76]. The designer can specify not only point and derivative, but also transfinite constraints (e.g., embedded curves). Their method leads to constrained minimization of a quadratic objective function with linear constraints.

A technique called Free-Form Deformation (FFD) can be useful for rough shaping, deforming, and sculpting of collections of curves and surfaces (which can form a closed object, for example). The geometry is considered to be embedded in a three-dimensional plastic environment, which in turn is defined by a trivariate, tensor product Bézier or B-spline function (mapping (u, v, w) parameters into (x, y, z) space). The surrounding plastic is deformed by moving its control points, $\mathbf{P}_{i,j,k}$, and the object geometry inherits this deformation. Further reading on FFDs can be found in [Sede86; Coqu90, 91; Hsu92].

B-spline surfaces (curves) can be superimposed to form a new surface (curve). In other words, if a collection of surfaces, $\mathbf{S}_1, \ldots, \mathbf{S}_n$, are all defined on the same u and v knot vectors, then a new surface is defined by the same knot vectors and the control points

$$\mathbf{P}_{i,j} = \alpha_1 \mathbf{P}^1_{i,j} + \cdots + \alpha_n \mathbf{P}^n_{i,j}$$

where $\{\alpha_k\}$ are scalars and $\{\mathbf{P}^k_{i,j}\}$ are the control points defining $\mathbf{S}_k$. The NURBS construction of Gordon and Coons surfaces are examples of this (see Eqs. [10.33] and [10.43]). This suggests the use of "additive" or "corrective" surfaces or surface patches. One can work with small, simple pieces and then superimpose them to form a new surface or to modify a region of an existing surface. Boundary conditions are easily controlled. Several authors have used such techniques for interactive surface design (see [Fors88; Mats92; Welc92]).

11.2 Control Point Repositioning

Let $\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i$ be a rational or nonrational B-spline curve. We want to reposition an arbitrary control point, $\mathbf{P}_k$, $0 \le k \le n$. Denote the new, translated control point by $\hat{\mathbf{P}}_k$, and the translation vector by $\mathbf{V} = \hat{\mathbf{P}}_k - \mathbf{P}_k$. Then the new curve, $\hat{\mathbf{C}}(u)$, is given by

$$\begin{aligned}\hat{\mathbf{C}}(u) &= R_{0,p}(u)\mathbf{P}_0 + \cdots + R_{k,p}(u)(\mathbf{P}_k + \mathbf{V}) + \cdots + R_{n,p}(u)\mathbf{P}_n \\ &= \mathbf{C}(u) + R_{k,p}(u)\mathbf{V} \qquad (11.1)\end{aligned}$$

Equation (11.1) expresses a functional translation of all curve points, $\mathbf{C}(u)$, for which $u \in [u_k, u_{k+p+1})$; all curve points outside this interval are unaffected. The maximum translation occurs at the maximum of the function $R_{k,p}(u)$. This is illustrated in Figure 11.1.

Now suppose $\bar{u}$ is a fixed parameter value with $\bar{u} \in [u_k, u_{k+p+1})$, and $\mathbf{P} = \mathbf{C}(\bar{u})$. If we want to move $\mathbf{P}$ a distance d in the direction $\mathbf{V}$ by repositioning $\mathbf{P}_k$, then we must determine the α such that

$$\hat{\mathbf{P}}_k = \mathbf{P}_k + \alpha\mathbf{V} \qquad (11.2)$$

effects the desired translation. Let $\hat{\mathbf{P}} = \hat{\mathbf{C}}(\bar{u})$. Clearly

$$|\hat{\mathbf{P}} - \mathbf{P}| = d = \alpha|\mathbf{V}|R_{k,p}(\bar{u})$$

which yields

$$\alpha = \frac{d}{|\mathbf{V}|R_{k,p}(\bar{u})} \qquad (11.3)$$

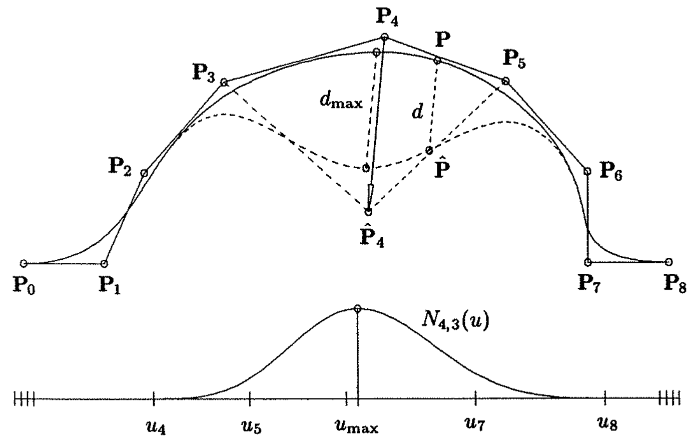


Figure 11.1. Moving $\mathbf{P} = \mathbf{C}(\bar{u} = 3/5)$ by repositioning $\mathbf{P}_4$; the corresponding node is $t_4 = 0.52$, and the curve degree is three.

From a user interface point of view, there are two broad approaches to control point repositioning, distinguished by whether the user picks on the control polygon or the curve. In the first scenario, the user might specify a vector direction and distance and then pick a point on the control polygon. If the picked point is an existing control point, the system simply translates it and redraws the curve. If not, then the system makes the point a control point through inverse knot insertion (see Chapter 5, Eqs. [5.19]–[5.21]), and it subsequently translates the new control point and redraws the curve. In the second scenario, the designer picks a curve point, $\mathbf{P}$ (from which the system computes $\bar{u}$), and then picks the point $\hat{\mathbf{P}}$ to which $\mathbf{P}$ is to be moved (from which the system computes d and $\mathbf{V}$). It remains for the system to choose a control point, $\mathbf{P}_k$, to be translated in order to produce the desired movement of $\mathbf{P}$. In general there are up to $p+1$ candidates, but it is usually desirable to choose $\mathbf{P}_k$ so that $R_{k,p}(u)$ is the basis function whose maximum lies closest to $\bar{u}$. A parameter value which generally lies quite close to (but not necessarily coincident with) the maximum of an arbitrary basis function, $R_{i,p}(u)$, is the so-called *node*, t_i, defined by (see [Ries73])

$$t_i = \frac{1}{p}\sum_{j=1}^{p} u_{i+j} \qquad i = 0,\ldots,n \tag{11.4}$$

Note that t_i is the average of the knots which are interior to the domain where $R_{i,p}(u)$ is nonzero. For interactive control point repositioning, we recommend choosing k based on the node values instead of actually computing the maximums of the $R_{i,p}(u)$; that is, k is chosen such that

$$|\bar{u} - t_k| = \min_i |\bar{u} - t_i| \tag{11.5}$$

Of course, the nodes can be used as start values to find the maximums using Newton iteration.

Figures 11.2 and 11.3 show two control point repositionings in which the index k was chosen based on the proximity of $\bar{u}$ to the nodes. Notice that Figure 11.2 is pleasing, while Figure 11.3 is asymmetric and not pleasing. The reason is that in Figure 11.2 $\bar{u}$ is very close to the node t_k (and to the maximum of $R_{k,p}(u)$), therefore the curve translation is rather symmetrical about $\mathbf{C}(\bar{u})$ and roughly maximum there. In Figure 11.3, $\bar{u}$ lies halfway between t_k and t_{k+1} (but t_k was chosen), therefore the maximum translation occurs to the left of $\mathbf{C}(\bar{u})$. There are two solutions to this dilemma:

- insert a knot in order to put a node at $\bar{u}$;
- reposition both $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$.

Let us insert the knot $\hat{u}$ which forces t_{k+1} (the new $(k+1)$th node) to have the value $\bar{u}$. From Eq. (11.4) it is clear that $u_{k+1} < \hat{u} < u_{k+p+1}$, hence

$$\bar{u} = t_{k+1} = \frac{1}{p}\left(\hat{u} + \sum_{j=2}^{p} u_{k+j}\right)$$

and

$$\hat{u} = p\bar{u} - \sum_{j=2}^{p} u_{k+j} \tag{11.6}$$

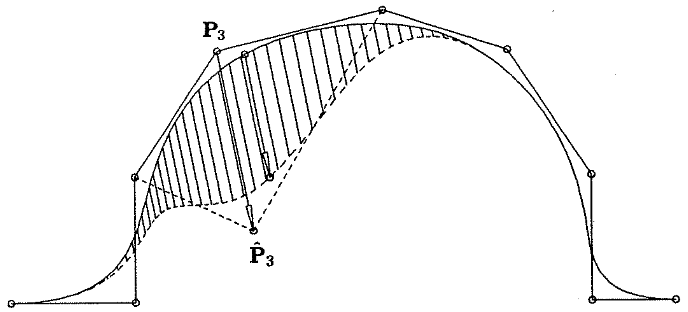


Figure 11.2. Moving $\mathbf{P} = \mathbf{C}(\bar{u} = 0.37)$ by repositioning $\mathbf{P}_3$; the corresponding node is $t_3 = 0.35$, and the curve degree is three. Notice the parallel translation of a segment of the curve.

Figure 11.4 shows the curve of Figure 11.3. The knot $\hat{u}$ from Eq. (11.6) was inserted to force $\bar{u}$ to be the $(k+1)$th node. The resulting $\mathbf{Q}_{k+1}$ was then moved to effect the desired curve modification. Notice the symmetry and the decreased domain of change. We caution the reader that $\hat{u}$ may already be a knot, and it may be undesirable to insert it. For example, let $\mathbf{C}(u)$ be a quadratic curve on $U = \{0,0,0,{}^1/_4,{}^1/_2,{}^3/_4,1,1,1\}$ (see Figure 11.5). The nodes are $t_i = \{0,{}^1/_8,{}^3/_8,{}^5/_8,{}^7/_8,1\}$. If the designer picks the point for which $\bar{u} = {}^1/_2$, then by Eq. (11.6) $\hat{u} = {}^1/_2$. If $\hat{u} = {}^1/_2$ is inserted then $\bar{u}$ becomes the node t_3, and if $\mathbf{Q}_3$ $(= \mathbf{C}({}^1/_2))$ is moved, the resulting curve has a cusp at $u = {}^1/_2$, as shown in Figure 11.5.

Suppose we now want to move $\mathbf{P} = \mathbf{C}(\bar{u})$ along $\mathbf{V}$ a distance d by translating both $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$ in the direction $\mathbf{V}$ (assuming $\bar{u} \in [u_{k+1}, u_{k+p+1})$). Letting

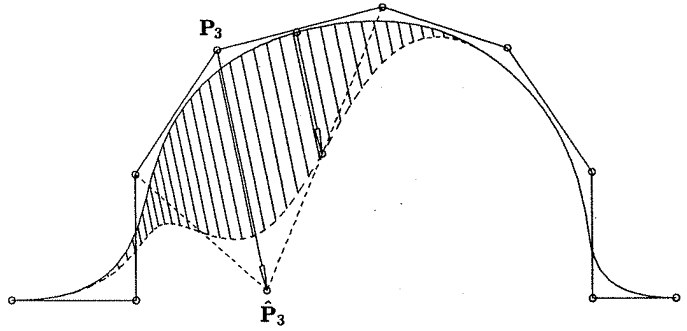


Figure 11.3. Moving $\mathbf{P} = \mathbf{C}(\bar{u})$ by repositioning $\mathbf{P}_3$. $\bar{u} = 0.43 = {}^1/_2(t_3 + t_4)$; the corresponding node is $t_3 = 0.35$, and the curve degree is three.

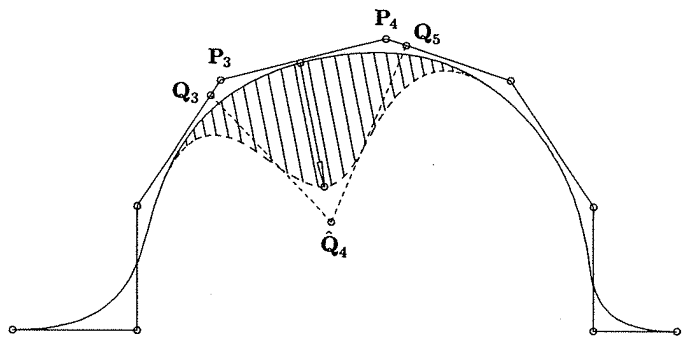


Figure 11.4. Repositioning a new control point obtained by knot insertion; the curve degree is three.

$0 \le \gamma \le 1$, we set

$$\hat{\mathbf{P}}_k = \mathbf{P}_k + (1-\gamma)\alpha\mathbf{V}$$

$$\hat{\mathbf{P}}_{k+1} = \mathbf{P}_{k+1} + \gamma\alpha\mathbf{V} \tag{11.7}$$

The parameter γ allows us to vary, on a linear scale, the movement of $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$ relative to one another. The designer can set γ, or a good default is

$$\gamma = \frac{\bar{u} - t_k}{t_{k+1} - t_k} \tag{11.8}$$

where $t_k \le \bar{u} \le t_{k+1}$. From translational invariance (P4.9) we have

$$|\hat{\mathbf{P}} - \mathbf{P}| = d = |(1-\gamma)\alpha\mathbf{V}R_{k,p}(\bar{u}) + \gamma\alpha\mathbf{V}R_{k+1,p}(\bar{u})|$$

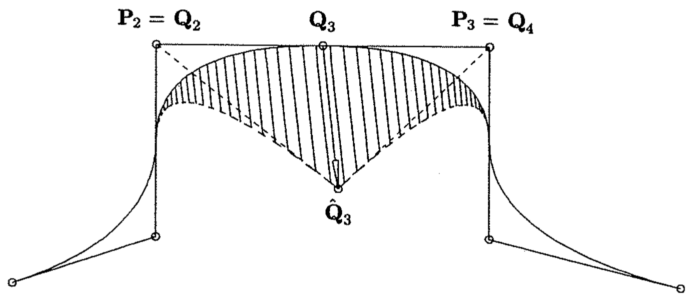


Figure 11.5. Repositioning a new control point obtained by knot insertion; the curve degree is two.

which implies that

$$\alpha = \frac{d}{|\mathbf{V}|\,[(1-\gamma)R_{k,p}(\bar{u}) + \gamma R_{k+1,p}(\bar{u})]} \tag{11.9}$$

Figure 11.6a shows the curve of Figure 11.3. Both $\mathbf{P}_3$ and $\mathbf{P}_4$ were moved, with $\gamma = 0.47$ (the default computed by Eq. [11.8]). Clearly, the disadvantage with this method is the broader domain of change, namely $[u_k, u_{k+p+2})$. In Figure 11.6b $\mathbf{P}_3$ and $\mathbf{P}_4$ were moved, with $\gamma = 4/5$.

Using these tools, it is easy to implement a user-friendly, interactive, graphical interface for control point repositioning. Notice that the value $R_{k,p}(\bar{u})$ in Eq. (11.3) does not change when $\mathbf{P}_k$ is repeatedly translated along the same vector direction (therefore neither does α). Furthermore, only that portion of the curve which is the image of $[u_k, u_{k+p+1})$ need be graphically updated for each

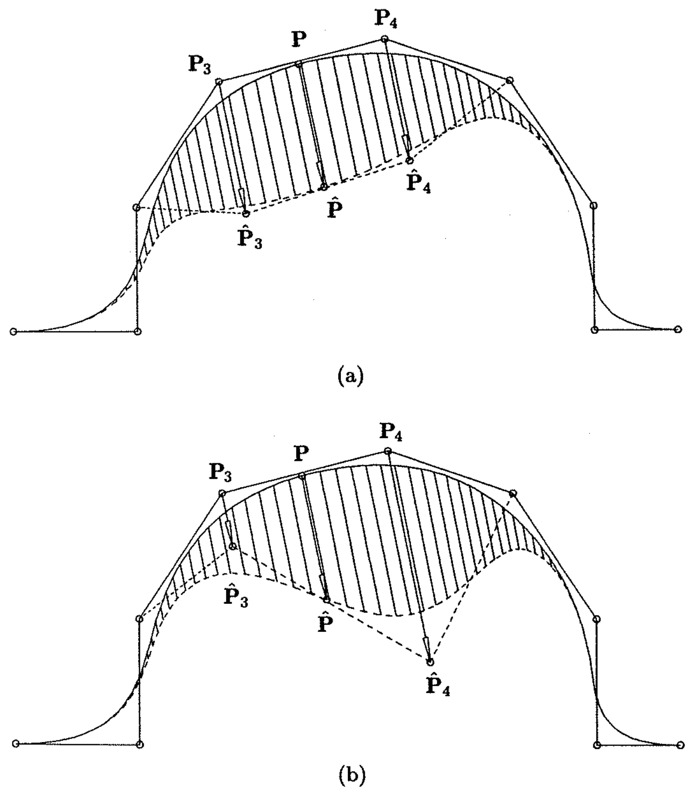


Figure 11.6. Simultaneous repositioning of $\mathbf{P}_3$ and $\mathbf{P}_4$; $\bar{u} = 0.43$. (a) $\gamma = 0.47$ (the default); (b) $\gamma = 4/5$.

repositioning. Figure 11.7 shows incremental repositioning of one control point. An increment, Δd, is specified, which fixes a $\Delta\alpha$. The relevant control point is then translated an increment $\Delta\alpha\mathbf{V}$, and the relevant curve segment is redrawn with each activation of a specified graphical input.

Finally, a designer can easily restrict the domain on which the curve should change by picking two more curve points, $\mathbf{Q}_1$ and $\mathbf{Q}_2$, which determine parameters $\bar{u}_1$ and $\bar{u}_2$, satisfying $\bar{u}_1 < \bar{u} < \bar{u}_2$. If $\bar{u}_1$ and $\bar{u}_2$ are not already knots, they are inserted one time each. Then, if necessary, additional knots are inserted on the intervals $[\bar{u}_1, \bar{u})$ and $[\bar{u}, \bar{u}_2)$ to guarantee that there is a suitable control point, $\mathbf{P}_k$, such that $\bar{u}$ is close to the node t_k and $\bar{u}_1 \le u_k$ and $u_{k+p+1} \le \bar{u}_2$. Knots should not be inserted with multiplicities greater than one, because this can cause unwanted discontinuities when control points are moved.

These techniques extend easily to surfaces. For completeness we state the relevant formulas. Let

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,p;j,q}(u,v)\,\mathbf{P}_{i,j}$$

be a NURBS surface, and assume that $\mathbf{P} = \mathbf{S}(\bar{u},\bar{v})$ is to be translated along a vector, $\mathbf{V}$, a distance d by repositioning the control point, $\mathbf{P}_{k,\ell}$. Then

$$\hat{\mathbf{P}}_{k,\ell} = \mathbf{P}_{k,\ell} + \alpha\mathbf{V} \tag{11.10}$$

with

$$\alpha = \frac{d}{|\mathbf{V}|R_{k,\ell}(\bar{u},\bar{v})} \tag{11.11}$$

Either the designer selects k and ℓ, or the system chooses them based on the proximity of $\bar{u}$ and $\bar{v}$ to the u and v nodes, s_i and t_j, defined by

$$s_i = \frac{1}{p}\sum_{r=1}^{p} u_{i+r} \qquad i = 0,\ldots,n$$

$$t_j = \frac{1}{q}\sum_{r=1}^{q} v_{j+r} \qquad j = 0,\ldots,m \tag{11.12}$$

Using formulas analogous to Eq. (11.6), a u or v knot, or both, can be computed and inserted in order to decrease the domain of change and to bring $\bar{u}$ and/or $\bar{v}$ to nodal positions. If $0 \le \delta,\gamma \le 1$, then $\mathbf{P}_{k,\ell}$, $\mathbf{P}_{k+1,\ell}$, $\mathbf{P}_{k,\ell+1}$, and $\mathbf{P}_{k+1,\ell+1}$ can be translated simultaneously in the direction $\mathbf{V}$ by

$$\hat{\mathbf{P}}_{k,\ell} = \mathbf{P}_{k,\ell} + (1-\delta)(1-\gamma)\alpha\mathbf{V}$$

$$\hat{\mathbf{P}}_{k+1,\ell} = \mathbf{P}_{k+1,\ell} + \delta(1-\gamma)\alpha\mathbf{V}$$

$$\hat{\mathbf{P}}_{k,\ell+1} = \mathbf{P}_{k,\ell+1} + (1-\delta)\gamma\alpha\mathbf{V}$$

$$\hat{\mathbf{P}}_{k+1,\ell+1} = \mathbf{P}_{k+1,\ell+1} + \delta\gamma\alpha\mathbf{V} \tag{11.13}$$

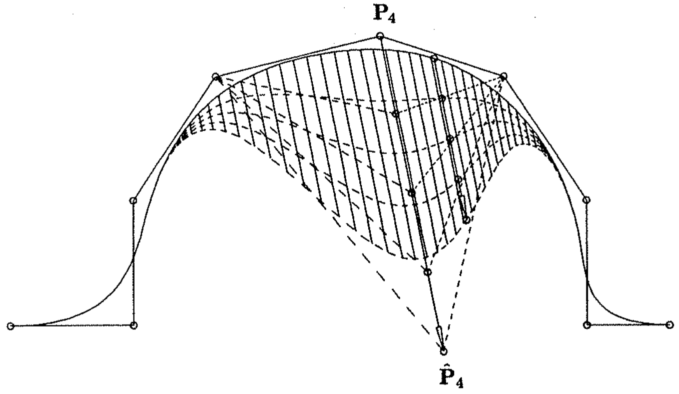
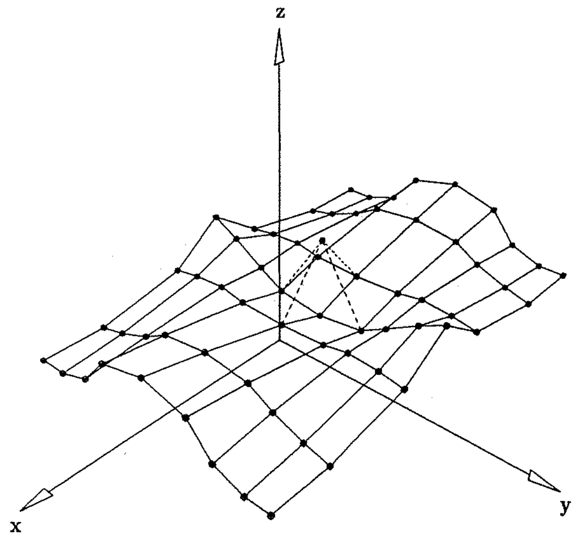


Figure 11.7. Pulling the curve with equal increments by repositioning $\mathbf{P}_4$; $\bar{u} = 3/5$.

with

$$\alpha = \frac{d}{|\mathbf{V}|\,[\,(1-\delta)(1-\gamma)R_{k,\ell}(\bar{u},\bar{v}) + \cdots + \delta\gamma R_{k+1,\ell+1}(\bar{u},\bar{v})\,]} \tag{11.14}$$

Figures 11.8a and 11.8b show translation of one control point, and Figures 11.9a and 11.9b illustrate simultaneous movement of four neighboring control points.



(a)

Figure 11.8. Repositioning $\mathbf{P}_{4,3}$, with $\bar{u} = \bar{v} = 0.55$. (a) Control net; (b) a surface with a local patch translated parallel to the direction of the control point movement.

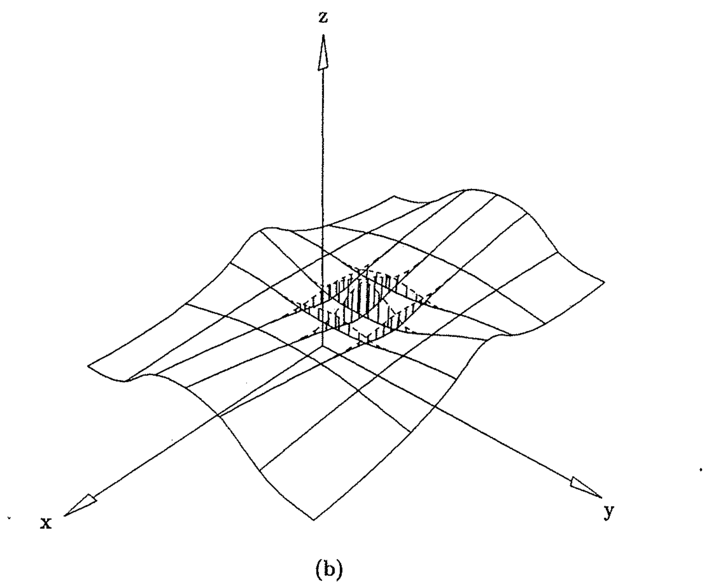


Figure 11.8. (*Continued.*)

Analogous to the curve case, knot insertion can be used to restrict the rectangular domain on which the surface is to change, and inverse knot insertion can be used to obtain additional control points at specified locations on the edges of the surface's control net. Point $\mathbf{Q}_u$, in the u direction, is inserted by the formula

$$\hat{u} = u_{k+1} + \alpha(u_{k+p+1} - u_{k+1}) \tag{11.15}$$

with
$$\alpha = \frac{w_{k,\ell}|\mathbf{Q}_u - \mathbf{P}_{k,\ell}|}{w_{k,\ell}|\mathbf{Q}_u - \mathbf{P}_{k,\ell}| + w_{k+1,\ell}|\mathbf{P}_{k+1,\ell} - \mathbf{Q}_u|}$$

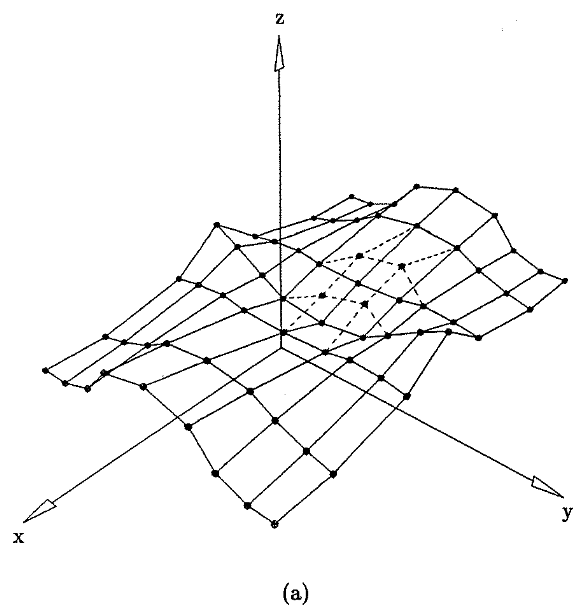
(see [Pieg89d]). Similarly, $\mathbf{Q}_v$ is inserted by

$$\hat{v} = v_{\ell+1} + \beta(v_{\ell+q+1} - v_{\ell+1}) \tag{11.16}$$

with
$$\beta = \frac{w_{k,\ell}|\mathbf{Q}_v - \mathbf{P}_{k,\ell}|}{w_{k,\ell}|\mathbf{Q}_v - \mathbf{P}_{k,\ell}| + w_{k,\ell+1}|\mathbf{P}_{k,\ell+1} - \mathbf{Q}_v|}$$

11.3 Weight Modification

In this section we investigate the effect of modifying weights, in particular one or two neighboring weights for a curve, and one weight for a surface.



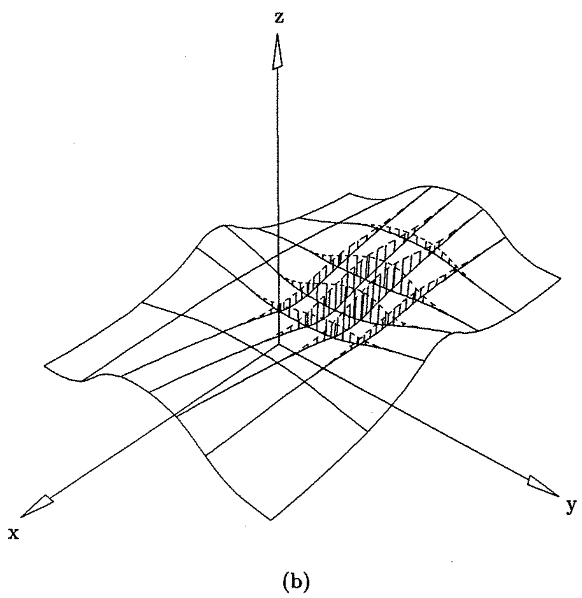


Figure 11.9. Repositioning four surface control points, $\mathbf{P}_{4,3}, \mathbf{P}_{5,3}, \mathbf{P}_{4,4}$, and $\mathbf{P}_{5,4}$, with $\bar{u} = \bar{v} = 3/5$. (a) Control net; (b) a surface with a local patch translated parallel to the direction of the control point movement.

11.3.1 MODIFICATION OF ONE CURVE WEIGHT

Let

$$\mathbf{C}(u)=\sum_{i=0}^{n}R_{i,p}(u)\mathbf{P}_i \tag{11.17}$$

be a NURBS curve and $\bar{u}$ a fixed parameter value satisfying $\bar{u}\in[u_k,u_{k+p+1})$. We pointed out in Chapter 4 that the effect of modifying the value of the weight w_k is to move the point $\mathbf{C}(\bar{u})$ along a straight line defined by $\mathbf{P}_k$ and the original point, $\mathbf{C}(\bar{u})$ (see Figures 4.2, 4.4, and 4.6). Hence, whereas control point repositioning has a translational effect on a curve, weight modification has a perspective effect, where all curve points in the affected domain are pulled (or pushed) along straight lines which meet at the control point corresponding to the weight being modified. In this section we prove this statement, and we establish the precise geometric meaning of the weights.

The weight w_k can be considered a variable in Eq. (11.17), hence we can differentiate $\mathbf{C}(u)$ with respect to w_k. Using the identity

$$R_{k,p}(u)=\frac{N_{k,p}(u)w_k}{\sum_{r=0}^{n}N_{r,p}(u)w_r}$$

and the quotient rule for differentiation, we obtain

$$\begin{aligned}\frac{\partial}{\partial w_k}\mathbf{C}(u;w_0,\ldots,w_n)&=\frac{\partial}{\partial w_k}\sum_{i=0}^{n}R_{i,p}(u)\mathbf{P}_i\\&=\sum_{i\neq k}\frac{-N_{i,p}(u)w_iN_{k,p}(u)}{\left(\sum_{r=0}^{n}N_{r,p}(u)w_r\right)^2}\mathbf{P}_i\\&\quad+\frac{N_{k,p}(u)\left(\sum_{r=0}^{n}N_{r,p}(u)w_r\right)-N_{k,p}(u)N_{k,p}(u)w_k}{\left(\sum_{r=0}^{n}N_{r,p}(u)w_r\right)^2}\mathbf{P}_k\\&=\sum_{i\neq k}\frac{-N_{k,p}(u)}{\sum_{r=0}^{n}N_{r,p}(u)w_r}R_{i,p}(u)\mathbf{P}_i\\&\quad+\frac{R_{k,p}}{w_k}\mathbf{P}_k-\frac{N_{k,p}(u)R_{k,p}(u)}{\sum_{r=0}^{n}N_{r,p}(u)w_r}\mathbf{P}_k\end{aligned}$$

$$\begin{aligned}&=\frac{R_{k,p}(u)}{w_k}\left(\mathbf{P}_k-\sum_{r=0}^{n}R_{r,p}(u)\mathbf{P}_r\right)\\&=\frac{R_{k,p}(u)}{w_k}\big(\mathbf{P}_k-\mathbf{C}(u)\big)=\alpha(k,u)\big(\mathbf{P}_k-\mathbf{C}(u)\big)\end{aligned} \tag{11.18}$$

Hence, the first derivative of $\mathbf{C}(u)$ with respect to w_k is a vector pointing in the direction from $\mathbf{C}(u)$ to $\mathbf{P}_k$. We now prove by induction that the derivatives of all orders have this form. Let $\alpha^{(1)}=\alpha(k,u)$, and assume the nth derivative has the form $\alpha^{(n)}\big(\mathbf{P}_k-\mathbf{C}(u)\big)$. Then

$$\begin{aligned}\frac{\partial^{n+1}\mathbf{C}(u)}{\partial w_k^{n+1}}&=\frac{\partial\left[\alpha^{(n)}\big(\mathbf{P}_k-\mathbf{C}(u)\big)\right]}{\partial w_k}\\&=\alpha^{(n)}\big(-\alpha^{(1)}\mathbf{P}_k+\alpha^{(1)}\mathbf{C}(u)\big)+\frac{\partial\alpha^{(n)}}{\partial w_k}\big(\mathbf{P}_k-\mathbf{C}(u)\big)\\&=\left(\frac{\partial\alpha^{(n)}}{\partial w_k}-\alpha^{(1)}\alpha^{(n)}\right)\big(\mathbf{P}_k-\mathbf{C}(u)\big)=\alpha^{(n+1)}\big(\mathbf{P}_k-\mathbf{C}(u)\big)\end{aligned}$$

where $$\alpha^{(n+1)}=\frac{\partial\alpha^{(n)}}{\partial w_k}-\alpha^{(1)}\alpha^{(n)}$$

This completes the proof.

It is easy to derive a recursive formula to compute the $\alpha^{(n+1)}$ and thereby obtain the formula for the higher derivatives

$$\frac{\partial^{n+1}\mathbf{C}(u)}{\partial w_k^{n+1}}=\alpha^{(n+1)}\big(\mathbf{P}_k-\mathbf{C}(u)\big)\qquad n=1,2,\ldots,\infty \tag{11.19}$$

with $$\alpha^{(n+1)}=(-1)(n+1)\alpha^{(n)}\alpha^{(1)} \tag{11.20}$$

Equations (11.18)–(11.20) prove that an arbitrary curve point, $\mathbf{C}(u)$, moves along a straight line toward $\mathbf{P}_k$ as w_k increases in value.

We now take a more geometric approach. Starting with the curve of Eq. (11.17), let w_k vary, $0\le w_k<\infty$. Denote the family of resulting curves by $\mathbf{C}(u;w_k)$ (see Figure 11.10). Assume $\bar{u}\in[u_k,u_{k+p+1})$, and define the points

$$\begin{aligned}\mathbf{R}&=\mathbf{C}(\bar{u};w_k=0)\\\mathbf{M}&=\mathbf{C}(\bar{u};w_k=1)\\\mathbf{P}&=\mathbf{C}(\bar{u};w_k\neq 0,1)\end{aligned} \tag{11.21}$$

Notice that $\mathbf{R}$ and $\mathbf{M}$ are points on specific curves, but $\mathbf{P}$ can lie anywhere on the line between $\mathbf{R}$ and $\mathbf{P}_k$, for $w_k\ge 0$. Now $\mathbf{M}$ and $\mathbf{P}$ can be expressed as

$$\begin{aligned}\mathbf{M}&=(1-s)\mathbf{R}+s\mathbf{P}_k\\\mathbf{P}&=(1-t)\mathbf{R}+t\mathbf{P}_k\end{aligned} \tag{11.22}$$

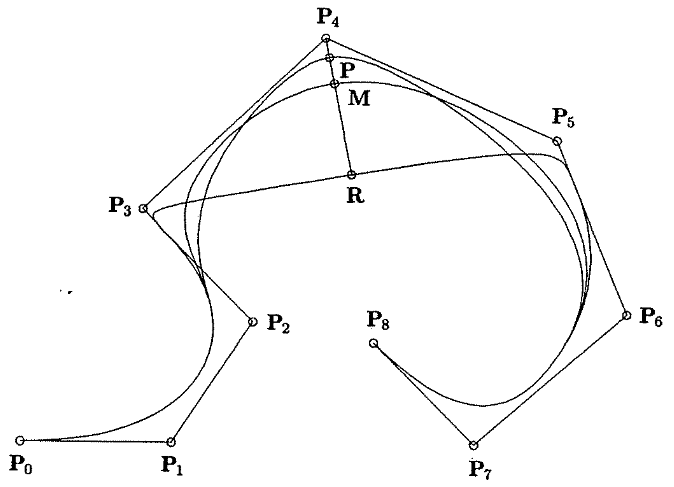


Figure 11.10. Geometric meaning of the weight.

where
$$s = \frac{N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}$$

$$t = \frac{N_{k,p}(\bar{u})\,w_k}{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i} = R_{k,p}(\bar{u}) \tag{11.23}$$

To derive these equations note that

$$\mathbf{R} = \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i\,\mathbf{P}_i}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i} \qquad w_k = 0$$

and thus we have

$$\mathbf{P} = \sum_{i=0}^{n} R_{i,p}(\bar{u})\,\mathbf{P}_i = \sum_{i\neq k} R_{i,p}(\bar{u})\,\mathbf{P}_i + R_{k,p}(\bar{u})\,\mathbf{P}_k$$
$$= \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i\,\mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i} + R_{k,p}(\bar{u})\,\mathbf{P}_k$$

$$= \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i}{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i}\,\mathbf{R} + R_{k,p}(\bar{u})\,\mathbf{P}_k$$

$$= \left(\frac{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i}{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i} - \frac{N_{k,p}(\bar{u})\,w_k}{\sum_{i=0}^{n} N_{i,p}(\bar{u})\,w_i}\right)\mathbf{R} + R_{k,p}(\bar{u})\,\mathbf{P}_k$$

$$= \left(1 - R_{k,p}(\bar{u})\right)\mathbf{R} + R_{k,p}(\bar{u})\,\mathbf{P}_k = (1-t)\,\mathbf{R} + t\,\mathbf{P}_k$$

where $t = R_{k,p}(\bar{u})$. Finally, setting $w_k = 1$ yields

$$\mathbf{M} = \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i\,\mathbf{P}_i}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})} + \frac{N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}\,\mathbf{P}_k$$

$$= \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}\,\mathbf{R} + \frac{N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}\,\mathbf{P}_k$$

$$= \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u}) - N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}\,\mathbf{R} + \frac{N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}\,\mathbf{P}_k$$

$$= (1-s)\,\mathbf{R} + s\,\mathbf{P}_k$$

where
$$s = \frac{N_{k,p}(\bar{u})}{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i + N_{k,p}(\bar{u})}$$

From Eqs. (11.22) and (11.23) we have

$$w_k = \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i}{N_{k,p}(\bar{u})} \Bigg/ \frac{\sum_{i\neq k} N_{i,p}(\bar{u})\,w_i}{N_{k,p}(\bar{u})\,w_k}$$

$$= \frac{1-s}{s} \Big/ \frac{1-t}{t} = \frac{\mathbf{P}_k\mathbf{M}}{\mathbf{R}\mathbf{M}} \Big/ \frac{\mathbf{P}_k\mathbf{P}}{\mathbf{R}\mathbf{P}} \tag{11.24}$$

Equation (11.24) says that w_k is the *cross ratio* of the four points, $\mathbf{P}_k$, $\mathbf{R}$, $\mathbf{M}$, and $\mathbf{P}$, denoted by $(\mathbf{P}_k, \mathbf{R}; \mathbf{M}, \mathbf{P})$ (see [Coxe74; Boeh94]). The cross ratio is a fundamental quantity of projective geometry. With respect to shape modifications of NURBS curves, we mention the following consequences of Eqs. (11.21)–(11.24):

- as w_k increases (decreases), t increases (decreases), thus the curve is pulled (pushed) toward (away from) $\mathbf{P}_k$;
- $\mathbf{P}$ moves along a straight line segment passing through $\mathbf{P}_k$; thus, a modification of w_k changes the curve in a predictable manner;
- as $\mathbf{P}$ approaches $\mathbf{P}_k$, t approaches 1 and w_k tends to infinity.

Now suppose the point $\mathbf{P} = \mathbf{C}(\bar{u})$ is to be pulled (pushed) toward (away from) $\mathbf{P}_k$ to a new location, $\hat{\mathbf{P}}$, by modifying w_k. Set $d = \mathbf{P}\hat{\mathbf{P}} = |\mathbf{P} - \hat{\mathbf{P}}|$. The required new weight, $\hat{w}_k$, is derived as

$$\hat{w}_k = \frac{\mathbf{P}_k\mathbf{M}\cdot(\mathbf{R}\mathbf{P} \pm d)}{\mathbf{R}\mathbf{M}\cdot(\mathbf{P}_k\mathbf{P} \mp d)}$$

$$= \frac{\mathbf{P}_k\mathbf{M}\cdot\mathbf{R}\mathbf{P}}{\mathbf{R}\mathbf{M}\cdot\mathbf{P}_k\mathbf{P} \mp d\cdot\mathbf{R}\mathbf{M}} \pm \frac{d\cdot\mathbf{P}_k\mathbf{M}}{\mathbf{R}\mathbf{M}\cdot(\mathbf{P}_k\mathbf{P} \mp d)}$$

Using

$$w_k = \frac{\mathbf{P}_k\mathbf{M}\cdot\mathbf{R}\mathbf{P}}{\mathbf{R}\mathbf{M}\cdot\mathbf{P}_k\mathbf{P}}$$

we have

$$\hat{w}_k = \frac{w_k\cdot\mathbf{P}_k\mathbf{P}}{\mathbf{P}_k\mathbf{P} \mp d} \pm \frac{d\cdot\mathbf{P}_k\mathbf{M}}{\mathbf{R}\mathbf{M}\cdot(\mathbf{P}_k\mathbf{P} \mp d)}$$

$$= w_k\left(\frac{\mathbf{P}_k\mathbf{P}}{\mathbf{P}_k\mathbf{P} \mp d} \pm \frac{d\cdot\mathbf{P}_k\mathbf{P}}{\mathbf{R}\mathbf{P}\cdot(\mathbf{P}_k\mathbf{P} \mp d)}\right)$$

$$= w_k\left(\frac{\mathbf{P}_k\mathbf{P}\cdot(\mathbf{R}\mathbf{P} \pm d)}{\mathbf{R}\mathbf{P}\cdot(\mathbf{P}_k\mathbf{P} \mp d)}\right) = w_k\left(\frac{(1-t)(\mathbf{R}\mathbf{P} \pm d)}{t\cdot(\mathbf{P}_k\mathbf{P} \mp d)}\right)$$

$$= w_k\left(\frac{(1-t)\cdot\mathbf{R}\mathbf{P} \mp td}{t\cdot\mathbf{P}_k\mathbf{P} \mp td} \pm \frac{d}{t\cdot(\mathbf{P}_k\mathbf{P} \mp d)}\right)$$

and from Eq. (11.24) we obtain

$$\hat{w}_k = w_k\left(1 \pm \frac{d}{R_{k,p}(\bar{u})(\mathbf{P}_k\mathbf{P} \mp d)}\right) \qquad (11.25)$$

Where $\pm$ or $\mp$ appears, the top sign is to be taken in case of a pull operation, and the bottom sign in case of a push operation [Pieg89c].

To implement a user interface for modification of one curve weight, a designer picks a point $\mathbf{P}$ on the curve and a point on the control polygon. The system

computes the parameter $\bar{u}$ such that $\mathbf{P} = \mathbf{C}(\bar{u})$. If the picked point on the control polygon is not already a control point, the system makes it a control point by using inverse knot insertion. Denote the control point by $\mathbf{P}_k$. The system then draws a line through $\mathbf{P}$ and $\mathbf{P}_k$. The designer picks the point $\hat{\mathbf{P}}$ on this line, allowing the system to then compute d, and finally $\hat{w}_k$ from Eq. (11.25). The reader should note that

- two user picks determine $\bar{u}$ and k. It is possible (but not likely) that $\bar{u} \notin [u_k, u_{k+p+1})$; the system should catch this and report it to the user as an invalid pick;
- in Eq. (11.25), the values $R_{k,p}(\bar{u})$ and $\mathbf{P}_k\mathbf{P}$ change with each modification of w_k, even when k and $\bar{u}$ remain constant; thus, if interactive incremental or continuous modification of the weight is desired, it is best to save the original $R_{k,p}(\bar{u})$, w_k, and $\mathbf{P}_k\mathbf{P}$ values and use them to compute new d and $\hat{w}_k$ values at each step;
- if we fix the weight change, Δw_k, and consider the distance $d(u) = |\mathbf{C}(u; w_k) - \mathbf{C}(u; \hat{w}_k)|$ as a function of u, it follows from Eq. (11.25) that $d(u)$ attains its maximum at the maximum of $R_{k,p}(u)$ (Figure 11.11);
- from a practical point of view, weight modification can/should only be used to effect small changes in a curve. For pull operations $\mathbf{P}_k$ is clearly a built-in bound on possible length of pull. This bound can be quite restrictive, particularly in cases where $\mathbf{P}_k$ was added through inverse knot insertion. Moreover, even moderate push/pulls often require substantial modifications in a weight; and widely varying weights can induce very poor

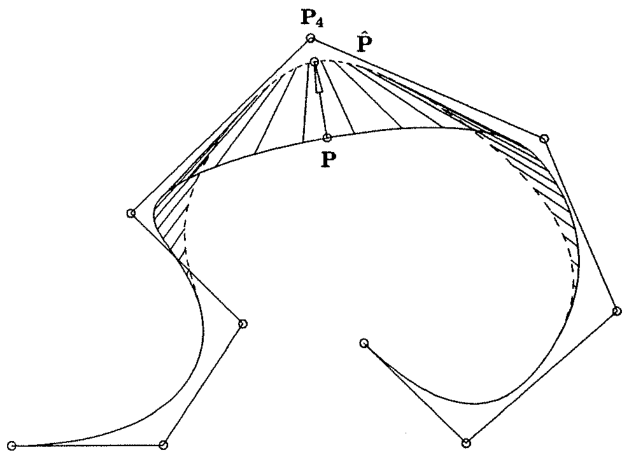


Figure 11.11. Pulling the curve at $t_4 = u = 0.52$ a distance $d = 0.075$ toward $\mathbf{P}_4$ by changing the weight $w_4 = 0.2$; the new weight computed is $\hat{w}_4 = 2.45$. Notice the perspective translation of the curve segment produced by the change of weight.

parameterizations, which often cause unexpected problems downstream. Figures 11.11 and 11.12 show such examples. For this reason, it may be advisable in a serious software implementation to restrict the range of allowable weight values;

- as was the case for control point repositioning, knot insertion can be used here to restrict the domain of curve change to lie between two user-specified curve points.

Figure 11.13 shows an example of incremental pulls. Note the exponential change of weights.

11.3.2 MODIFICATION OF TWO NEIGHBORING CURVE WEIGHTS

We consider now the simultaneous modification of two neighboring curve weights, w_k and w_{k+1}. Such a modification pulls (pushes) the curve toward (away from) the leg $\mathbf{P}_k\mathbf{P}_{k+1}$ of the control polygon. Fixing $\bar{u} \in [u_{k+1}, u_{k+p+1})$, let $\mathbf{P} = \mathbf{C}(\bar{u}; w_k, w_{k+1})$ and define

$$\mathbf{R} = \mathbf{C}(\bar{u}; w_k = w_{k+1} = 0) \tag{11.26}$$

(see Figure 11.14). Then $\mathbf{P}$ can be expressed as

$$\mathbf{P} = (1 - a_k - a_{k+1})\mathbf{R} + a_k\mathbf{P}_k + a_{k+1}\mathbf{P}_{k+1} \tag{11.27}$$

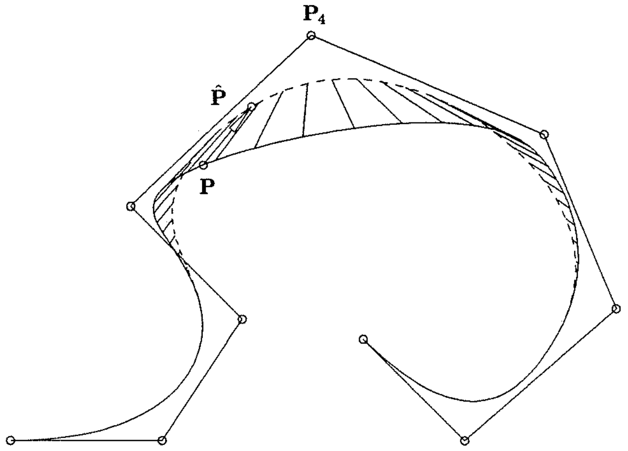


Figure 11.12. Pulling the curve at $u = 0.46$ a distance $d = 0.075$ toward $\mathbf{P}_4$ by changing the weight $w_4 = 0.2$; the new weight computed is $\hat{w}_4 = 0.97$.

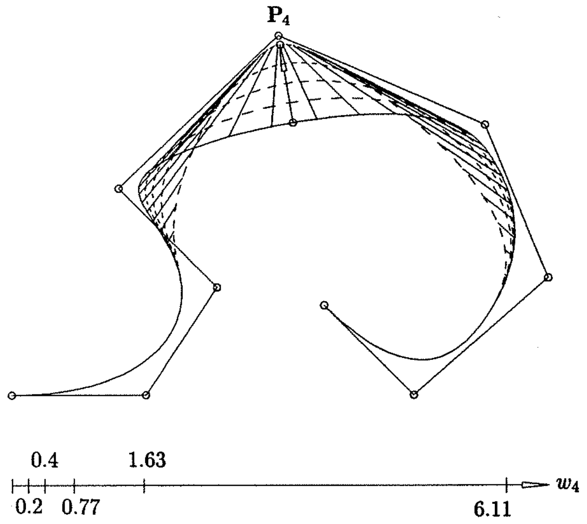


Figure 11.13. Pulling the curve at $t_4 = u = 0.52$ with equal increments $\Delta = 0.022$ toward $\mathbf{P}_4$ by changing the weight $w_4 = 0.2$; the new weights computed are $\hat{w}_4 = \{0.4, 0.77, 1.63, 6.11\}$.

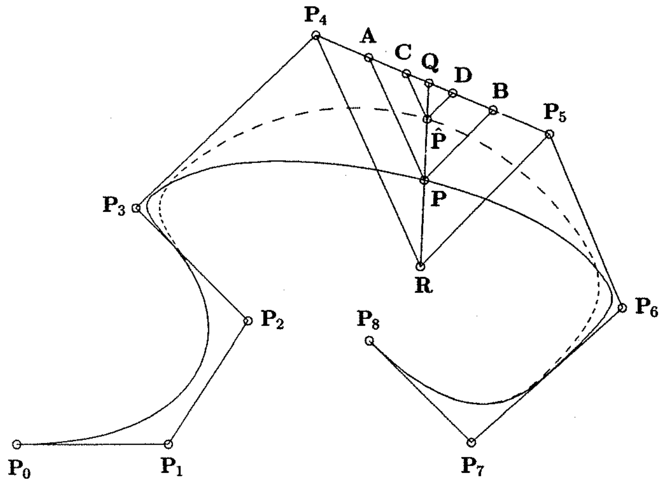


Figure 11.14. Simultaneous pull towards $\mathbf{P}_4$ and $\mathbf{P}_5$ by changing w_4 and w_5 at the same time.

$$\text{with} \qquad a_k = \frac{N_{k,p}(\bar{u})w_k}{\sum_{i=0}^{n} N_{i,p}(\bar{u})w_i} \qquad a_{k+1} = \frac{N_{k+1,p}(\bar{u})w_{k+1}}{\sum_{i=0}^{n} N_{i,p}(\bar{u})w_i} \tag{11.28}$$

Equation (11.27) says that $\mathbf{P}$ lies on the plane of the barycentric coordinate system $(\mathbf{R}\mathbf{P}_k\mathbf{P}_{k+1})$ (readers not familiar with barycentric coordinates should consult Boehm and Prautzsch [Boeh94] or Farin [Fari93]). Assume now that $\mathbf{P}$ moves to a new position, $\hat{\mathbf{P}}$. We must determine the new weights, $\hat{w}_k$ and $\hat{w}_{k+1}$, which cause the desired movement. By Eq. (11.27) $\hat{\mathbf{P}}$ can be expressed as

$$\hat{\mathbf{P}} = (1 - \hat{a}_k - \hat{a}_{k+1})\,\mathbf{R} + \hat{a}_k\,\mathbf{P}_k + \hat{a}_{k+1}\,\mathbf{P}_{k+1} \tag{11.29}$$

$$\text{with} \qquad \hat{a}_k = \frac{N_{k,p}(\bar{u})\hat{w}_k}{d} \qquad \hat{a}_{k+1} = \frac{N_{k+1,p}(\bar{u})\hat{w}_{k+1}}{d}$$

$$\text{where} \qquad d = \sum_{i \neq k,k+1} N_{i,p}(\bar{u})w_i + N_{k,p}(\bar{u})\hat{w}_k + N_{k+1,p}(\bar{u})\hat{w}_{k+1}$$

If $\hat{w}_k$ and $\hat{w}_{k+1}$ are expressed in the form

$$\hat{w}_k = \beta_k w_k \qquad \hat{w}_{k+1} = \beta_{k+1} w_{k+1} \tag{11.30}$$

then after some manipulation one gets

$$\beta_k = \frac{1 - a_k - a_{k+1}}{a_k} \Bigg/ \frac{1 - \hat{a}_k - \hat{a}_{k+1}}{\hat{a}_k}$$

$$\beta_{k+1} = \frac{1 - a_k - a_{k+1}}{a_{k+1}} \Bigg/ \frac{1 - \hat{a}_k - \hat{a}_{k+1}}{\hat{a}_{k+1}} \tag{11.31}$$

Equations (11.27)–(11.31) result in a simple geometric method to solve for the unknowns, β_k and β_{k+1}, and thus to obtain $\hat{w}_k$ and $\hat{w}_{k+1}$ from w_k and w_{k+1}. Since a_k, a_{k+1}, $\hat{a}_k$, and $\hat{a}_{k+1}$ are barycentric coordinates with respect to $\mathbf{R}$, $\mathbf{P}_k$, and $\mathbf{P}_{k+1}$, they can be expressed by (see Figure 11.14)

$$a_k = \frac{|\mathbf{B} - \mathbf{P}_{k+1}|}{|\mathbf{P}_{k+1} - \mathbf{P}_k|} \qquad a_{k+1} = \frac{|\mathbf{A} - \mathbf{P}_k|}{|\mathbf{P}_{k+1} - \mathbf{P}_k|}$$

$$\hat{a}_k = \frac{|\mathbf{D} - \mathbf{P}_{k+1}|}{|\mathbf{P}_{k+1} - \mathbf{P}_k|} \qquad \hat{a}_{k+1} = \frac{|\mathbf{C} - \mathbf{P}_k|}{|\mathbf{P}_{k+1} - \mathbf{P}_k|} \tag{11.32}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are points along $\mathbf{P}_k\mathbf{P}_{k+1}$ such that $\mathbf{PA}$ and $\hat{\mathbf{P}}\mathbf{C}$ are parallel to $\mathbf{R}\mathbf{P}_k$, and $\mathbf{PB}$ and $\hat{\mathbf{P}}\mathbf{D}$ are parallel to $\mathbf{R}\mathbf{P}_{k+1}$. To have nonvanishing a_k, $\bar{u}$ must be within $[u_k, u_{k+p+1})$; similarly, for nonvanishing a_{k+1} $\bar{u}$ must be within $[u_{k+1}, u_{k+p+2})$. If $\mathbf{P}$ is chosen so that $\bar{u}$ is within the intersection of these intervals, neither a_k nor a_{k+1} vanishes, and consequently $\hat{\mathbf{P}}$ can be placed

anywhere within the triangle $\mathbf{R}\mathbf{P}_k\mathbf{P}_{k+1}$. If, however, $\bar{u} \in [u_k, u_{k+1})$, then $a_{k+1} = 0$ and Eq. (11.27) degenerates to

$$\mathbf{P} = (1 - a_k)\,\mathbf{R} + a_k\,\mathbf{P}_k \tag{11.33}$$

i.e., the simultaneous modification degenerates to a single weight modification.

Figures 11.15 and 11.16 show simultaneous push/pulls. Notice that the curve change is pleasing and symmetric in Figure 11.15 but not in Figure 11.16. This is due to the fact that in Figure 11.15 the movement is along the line defined by $\mathbf{R}$ and the initial $\mathbf{P} = \mathbf{C}(\bar{u}; w_k)$. In this case we have

$$\frac{\hat{a}_k}{a_k} = \frac{\hat{a}_{k+1}}{a_{k+1}}$$

which implies that $\beta_k = \beta_{k+1}$ and

$$\frac{w_k}{w_{k+1}} = \frac{\hat{w}_k}{\hat{w}_{k+1}}$$

that is, the ratio of the weights is preserved.

A user interface for simultaneous weight modifications might be as follows. The designer picks a control polygon leg, $\mathbf{P}_k\mathbf{P}_{k+1}$. The system computes the corresponding nodes, t_k and t_{k+1}, and sets $\bar{u} = 1/2(t_k + t_{k+1})$. The system then computes $\mathbf{R}$, $\mathbf{P} = \mathbf{C}(\bar{u})$, and $\mathbf{Q}$ (the intersection of $\mathbf{RP}$ with $\mathbf{P}_k\mathbf{P}_{k+1}$; see

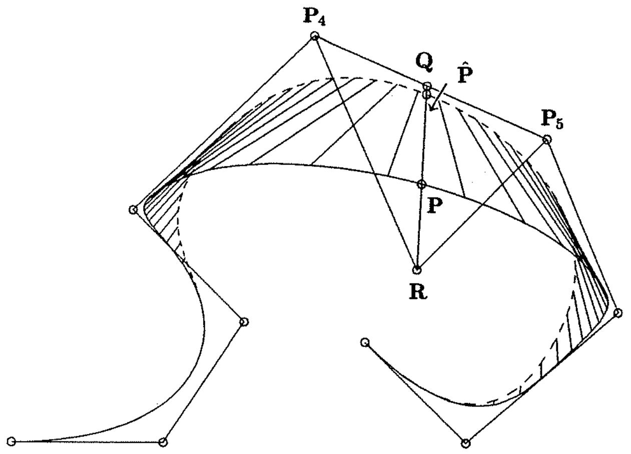


Figure 11.15. Simultaneous pull at $u = 1/2(t_4 + t_5) = 0.608$ a distance $d = 0.088$ in the direction of $\mathbf{RQ}$ by changing $w_4 = w_5 = 0.05$; the new weights are $\hat{w}_4 = \hat{w}_5 = 1.23$. The curve change is neither perspective nor parallel.

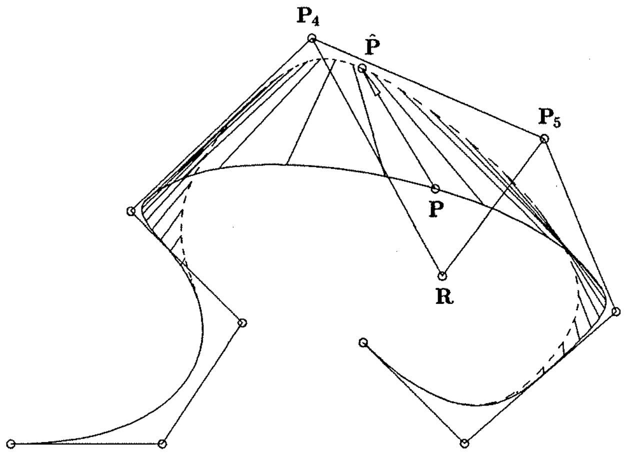


Figure 11.16. Simultaneous pull at $u = 0.612$ a distance $d = 0.088$ in a direction different from $\mathbf{RQ}$ by changing $w_4 = w_5 = 0.05$; the new weights are $\hat{w}_4 = 1.89$ and $\hat{w}_5 = 0.48$.

Figure 11.14). These components, including the line segment $\mathbf{RQ}$, are drawn on the screen. The designer can now interactively reposition $\mathbf{P}$ along the line $\mathbf{RQ}$ while the system computes new weights and redraws the affected curve segment. Of course, the designer should be able to override the default, $\bar{u}$, and the default line segment, $\mathbf{RQ}$.

Another scenario is to allow the designer to pick the curve point, $\mathbf{P}$ (and thereby $\bar{u}$), and the system subsequently chooses the polygon leg $\mathbf{P}_k\mathbf{P}_{k+1}$ by finding two nodes, t_k and t_{k+1}, such that $t_k < \bar{u} < t_{k+1}$. However, this is problematic if $\bar{u}$ is equal to or close to a node, as the push/pull effect is asymmetric if a polygon leg is arbitrarily chosen. In this case a knot, $\hat{u}$, can be inserted, which puts $\bar{u}$ halfway between two nodes. Specifically

$$\begin{aligned}\bar{u} &= \frac{1}{2}\left(\hat{t}_k + \hat{t}_{k+1}\right)\\ &= \frac{1}{2p}\left(u_{k+1} + \cdots + u_{k+p-1} + \hat{u} + u_{k+2} + \cdots + u_{k+p} + \hat{u}\right)\\ &= \frac{1}{2p}\left(u_{k+1} + u_{k+p} + 2\hat{u} + 2\sum_{j=k+2}^{k+p-1} u_j\right)\end{aligned}$$

which implies that

$$\hat{u} = p\bar{u} - \frac{1}{2}\left(u_{k+1} + u_{k+p}\right) - \sum_{j=2}^{p-1} u_{k+j}$$

However, we caution the reader that $\hat{u}$ may already be a knot; furthermore, the resulting new polygon leg, $\mathbf{P}_k\mathbf{P}_{k+1}$, may pass very close to $\mathbf{P} = \mathbf{C}(\bar{u})$, in which case the available pull length is very limited.

Finally, note that we assumed previously that the two control points whose weights we modified were neighbors. This was done because it yields pleasing results and is generally what is desired. However, the derivations of Eqs. (11.27)–(11.32) do not make use of this assumption. In fact, if $\mathbf{P}_r$ and $\mathbf{P}_s$ are chosen, then the only requirement is that $\bar{u} \in [u_r, u_{r+p+1})$ and $\bar{u} \in [u_s, u_{s+p+1})$, so that the barycentric coordinates, a_r and a_s, do not vanish. See [Pieg89c] for more details of this case.

11.3.3 MODIFICATION OF ONE SURFACE WEIGHT

Modification of a single surface weight, $w_{k,\ell}$, is similar to the curve case. Assume $\bar{u} \in [u_k, u_{k+p+1})$ and $\bar{v} \in [v_\ell, v_{\ell+q+1})$, and define

$$\begin{aligned}\mathbf{R} &= \mathbf{S}(\bar{u},\bar{v}; w_{k,\ell} = 0)\\ \mathbf{M} &= \mathbf{S}(\bar{u},\bar{v}; w_{k,\ell} = 1)\\ \mathbf{P} &= \mathbf{S}(\bar{u},\bar{v}; w_{k,\ell} \neq 0, 1)\end{aligned} \tag{11.35}$$

Then $\mathbf{M}$ and $\mathbf{P}$ can be expressed as

$$\begin{aligned}\mathbf{M} &= (1-s)\,\mathbf{R} + s\mathbf{P}_{k,\ell}\\ \mathbf{P} &= (1-t)\,\mathbf{R} + t\mathbf{P}_{k,\ell}\end{aligned} \tag{11.36}$$

where

$$s = \frac{N_{k,p}(\bar{u})N_{\ell,q}(\bar{v})}{\sum\sum_{i,j\neq k,\ell} N_{i,p}(\bar{u})N_{j,q}(\bar{v})\,w_{i,j} + N_{k,p}(\bar{u})N_{\ell,q}(\bar{v})}$$

$$t = \frac{N_{k,p}(\bar{u})N_{\ell,q}(\bar{v})\,w_{k,\ell}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\bar{u})N_{j,q}(\bar{v})\,w_{i,j}} = R_{k,\ell}(\bar{u},\bar{v}) \tag{11.37}$$

($i,j \neq k,\ell$ means exclude the (k,ℓ)th term.) Using Eqs. (11.36) and (11.37), one obtains the cross ratio

$$w_{k,\ell} = \frac{1-s}{s} \bigg/ \frac{1-t}{t} = \frac{\mathbf{P}_{k,\ell}\mathbf{M}}{\mathbf{RM}} \bigg/ \frac{\mathbf{P}_{k,\ell}\mathbf{P}}{\mathbf{RP}} \tag{11.38}$$

Equations (11.35)–(11.38) imply that

- as $w_{k,\ell}$ increases (decreases) t increases (decreases), and thus the surface is pulled (pushed) toward (away from) $\mathbf{P}_{k,\ell}$;

- $\mathbf{P}$ moves along a straight line segment passing through $\mathbf{P}_{k,\ell}$; thus, a modification of $w_{k,\ell}$ changes the surface in a predictable manner;
- as $\mathbf{P}$ approaches $\mathbf{P}_{k,\ell}$, t approaches 1 and $w_{k,\ell}$ tends to infinity.

Finally, if $\mathbf{P}$ is to move to the new location, $\hat{\mathbf{P}}$, and $d = |\mathbf{P} - \hat{\mathbf{P}}|$, then the required new weight value, $\hat{w}_{k,\ell}$, is

$$\hat{w}_{k,\ell} = w_{k,\ell}\left(1 \pm \frac{d}{R_{k,\ell}(\bar{u},\bar{v})(\mathbf{P}_{k,\ell}\mathbf{P} \mp d)}\right) \tag{11.39}$$

Where $\pm$ or $\mp$ appears, the top sign is to be taken in case of a pull operation, and the bottom sign in case of a push operation [Pieg89d].

A user interface for modification of a single surface weight can be implemented in a manner analogous to that presented for curves. The designer can pick either an existing control point or one or two points lying on edges of the control net; if two, then they must lie on a quadrilateral of the control net, one on a u-directional side and one on a v-directional side. If an existing control point is not picked, then inverse knot insertion is used to obtain $\mathbf{P}_{k,\ell}$. From there, the process proceeds as for curve modification. Statements analogous to the five remarks mentioned previously in regard to curve weight modification also apply to surface weight modification. Figure 11.17 illustrates a surface push.

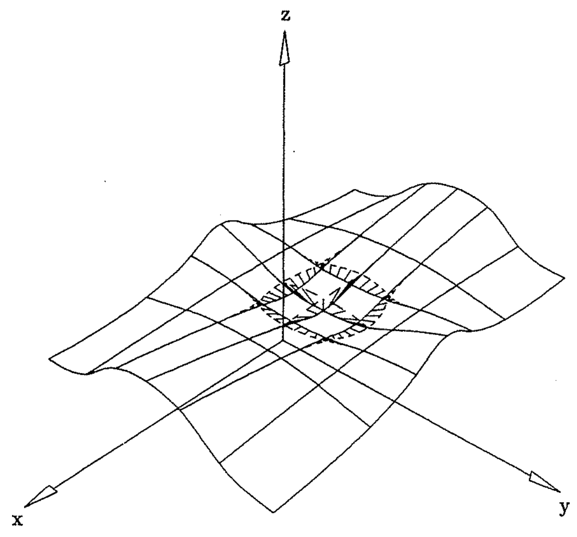


Figure 11.17. Surface push at $u = v = 0.55$ toward $\mathbf{P}_{4,3}$ a distance $d = 1$; notice the perspective change in surface shape.

11.4 Shape Operators

As defined here, shape operators are tools for rough sculpting of curves and surfaces. They operate as follows:

1. a local curve segment or surface region is specified, within which the shape modification is to occur; knot refinement is then used to introduce more control points locally;
2. control points whose influence is local to the specified segment or region are repositioned according to various parameters and/or functions;
3. knot removal to within a specified tolerance is applied to the local segment or region in order to reduce the number of defining control points.

We present three operators here: warp, flatten, and bend. Additional similar operators can be found in the literature, e.g., stretch, twist, and taper [Barr83; Cobb84]. We emphasize that our algorithms are based purely on geometry; they do not consider physical properties of materials or theories of elasticity and plasticity.

Before presenting the operators, we mention a few issues relating to the knot refinement and removal steps. The method and level of refinement affects the results of applying a shape operator. Hence, although the designer should be spared the details, he must be allowed to control the level of refinement. Assume that $[u_s, u_e]$ is a curve segment of interest. If u_s and u_e are not already knots, they are inserted. A good way to insert n additional knots into $[u_s, u_e]$ is to recursively insert a knot at the midpoint of the longest knot span in $[u_s, u_e]$. This tends to generate roughly evenly spaced knots, and hopefully (but not necessarily) the control points will also be roughly evenly spaced. Now if u_s and u_e are knots defining the limits of the final refined interval (note that e changes during refinement), then the following control points can be repositioned without modifying the curve outside of $[u_s, u_e]$

$$\mathbf{P}_s, \ldots, \mathbf{P}_{e-p-1} \tag{11.40}$$

where p is the degree of the curve. After the shaping operation, knot removal typically removes many of the inserted knots. If the curve is not to be modified outside $[u_s, u_e]$, then only the following knots are candidates for removal

$$\begin{array}{lll} u_i & i = s + \dfrac{p}{2}, \ldots, e - \dfrac{p}{2} - 1 & \text{if } p \text{ is even} \\ u_i & i = s + \dfrac{p+1}{2}, \ldots, e - \dfrac{p+1}{2} & \text{if } p \text{ is odd} \end{array} \tag{11.41}$$

11.4.1 WARPING

Warping is a tool which can be used to rather arbitrarily deform a local segment of a curve or region on a surface. First consider *curve warping*, which is implemented by means of control point repositioning based on the formula

$$\hat{\mathbf{P}}_i = \mathbf{P}_i + fd\mathbf{W} \tag{11.42}$$

Generally, f is a function, d a constant, and $\mathbf{W}$ can be either a function or a constant. f controls the warp shape, d is an upper bound on control point movement (and hence on warp distance), and $\mathbf{W}$ gives the direction of warp. The designer generally determines the extent of the warp by selecting curve points $\mathbf{C}(u_s)$ and $\mathbf{C}(u_e)$. The curve warp is restricted to this segment. A useful warping function is given by

$$f(t) = \frac{R_{3,3}(t)}{R_{\max}} \tag{11.43}$$

where $R_{\max} = R_{3,3}(1/2)$ and $R_{3,3}(t)$ is the rational cubic B-spline basis function

$$R_{3,3}(t) = \frac{N_{3,3}(t)w_3}{\sum_{i=0}^{6} N_{i,3}(t)w_i} \tag{11.44}$$

defined by the knot vector

$$T = \left\{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\right\} \tag{11.45}$$

and weights

$$w_i = \{1,1,1,w_3,1,1,1\}$$

Figure 11.18 shows $f(t)$ for several different values of w_3. Clearly, functions of this form are appropriate for pulling out a bump or pushing in an indentation in a curve. Now suppose w_3 is fixed and the designer specifies a maximum

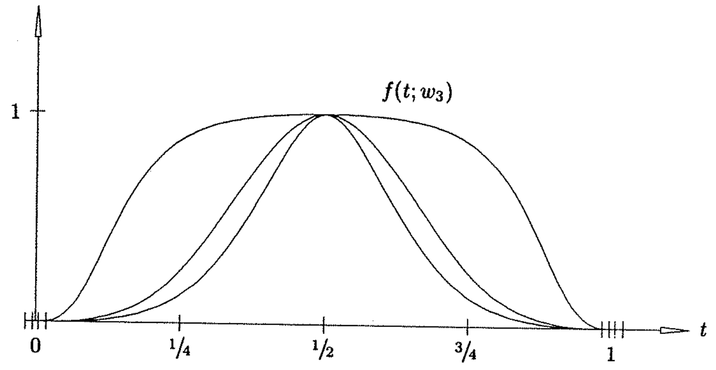


Figure 11.18. Warping functions using the rational basis function $R_{3,3}(t; w_3)$ with $w_3 = \{1/5, 1, 30\}$.

warp distance, d. The warp direction, $\mathbf{W}$, remains to be specified. Reasonable choices are:

- a variable direction given by

$$\mathbf{W}(t) = \pm\mathbf{N}(u(t)) \tag{11.46}$$

  where $\mathbf{N}(u)$ is the unit normal vector of $\mathbf{C}(u)$, and $u(t) = (u_e - u_s)t + u_s$;
- the constant direction given by

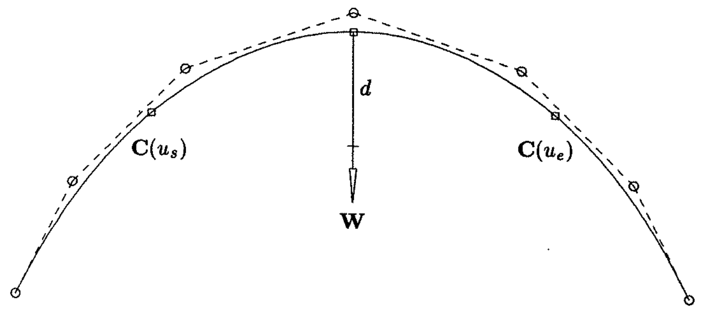
$$\mathbf{W} = \pm\mathbf{N}\left(\frac{1}{2}(u_s + u_e)\right) \tag{11.47}$$

The interval $[u_s, u_e]$ is now refined, yielding control points $\mathbf{P}_i$, $i = s, \ldots, e-p-1$ (Eq. [11.40]), which can be repositioned according to Eq. (11.42). In order to apply that equation, we must assign a parameter t_i to each $\mathbf{P}_i$, $i = s, \ldots, e-p-1$. Let δ denote the total accumulated chordal distance along the control polygon from $\mathbf{P}_{s-1}$ to $\mathbf{P}_{e-p}$. Denote by t_i the accumulated chordal distance from $\mathbf{P}_{s-1}$ to $\mathbf{P}_i$, normalized by δ. Notice that $0 \le t_i \le 1$ for $s \le i \le e-p-1$. Then $\mathbf{P}_i$ is repositioned by

$$\hat{\mathbf{P}}_i = \mathbf{P}_i + f(t_i)d\mathbf{W}(t_i) \qquad i = s, \ldots, e-p-1 \tag{11.48}$$

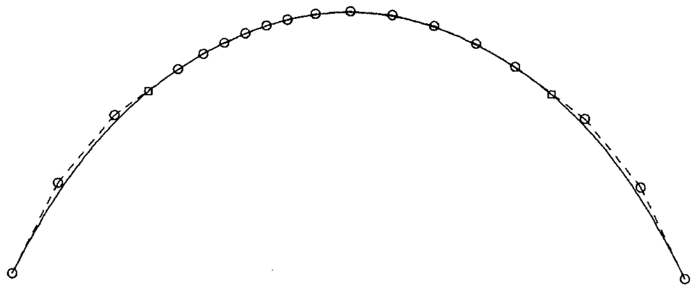
Finally, knot removal is applied to the interval as discussed previously.

Figures 11.19a–11.19d show the steps in curve warping. In Figure 11.19a the curve segment $[\mathbf{C}(u_s), \mathbf{C}(u_e)]$, the (constant) direction $\mathbf{W}$, and the distance d are selected. Figure 11.19b shows the knot refined curve. Local control points are repositioned, as shown in Figure 11.19c. Figure 11.19d depicts the final curve after knot removal is applied.

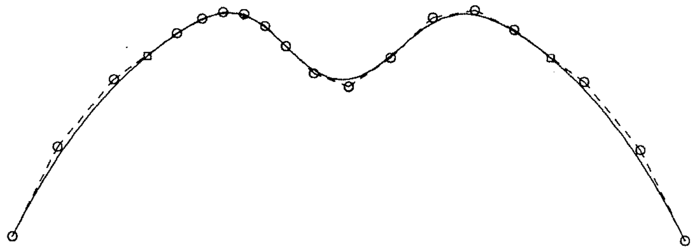


(a)

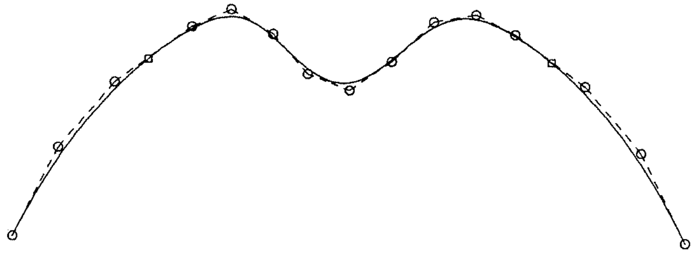
Figure 11.19. Curve warping. (a) The original curve with span $[\mathbf{C}(u_s), \mathbf{C}(u_e)]$ to be warped in the direction of $\mathbf{W}$ a distance d; (b) a curve span refined; (c) control points repositioned; (d) knots removed.



(b)



(c)



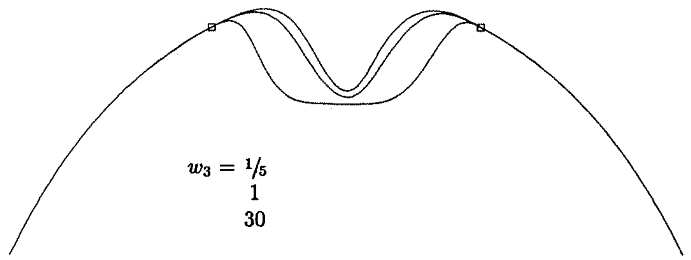
(d)

Figure 11.19. (*Continued.*)

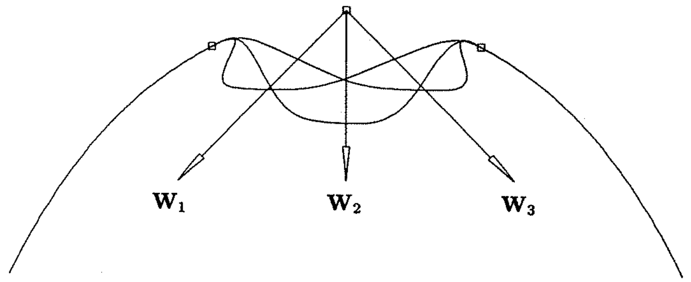
The effects of changing the defining parameters are demonstrated in Figures 11.20a–11.20c. In Figure 11.20a different weights are used to "square" down the warped region. Figure 11.20b shows the effect of changing the warp direction while keeping both the weight and the distance constant. Finally, in Figure 11.20c different distances are applied to control the magnitude of the warp.

We conclude curve warping by noting that:

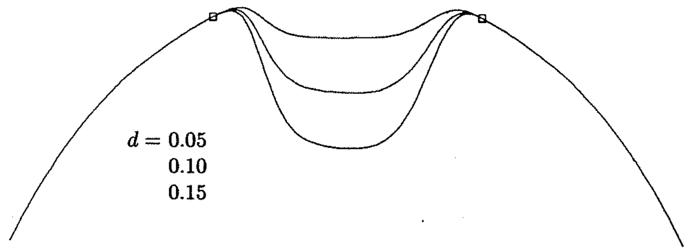
- clearly warps of almost any shape and magnitude are possible; one needs only to define the functions $f(t)$ and $\mathbf{W}(t)$ suitably. Two further examples are shown in Figures 11.21a and 11.21b;



(a)



(b)



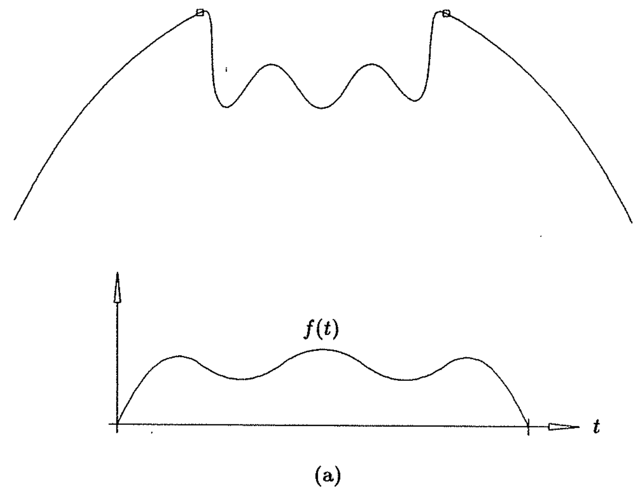
(c)

Figure 11.20. Curve warps (a) using different weights $w_3 = \{1/5, 1, 30\}$; (b) using different directions with $w_3 = 20$; (c) using different distances $d = \{0.05, 0.1, 0.15\}$.

- for interactive warping the designer may wish to select a curve point which is to be the center of the warp, and then to modify on a sliding scale the magnitude d, the weight w_3, and the extent of the warp segment. The segment is chosen to be as symmetric as possible about the warp center, measured in distance along the control polygon. Knot removal is done after completion of the interaction; knot refinement must be done either as a preprocessing step (either on the entire curve, or on a user-specified "maximum" segment), or the refinement must be updated (expanded) as the segment length is expanded;

- instead of computing the t_i as normalized distances, one can use the B-spline nodes computed by Eq. (11.4); in this case, the functions f and $\mathbf{W}$ must be parameterized on the interval $[u_s, u_e]$.

Surface warping is similar to curve warping. We distinguish two types, *region warping* and *surface curve warping.* In region warping we apply a warping function to a region of nonzero surface area; for surface curve warping we allow a univariate warping function to slide along a curve lying on the surface.



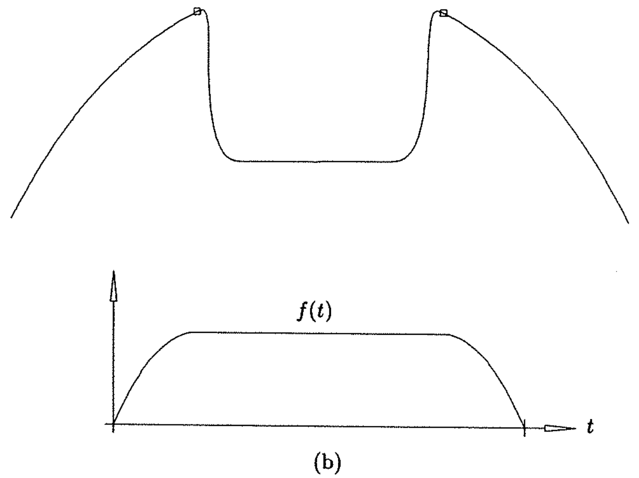


Figure 11.21. Curve warps with different shape functions.

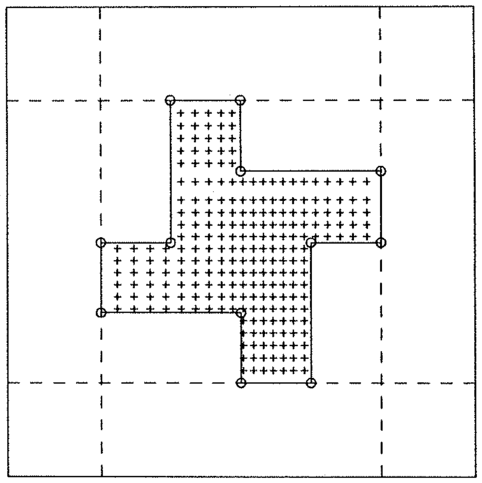
Let $\mathcal{R}$ be a closed region in the parameter space of a surface, $\mathbf{S}(u,v)$. Examples are shown in Figures 11.22a and 11.23a. Region warping is produced by repositioning control points, based on the formula

$$\hat{\mathbf{P}}_{i,j} = \mathbf{P}_{i,j} + f d\mathbf{W} \tag{11.49}$$

Analogous to curve warping, knot refinement is applied to the area $[u_\alpha, u_\beta] \times [v_\gamma, v_\delta]$ (shown by dashed lines in Figures 11.22a and 11.23a) before using Eq. (11.49). Knot removal is applied in the appropriate region (see Eq. [11.41]) as a postprocessing step. A parameter pair (s_i, t_j) must be associated with each control point $\mathbf{P}_{i,j}$, $i = \alpha, \ldots, \beta - p - 1$, $j = \gamma, \ldots, \delta - q - 1$. These parameters are used to determine which $\mathbf{P}_{i,j}$ are to be repositioned and to evaluate the functions f and $\mathbf{W}$ of Eq. (11.49). Analogous to curve warping, we use normalized accumulated chord length. In particular, we apply Algorithm A9.3 to the set of points $\mathbf{P}_{i,j}$, $i = \alpha - 1, \ldots, \beta - p$ and $j = \gamma - 1, \ldots, \delta - q$. A $\mathbf{P}_{i,j}$ is repositioned by Eq. (11.49) only if (u_i, v_j) is in the warp region $\mathcal{R}$, where

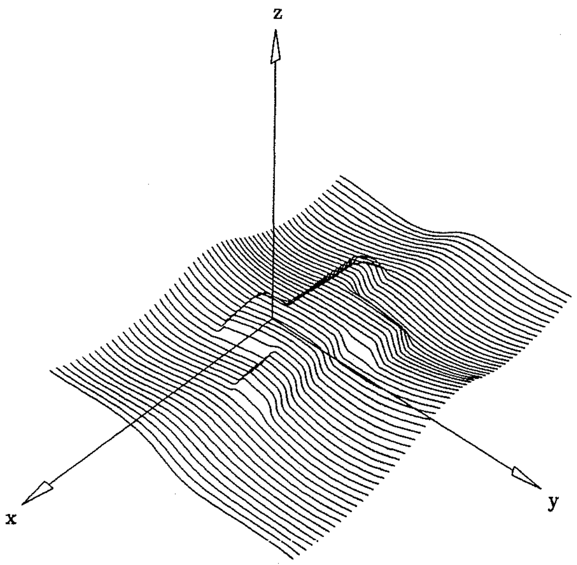
$$u_i = (u_\beta - u_\alpha)s_i + u_\alpha \qquad v_j = (v_\delta - v_\gamma)t_j + v_\gamma$$

f can be any scalar-valued function defined on $[0,1] \times [0,1]$. As was the case for curves, B-spline basis functions are appropriate for pulls and pushes. For example, the normalized, bivariate, bicubic, rational B-spline basis function is suitable for rectangular regions



(a)

Figure 11.22. Surface region warping. (a) A region with bounding box and control points local to the region; (b) a surface warp after control point repositioning.



(b)

Figure 11.22. (*Continued.*)

$$f(s,t) = \frac{R_{3,3;3,3}(s,t)}{R_{\max}} \tag{11.50}$$

defined on the knot vectors

$$S = \left\{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\right\}$$

and $$T = \left\{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\right\}$$

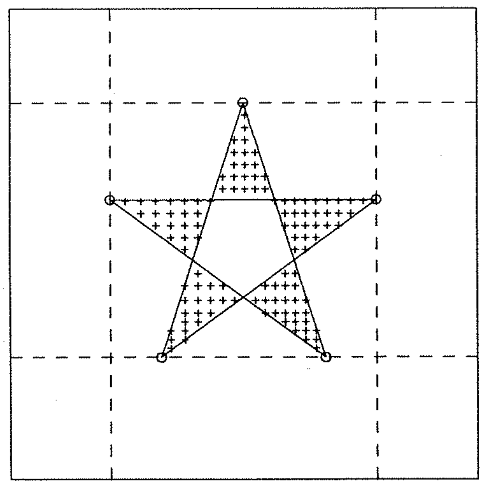
All weights are set to 1 except $w_{3,3}$, which is freely chosen. As another example, a rotated, univariate B-spline basis function can be used if the desired warp region exhibits circular symmetry.

Defining a warping function over arbitrary domains is difficult. The region warps of Figures 11.22 and 11.23 were created using the simple function

$$f(s,t) = \begin{cases} 1 & \text{if } \big(u(s), v(t)\big) \in \mathcal{R} \\ 0 & \text{otherwise} \end{cases} \tag{11.51}$$

where $$u(s) = (u_\beta - u_\alpha)s + u_\alpha \qquad v(t) = (v_\delta - v_\gamma)t + v_\gamma$$

This warping function acts like a die punch, i.e., each control point for which



(a)

Figure 11.23. Surface region warping. (a) A region with bounding box and control points local to the region; (b) a surface warp after control point repositioning.

$\big(u(s_i), v(t_j)\big) \in \mathcal{R}$ is displaced a distance, d, in the direction of $\mathbf{W}$. The vector $\mathbf{W}$ can be the surface normal at each point, or any other user-defined vector. For Figures 11.22 and 11.23 a ray casting, point-in-polygon routine was used to determine point containment in the polygonal regions. Notice in Figure 11.23 that the routine handles self-intersecting polygons.

We now describe surface curve warping. Suppose

$$\mathbf{C}(\lambda) = \big(u(\lambda), v(\lambda)\big) \tag{11.52}$$

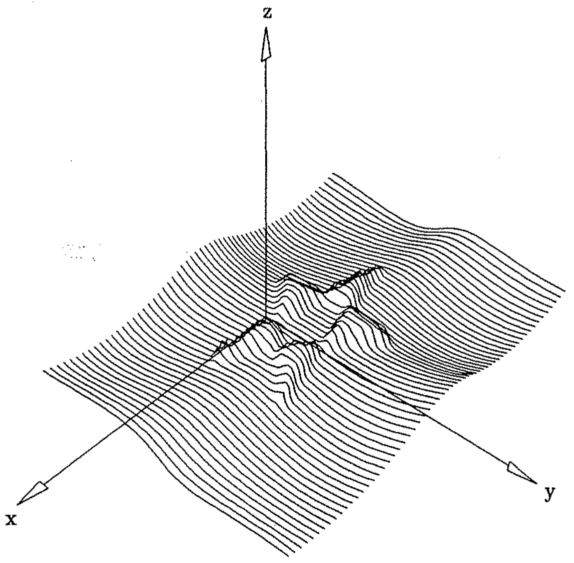
is a curve lying in the parameter space of $\mathbf{S}(u,v)$, and r is a distance measured to both sides of $\mathbf{C}(\lambda)$ along the line normal to $\mathbf{C}(\lambda)$ at each λ. Surface curve warping is applied by repositioning all those surface control points $\mathbf{P}_{i,j}$ whose parameters $(u(s_i), v(t_j))$ lie within distance r of $\mathbf{C}(\lambda)$ (the *field of gravity*). The relevant $\mathbf{P}_{i,j}$ are moved according to the formula

$$\hat{\mathbf{P}}_{i,j} = \mathbf{P}_{i,j} + f(d_{i,j})d\,\mathbf{W}(\lambda_{i,j}) \tag{11.53}$$

where $d_{i,j}$ is the distance of $\big(u(s_i), v(t_j)\big)$ to $\mathbf{C}(\lambda)$, and $\lambda_{i,j}$ corresponds to the projection of $(u(s_i), v(t_j))$ onto $\mathbf{C}(\lambda)$. A reasonable definition of the warp direction is

$$\mathbf{W}(\lambda) = \mathbf{N}\big(\mathbf{C}(\lambda)\big) \tag{11.54}$$

where $\mathbf{N}(u,v)$ is the surface normal vector.



(b)

Figure 11.23. (*Continued.*)

Again, the definition of f can be delicate. We used the function

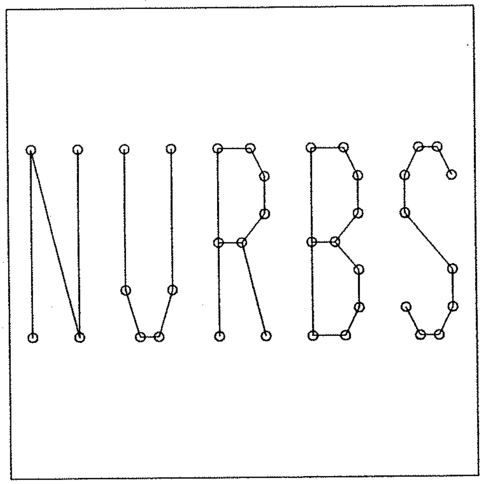
$$f(d_{i,j}) = \begin{cases} 1 & \text{if } d_{i,j} \le r \\ 0 & \text{otherwise} \end{cases} \tag{11.55}$$

The critical step in applying Eq. (11.53) is the computation of the $d_{i,j}$. This is simplified if $\mathbf{C}(\lambda)$ is a polyline. Figures 11.24 show an example of a *polyline warp*. Figure 11.24a depicts the acronym NURBS described as five polylines. Figure 11.24b shows the bounding box of the control points (marked with +s) that lie within the gravity fields of the five letters. The final polyline warped surface is illustrated in Figure 11.24c.

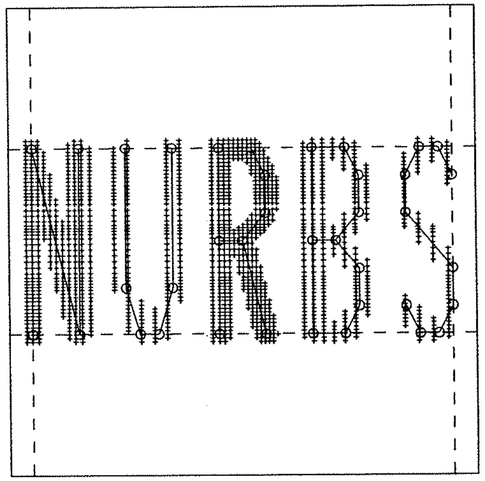
11.4.2 FLATTENING

The flatten operator allows a designer to easily introduce straight line segments into curves and planar regions into surfaces. No multiple knots are inserted, so the transition from the curved portions of the curve or surface into the flat portions possesses the smoothness inherent in the original geometry.

Consider curve flattening first, as shown in Figures 11.25 and 11.26; a curve, $\mathbf{C}(u)$, and a line, $\mathbf{L}$, are given. Curve flattening consists of three steps:



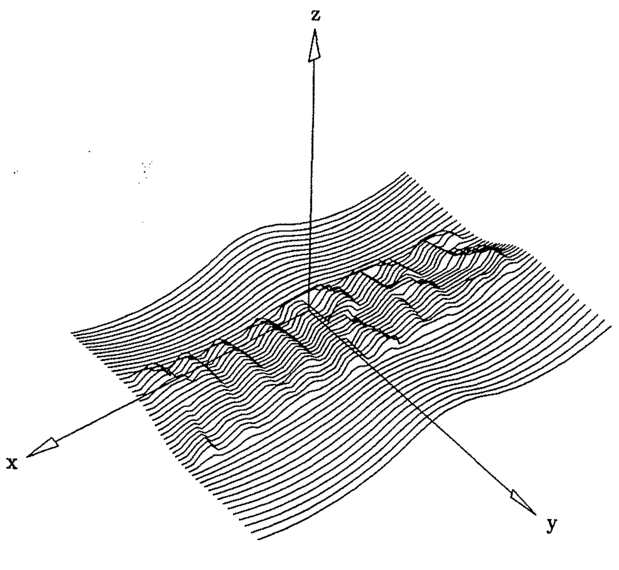
(a)



(b)

Figure 11.24. Surface polyline warping. (a) A polyline representing the acronym **NURBS**; (b) control points local to the gravity field of the polyline; (c) a surface polyline warp after control point repositioning.

1. refine the curve's knots to obtain more control points;
2. project some specified control points, $\mathbf{P}_i$, onto the line; denoting the projection of $\mathbf{P}_i$ by $\hat{\mathbf{P}}_i$, replace $\mathbf{P}_i$ by $\hat{\mathbf{P}}_i$;
3. remove unnecessary knots.



(c)

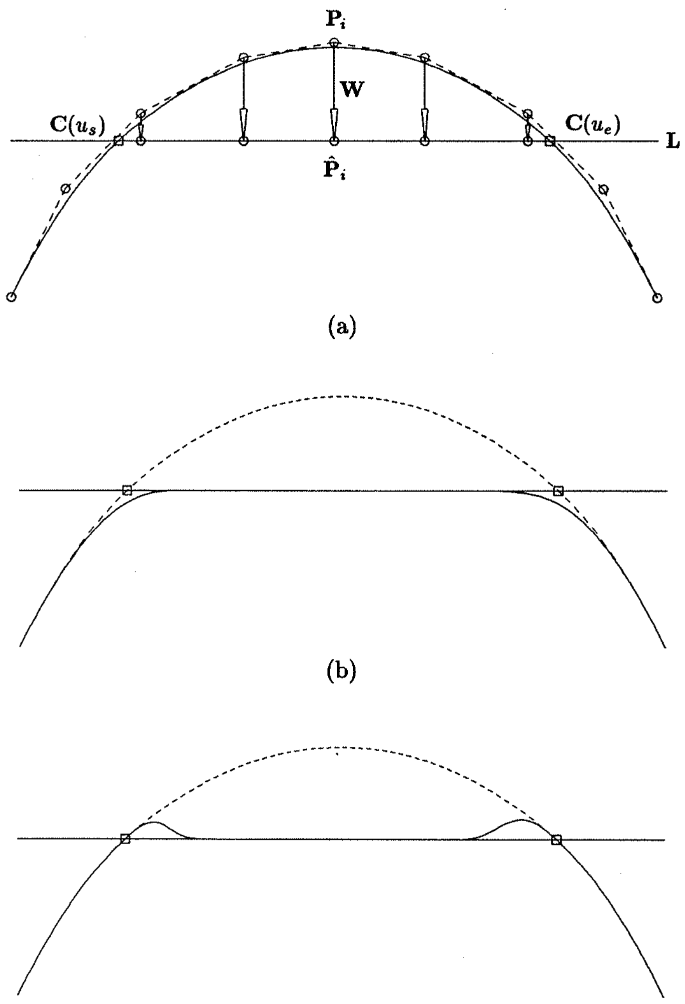
Figure 11.24. (*Continued.*)

In addition to **L** and $\mathbf{C}(u)$, the following items can be specified, either by the designer or defaulted by the system (Figure 11.25a):

- a vector, **W**, defining the line of projection, i.e., $\hat{\mathbf{P}}_i = \mathbf{P}_i + \alpha\mathbf{W}$ (α can be negative or positive); this vector can be defaulted to be perpendicular to **L**;
- parameters u_s and u_e: A control point, $\mathbf{P}_i$, is projected only if it is local to the interval $[u_s, u_e]$, i.e., $i = s, \ldots, e-p-1$;
- two points, $\mathbf{Q}_1$ and $\mathbf{Q}_2$, on the line **L**; a control point, $\mathbf{P}_i$, is replaced by its projection, $\hat{\mathbf{P}}_i$, only if $\hat{\mathbf{P}}_i$ lies within the segment $\mathbf{Q}_1\mathbf{Q}_2$.

Notice that the last two items are rules indicating which $\mathbf{P}_i$ to project and which $\hat{\mathbf{P}}_i$ to accept, respectively. For the situation shown in Figure 11.25a, setting $\mathbf{Q}_1 = \mathbf{C}(u_s)$ and $\mathbf{Q}_2 = \mathbf{C}(u_e)$ to be the points where **L** intersects $\mathbf{C}(u)$ is a reasonable system default. If the curve is xy planar and intersects **L**, as in Figure 11.25a, another intuitive rule is to project only those $\mathbf{P}_i$ which lie to one side of **L**, as shown in Figure 11.25b. If only those control points are projected which are local to $[u_s, u_e]$, unexpected results can occur if **L** intersects the curve (Figure 11.25c). If the curve does not intersect **L**, projecting all control points local to $[u_s, u_e]$ flattens the segment $[\mathbf{C}(u_s), \mathbf{C}(u_e)]$, leaving the rest of the curve unaffected. This is illustrated in Figures 11.26a and 11.26b.

Finally, we caution that a flat segment is obtained only if the projection results in at least $p+1$ consecutive points, $\hat{\mathbf{P}}_i$, where p is the degree of the curve. If



(c)

Figure 11.25. Curve flattening. (a) The original curve with span $[\mathbf{C}(u_s), \mathbf{C}(u_e)]$ to be flattened in the direction of **W** to the line **L** intersecting the curve; (b) a flattened curve obtained by projecting all control points lying on one side of **L**; (c) a flattened curve obtained by projecting control points local to the span $[u_s, u_e]$.

this fails to be the case, then more knot refinement is required. If u_s and u_e are specified, then refinement is required only in the interval $[u_i, u_j]$, where u_i is the largest knot satisfying $u_i < u_s$, and u_j is the smallest knot such that $u_e < u_j$.

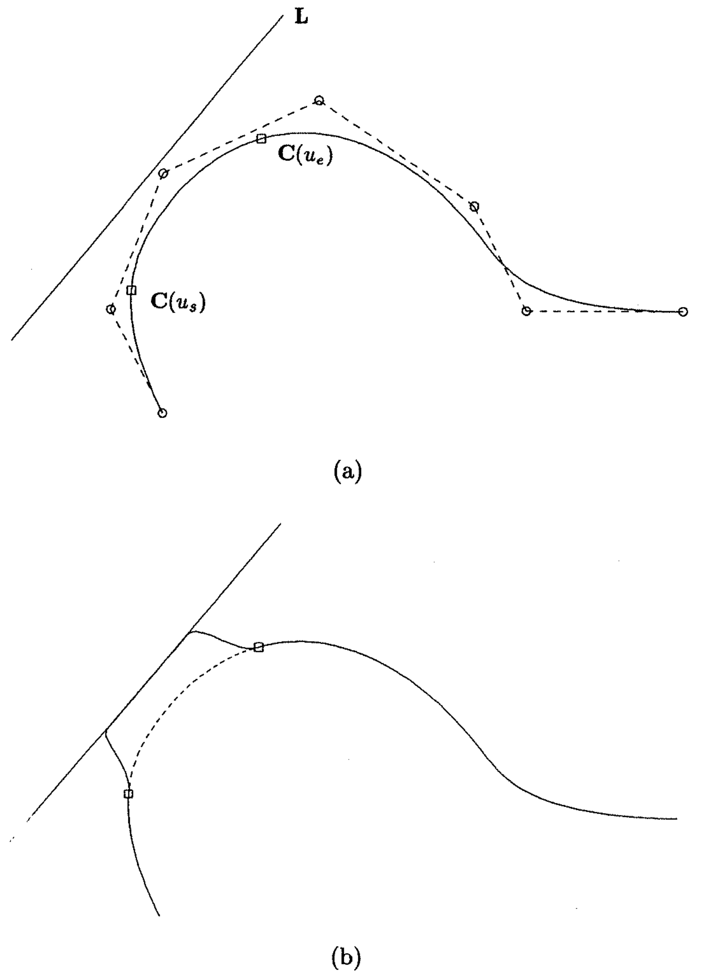


Figure 11.26. Curve flattening. (a) The original curve with span $[\mathbf{C}(u_s), \mathbf{C}(u_e)]$ to be flattened in the direction perpendicular to the line $\mathbf{L}$ not intersecting the curve; (b) a flattened curve obtained by projecting control points local to the span $[u_s, u_e]$.

A surface region is flattened by projecting control points $\mathbf{P}_{i,j}$ onto a plane, π. The $\mathbf{P}_{i,j}$ are replaced by their projections, $\hat{\mathbf{P}}_{i,j}$. In addition to π, one can specify (or default) the following:

- a vector $\mathbf{W}$ defining the line of projection, i.e., $\hat{\mathbf{P}}_{i,j} = \mathbf{P}_{i,j} + \alpha\mathbf{W}$ (α can be negative or positive); this vector can be defaulted to be perpendicular to π;

- a region, $\mathcal{R}_{uv}$, in the uv space of $\mathbf{S}(u,v)$ defined by a boundary curve, $\mathbf{C}_{uv}(w) = (u(w), v(w))$; $\mathbf{P}_{i,j}$ is projected only if its influence is local to the region $\mathcal{R}_{uv}$. If $[u_\alpha, u_\beta] \times [v_\gamma, v_\delta]$ is the minmax rectangle containing $\mathcal{R}_{uv}$, then knot refinement should be applied to the intervals $[u_i, u_j]$ and $[v_k, v_\ell]$, where u_i and v_k are the largest knots satisfying $u_i < u_\alpha$ and $v_k < v_\gamma$, and u_j and v_ℓ are the smallest knots satisfying $u_\beta < u_j$ and $v_\delta < v_\ell$;
- a region, $\mathcal{R}_\pi$, on the plane π, defined by a boundary curve, $\mathbf{C}_\pi(w)$; a control point, $\mathbf{P}_{i,j}$, is replaced by its projection, $\hat{\mathbf{P}}_{i,j}$, only if $\hat{\mathbf{P}}_{i,j}$ lies in the region $\mathcal{R}_\pi$.

Most often, $\mathbf{C}_{uv}(w)$ and $\mathbf{C}_\pi(w)$ are simple curves, such as polygons or circles, for which determination of point containment in the respective regions is easy and efficient. If the plane π intersects the surface, then instead of requiring a region $\mathcal{R}_{uv}$, a designer may want to specify that only those $\mathbf{P}_{i,j}$ lying on one side of π be projected. Cobb [Cobb84] suggests the use of additional planes to restrict the region of the control net to be projected. And finally, the projection must produce a connected grid of at least $(p+1) \times (q+1)$ $\hat{\mathbf{P}}_{i,j}$ in order to obtain a flat region in the surface. Figures 11.27a and 11.27b show an example of surface flattening.

11.4.3 BENDING

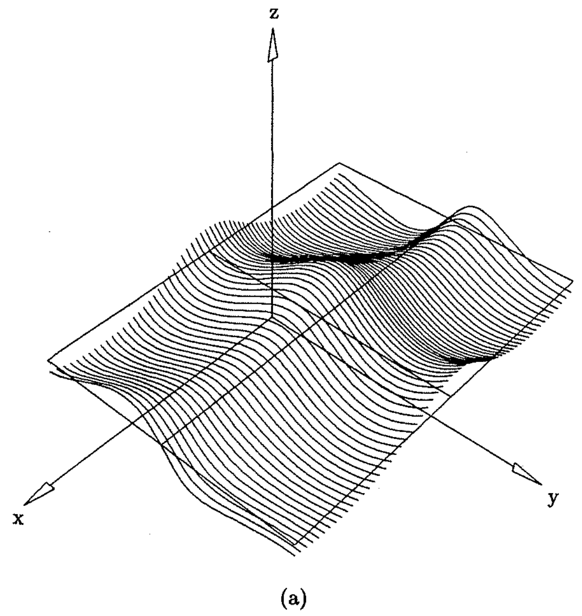
Several authors have investigated circular bending of curves and cylindrical bending of surfaces, e.g., see references [Barr83; Four83; Cobb84]. These authors develop equations based on trigonometry, which map points onto circles. Under these mappings a straight line segment maps onto a circle, and a general curve segment is bent in a circular fashion. Cobb [Cobb84] applies this technique, together with knot refinement, to map control points, thereby bending B-spline curves and surfaces.

We take a more general, geometric approach to bending. Analogous to the other shape operators, bending takes place in three steps:

1. refine the region of the curve or surface to be bent;
2. bend the region by repositioning control points;
3. remove unnecessary knots.

Only Step 2 requires elaboration. Consider curves first (see Figure 11.28). We bend the region of the curve $\mathbf{C}(u)$ between $\mathbf{C}(u_s)$ and $\mathbf{C}(u_e)$ about the *bend curve*, $\mathbf{B}(w)$, by mapping the relevant control points, $\mathbf{P}_i$ of $\mathbf{C}(u)$, "toward" $\mathbf{B}(w)$. Assuming u_s and u_e are knots, only $\mathbf{P}_s, \ldots, \mathbf{P}_{e-p-1}$ are mapped. Denote the image of $\mathbf{P}_i$ by $\hat{\mathbf{P}}_i$. We require one additional point, $\mathbf{O}$, called the *bend center*. Points $\mathbf{R}_i$ are obtained by intersecting each line segment, $\mathbf{O}\mathbf{P}_i$, with $\mathbf{B}(w)$. Finally, fixing a cross ratio, λ, the location of each $\hat{\mathbf{P}}_i$ is determined by λ, $\mathbf{O}$, $\mathbf{R}_i$, and $\mathbf{P}_i$ as follows. Analogous to Eq. (11.22), we have

$$\mathbf{R}_i = (1 - s_i)\mathbf{O} + s_i\mathbf{P}_i$$



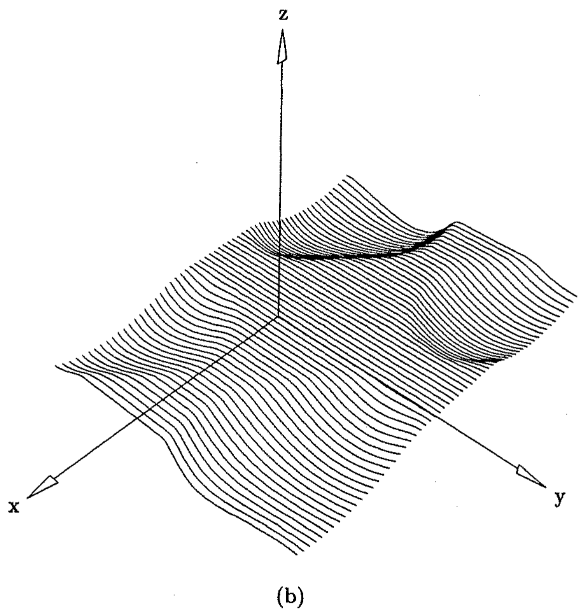


Figure 11.27. Surface flattening. (a) A surface and plane; (b) a flattened surface obtained by projecting all control points lying on one side of the plane.

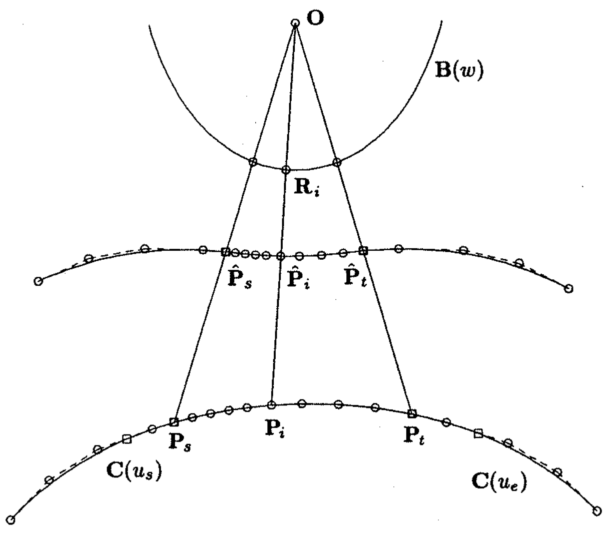


Figure 11.28. Curve bending.

$$\hat{\mathbf{P}}_i = (1-t_i)\mathbf{O} + t_i\mathbf{P}_i \tag{11.56}$$

It follows that

$$s_i = \frac{\mathbf{O}\mathbf{R}_i}{\mathbf{O}\mathbf{P}_i} \tag{11.57}$$

Finally

$$\lambda = \frac{t_i(1-s_i)}{s_i(1-t_i)}$$

yields

$$t_i = \frac{\lambda s_i}{1+(\lambda-1)s_i} \tag{11.58}$$

Substituting t_i into Eq. (11.56) produces $\hat{\mathbf{P}}_i$.

Summarizing, u_s, u_e, $\mathbf{B}(w)$, $\mathbf{O}$, and λ are given, and the bend algorithm computes, in order: $\mathbf{R}_i$ (line/curve intersection), s_i, t_i, and $\hat{\mathbf{P}}_i$. The cross ratio λ can be modified interactively, in a range from $-\infty$ to ∞. Notice that $\lambda = 1$ implies $t_i = s_i$, hence $\hat{\mathbf{P}}_i = \mathbf{R}_i$; i.e., $\lambda = 1$ maps all $\mathbf{P}_i$ onto $\mathbf{B}(w)$, and thus $\mathbf{C}(u)$ assumes the (approximate) shape of $\mathbf{B}(w)$ in the region between $\mathbf{C}(u_s)$ and $\mathbf{C}(u_e)$. For $\lambda = 0$, all $\hat{\mathbf{P}}_i = \mathbf{O}$; and $\hat{\mathbf{P}}_i \to \mathbf{P}_i$ as $\lambda \to \infty$.

Let $t = e-p-1$. After computing $\hat{\mathbf{P}}_s, \ldots, \hat{\mathbf{P}}_t$, the control points $\mathbf{P}_0, \ldots, \mathbf{P}_{s-1}$ and $\mathbf{P}_{t+1}, \ldots, \mathbf{P}_n$ must be rotated and translated into positions which yield natural transitions of the unchanged curve segments on $[u_{\min}, u_s]$ and $[u_e, u_{\max}]$ into the bent segment on $[u_s, u_e]$. Let θ denote the angle through which the bend mapping rotates the vector $\mathbf{P}_t - \mathbf{P}_{t-1}$. The new $\hat{\mathbf{P}}_{t+1}, \ldots, \hat{\mathbf{P}}_n$ are then obtained by rotating the $\mathbf{P}_{t+1}, \ldots, \mathbf{P}_n$ an angle, θ, about the point $\mathbf{P}_t$, and subsequently translating by the vector $\hat{\mathbf{P}}_t - \mathbf{P}_t$. An analogous rotation (defined by the rotation of $\mathbf{P}_{s+1} - \mathbf{P}_s$) and translation must be applied to $\mathbf{P}_0, \ldots, \mathbf{P}_{s-1}$.

Figures 11.29–11.31 show examples of circular and parabolic bends. In Figure 11.29 the effect of varying λ is illustrated. The bend region is symmetric about line $\mathbf{L}$ passing through the bend center, $\mathbf{O}$, and is perpendicular to $\mathbf{C}(u)$. Note that as λ approaches 1 the bend region gradually assumes the shape of the circle. The effect of positioning the center, $\mathbf{O}$, within the circle is shown in Figure 11.30. $\mathbf{O}_1$ is the circle center, $\mathbf{O}_2$ lies on the line $\mathbf{L}$, whereas $\mathbf{O}_3$ is an arbitrary point. Figure 11.31 illustrates the use of a parabolic bend curve to eliminate two unwanted inflection points in a curve.

Finally, we remark that for additional shape control the cross ratio can be a function, $\lambda(u)$, defined for $u \in [u_s, u_e]$. Then for $\mathbf{P}_i$, $\lambda_i = \lambda(t_i)$ is computed, where t_i is the ith node (not related to the t_i of Eq. [11.58]).

We now turn to surface bending. The natural extension of the curve bending technique to surfaces yields a very general and powerful surface modification capability. We start with a general definition of surface bending, but we then restrict our detailed discussion to a simple, but useful, special case.

A surface, $\mathbf{S}(u,v)$, is bent about a bend surface, $\mathbf{B}(s,t)$, either toward a central point, $\mathbf{O}$, or a central curve, $\mathbf{O}(w)$. Let the bend region, $\mathcal{R}$ in uv space, be bounded by the minmax rectangle, $\mathcal{M} = [u_\alpha, u_\beta] \times [v_\gamma, v_\delta]$. After knot refinement on $\mathcal{M}$ the subset of the control points, $\mathbf{P}_{i,j}$, $i = \alpha, \ldots, \beta-p-1$, $j = \gamma, \ldots, \delta-q-1$, whose influence is local to $\mathcal{R}$, are mapped to new locations, $\hat{\mathbf{P}}_{i,j}$, using a cross ratio, λ; a function $\lambda(u,v)$ can also be used. If a point $\mathbf{O}$ is used, then points $\mathbf{R}_{i,j}$

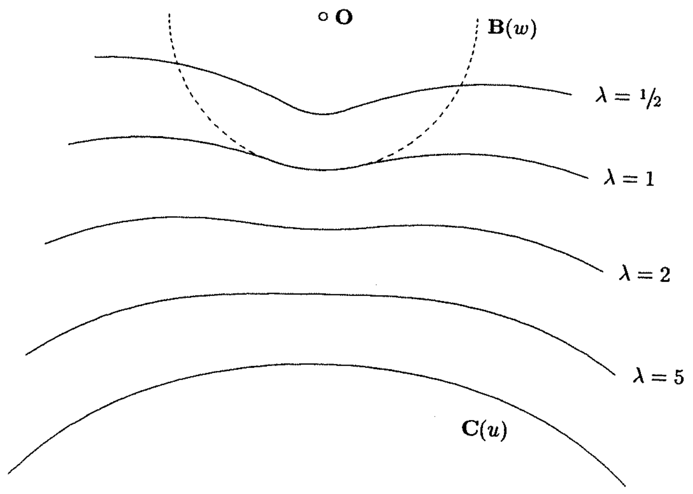


Figure 11.29. Curve bending examples using a circular arc as the bending curve $\mathbf{B}(w)$; the cross ratios are $\lambda = \{1/2, 1, 2, 5\}$.

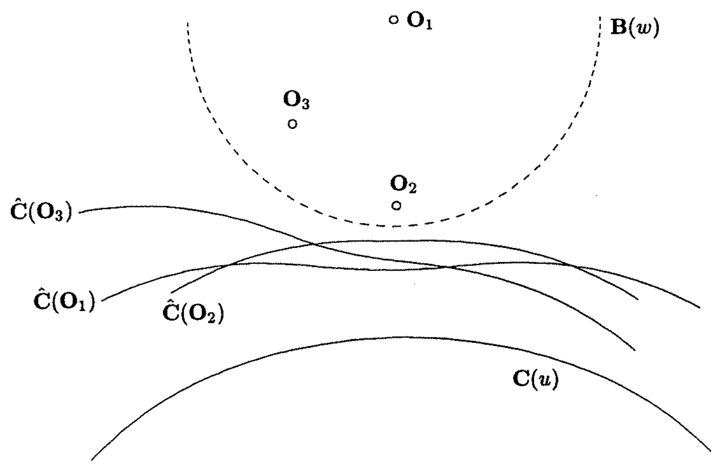


Figure 11.30. Curve bending examples obtained by using different positions of the bend center.

are obtained as the intersections of the line segments, $\mathbf{OP}_{i,j}$, with the surface, $\mathbf{B}(s,t)$. If a curve, $\mathbf{O}(w)$, is given, then parameters $w_{i,j}$ must be computed, and the $\mathbf{R}_{i,j}$ are obtained by intersecting $\mathbf{O}(w_{i,j})\mathbf{P}_{i,j}$ with $\mathbf{B}(s,t)$. The $\hat{\mathbf{P}}_{i,j}$ are then computed using

$$s_{i,j} = \frac{\mathbf{OR}_{i,j}}{\mathbf{OP}_{i,j}} \quad \text{or} \quad s_{i,j} = \frac{\mathbf{O}(w_{i,j})\mathbf{R}_{i,j}}{\mathbf{O}(w_{i,j})\mathbf{P}_{i,j}} \tag{11.59}$$

$$t_{i,j} = \frac{\lambda s_{i,j}}{1 + (\lambda - 1)s_{i,j}} \tag{11.60}$$

Finally

$$\hat{\mathbf{P}}_{i,j} = (1 - t_{i,j})\mathbf{O} + t_{i,j}\mathbf{P}_{i,j} \quad \text{or} \quad \hat{\mathbf{P}}_{i,j} = (1 - t_{i,j})\mathbf{O}(w_{i,j}) + t_{i,j}\mathbf{P}_{i,j} \tag{11.61}$$

Subsequently, the unmapped control points must be rotated and translated into positions which ensure a smooth and natural transition of the bent region into the unbent region. Knot removal is applied as a final step. For example, one obtains a spherical bend by letting $\mathbf{B}(s,t)$ be a sphere, with $\mathbf{O}$ its center. A cylindrical bend is produced if $\mathbf{B}(s,t)$ is a cylinder and $\mathbf{O}(w)$ is its axis.

In its full generality, surface bending is clearly complex. The computation of $\mathbf{R}_{i,j}$ requires line/surface intersections, obtaining suitable parameters $w_{i,j}$ can be difficult, and it is not obvious how to rotate and translate the unmapped control points. We detail here the special case of general cylindrical bends about the full length of an isocurve of a surface; that is, the bend region has one of the forms $[u_\alpha, u_\beta] \times [v_{\min}, v_{\max}]$ or $[u_{\min}, u_{\max}] \times [v_\gamma, v_\delta]$, and $\mathbf{B}(s,t)$ is a general circular cylinder obtained by sweeping a circle along the (possibly curved) axis,

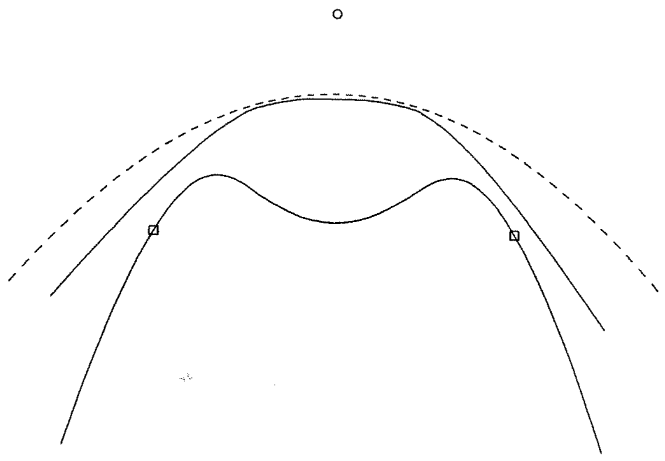


Figure 11.31. Parabolic bending used to eliminate inflexion points.

$\mathbf{O}(w)$. We present bending about an isocurve at fixed $u = u_c$; the case of fixed $v = v_c$ is analogous. Assume that u_c, u_α, and u_β satisfy $u_\alpha < u_c < u_\beta$. Let $\mathbf{C}_{u_c}(v)$ denote the isocurve on $\mathbf{S}(u,v)$ at $u = u_c$ (see Figure 11.32). The axis $\mathbf{O}(w)$ can be defined in either of two ways. It can be defined as the *offset* of

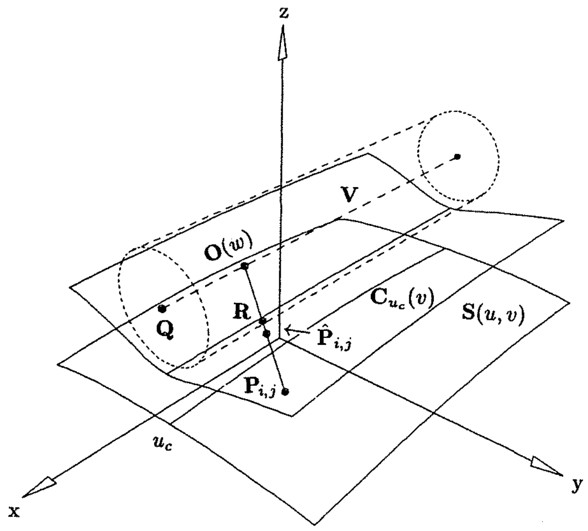


Figure 11.32. Surface bending using a cylinder.

$\mathbf{C}_{u_c}(v)$ along the surface normal, i.e.

$$\mathbf{O}(v) = \mathbf{C}_{u_c}(v) + d\mathbf{N}(u_c, v) \tag{11.62}$$

where $\mathbf{N}(u_c, v)$ is the unit length surface normal vector at (u_c, v) and d is a constant offset distance. We remark that, in general, the shape of $\mathbf{O}(v)$ can be radically different from that of $\mathbf{C}_{u_c}(v)$, and $\mathbf{O}(v)$ cannot be represented precisely as a NURBS curve (e.g., see [Till84; Coqu87b; Hosc88]). However, we assume here that $\mathbf{O}(v)$ is well-behaved and similar in shape to $\mathbf{C}_{u_c}(v)$; for surface bending this is a reasonable assumption. Moreover, we only need to compute selected points on $\mathbf{O}(v)$ (using Eq. [11.62]); we do not require an explicit NURBS representation of $\mathbf{O}(v)$. Finally, we note that in this case the parameter w corresponds to v.

The second method of defining $\mathbf{O}(w)$ is to let it be a straight line lying in the plane defined by $\mathbf{C}_{u_c}(v)$. In this case w does not correspond to v. By computing a least squares plane (defined by the control points of $\mathbf{C}_{u_c}(v)$), this method can be used even for nonplanar $\mathbf{C}_{u_c}(v)$.

$\mathbf{S}(u,v)$ is bent by applying the curve bending method to the $m+1$ rows of control points; i.e., for $j = 0, \ldots, m$, do the following:

1. for $i = s, \ldots, e-p-1$:

   1a. compute $w_{i,j}$ so that $\mathbf{O}(w_{i,j})$ is the point on $\mathbf{O}(w)$ closest to $\mathbf{P}_{i,j}$; if $\mathbf{O}(w) = \mathbf{Q} + w\mathbf{V}$ is a straight line, then $w_{i,j}$ is given by the formula

   $$w_{i,j} = \frac{\mathbf{V}\cdot(\mathbf{P}_{i,j} - \mathbf{Q})}{|\mathbf{V}|^2} \tag{11.63}$$

   If $\mathbf{O}(w)$ is the offset of $\mathbf{C}_{u_c}(v)$, the point to curve projection technique of Section (6.1) can be used; the jth v node, t_j, is a good start point for the Newton iterations;

   1b. then set

   $$\mathbf{R}_{i,j} = \mathbf{O}(w_{i,j}) + r\frac{\mathbf{P}_{i,j} - \mathbf{O}(w_{i,j})}{|\mathbf{P}_{i,j} - \mathbf{O}(w_{i,j})|} \tag{11.64}$$

   where r is the radius of the general circular cylinder $\mathbf{B}(s,t)$;

   1c. compute $\hat{\mathbf{P}}_{i,j}$ using Eqs. (11.59)–(11.61);

2. rotate and translate $\mathbf{P}_{0,j}, \ldots, \mathbf{P}_{s-1,j}$ and $\mathbf{P}_{e-p,j}, \ldots, \mathbf{P}_{n,j}$ into new positions, by a process similar to that used for curve bending.

Figures 11.33 and 11.34 show examples of cylindrical surface bending. In Figure 11.33 a planar surface is bent using $\lambda = \{1/2, 3/2, -3\}$. Note that a positive λ bends the surface toward the cylinder, whereas a negative one bends it away from the cylinder. Figure 11.34 shows cylindrical bends of the same planar surface using a cylinder touching the surface along the isoparametric curve $\mathbf{C}_{u_c}(v)$. Again, the sign of λ allows the surface to be bent in both directions. It is important to note that although the surface is planar, it is not degree (1×1); the surface in Figure 11.34 has degree (3×2). In general, the surface must have degree at least (2×2) in order to avoid discontinuities after control points are repositioned.

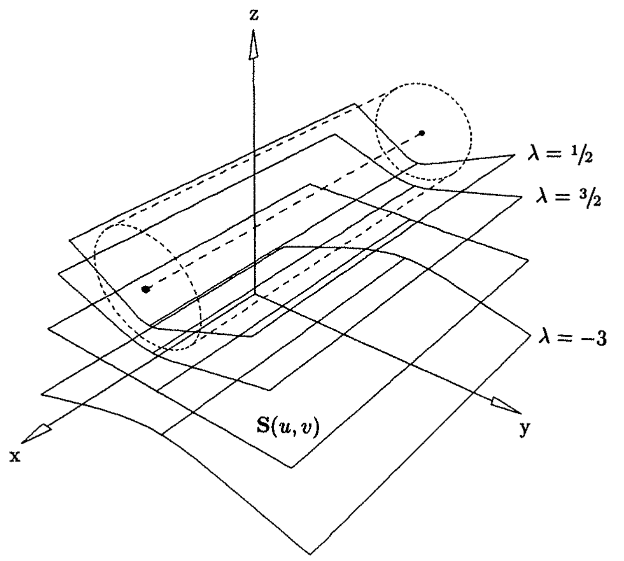


Figure 11.33. Surface bending examples with a nontouching cylinder, $\lambda = \{1/2, 3/2, -3\}$.

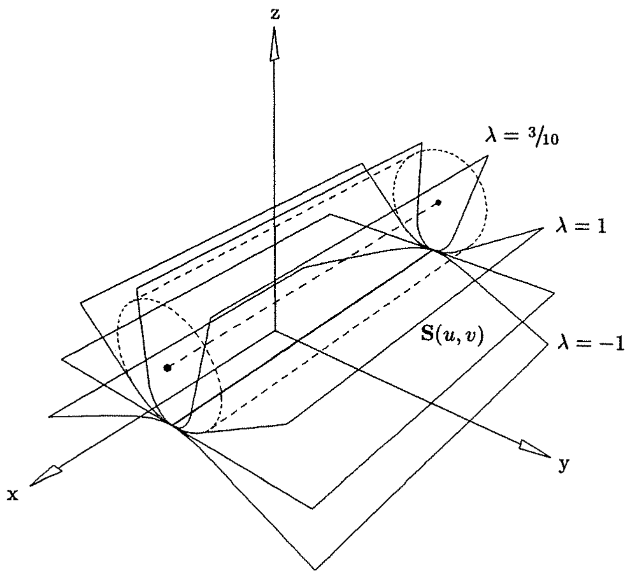


Figure 11.34. Surface bending examples with a touching cylinder, $\lambda = \{3/10, 1, -1\}$.

11.5 Constraint-based Curve and Surface Shaping

Next we present a method of forcing a curve or surface to assume specified derivatives at selected parameter values. The method allows for derivatives of arbitrary order, including zeroth order, hence point constraints are also included. By constraining first and second derivatives, curve/surface tangents and curvature can be controlled. The constraints are satisfied by repositioning control points in a way that minimizes the combined movement of all control points. Either a fully or an underdetermined system of linear equations is solved in a two-step process. This two-step solution allows the constraint-based modifications to be realized at interactive speeds. More material on this topic can be found in [Fowl92, 93].

11.5.1 CONSTRAINT-BASED CURVE MODIFICATION

Let $\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i$ be a NURBS curve. Suppose we are given parameter values $\{u_r\}$, $r = 1, \ldots, R$, and a set of derivative constraints, $\{\Delta\mathbf{D}_r^{(k)}\}$. $\Delta\mathbf{D}_r^{(k)}$ denotes the *change* in the kth derivative at $u = u_r$. k can have any nonnegative value, but typically $k = 0, 1, 2$, because these are the values which control the intuitive geometric properties of position, tangent, and curvature. Notice that we specify constraints in terms of *changes* in derivatives, not in terms of the target values.

Now fix k and r, and denote by $\mathbf{D}_r^{(k)}$ the kth derivative of $\mathbf{C}(u)$ at $u = u_r$. Then

$$\mathbf{D}_r^{(k)} = \sum_{i=0}^{n} R_{i,p}^{(k)}(u_r)\mathbf{P}_i \tag{11.65}$$

where $R_{i,p}^{(k)}$ denotes the kth derivative of the basis function. If the curve is nonrational, the derivatives of the basis functions are computed using Algorithm A2.3. Otherwise, the $R_{i,p}^{(k)}$ are computed utilizing the same technique that was used in Section 4.3 to obtain the derivatives of a rational curve, that is

$$R_{i,p}(u) = \frac{w(u)R_{i,p}(u)}{w(u)} = \frac{A_i(u)}{w(u)}$$

where
$$w(u) = \sum_{j=0}^{n} w_j N_{j,p}(u)$$

Using Leibnitz' Rule to differentiate $A_i(u)$ yields

$$R_{i,p}^{(k)}(u) = \frac{A_i^{(k)}(u) - \sum_{j=1}^{k} \binom{k}{j} w^{(j)}(u) R_{i,p}^{(k-j)}(u)}{w(u)} \tag{11.66}$$

with
$$A_i^{(k)}(u) = w_i N_{i,p}^{(k)}(u)$$

Applying the constraint $\Delta\mathbf{D}_r^{(k)}$ at $u = u_r$, we obtain the new kth derivative

$$\sum_{i=0}^{n} R_{i,p}^{(k)}(u_r)(\mathbf{P}_i + \Delta\mathbf{P}_i) = \mathbf{D}_r^{(k)} + \Delta\mathbf{D}_r^{(k)} \tag{11.67}$$

and it follows that

$$\sum_{i=0}^{n} R_{i,p}^{(k)}(u_r)\Delta\mathbf{P}_i = \Delta\mathbf{D}_r^{(k)} \tag{11.68}$$

Equation (11.68) is one linear equation in the $n+1$ unknowns $\{\Delta\mathbf{P}_i\}$ (actually $p+1$ unknowns, due to the local support property of B-splines). We remark that $\Delta\mathbf{D}_r^{(k)}$ can be the zero vector, in fact this is how one specifies that a point or derivative is not to change. If M is the total number of constraints, $M \geq R$, then applying Eq. (11.68) to each constraint yields a system of M linear equations in $n+1$ unknowns, that is

$$B\,[\Delta\mathbf{P}_i] = \left[\Delta\mathbf{D}_r^{(k)}\right] \tag{11.69}$$

Before we show how to solve Eq. (11.69), the following remarks are important:

- Equation (11.69) assumes some ordering of the constraints; the ordering can be arbitrary, but a natural method is to sort first on r, then on k. That is, suppose the $\Delta\mathbf{D}_r^{(k)}$ are ordered as $\Delta\mathbf{D}_1, \ldots, \Delta\mathbf{D}_M$, and write $\Delta\mathbf{D}_i = \Delta\mathbf{D}_{r_i}^{(k_i)}$. Then $i < j$ implies $r_i \leq r_j$, and additionally, either $r_i < r_j$ or $k_i < k_j$;
- due to the local support property of B-splines and the fact that constraints are usually applied only at a small number of parameter locations, many columns of B may be zero; that is, there are fewer than $n+1$ unknown $\Delta\mathbf{P}_i$. An efficient algorithm should set up the system Eq. (11.69) to contain only those $\Delta\mathbf{P}_i$ which change; denote by N the actual number of unknown $\Delta\mathbf{P}_i$;
- we require a precise solution to Eq. (11.69); however, the system may be under-, fully, or overdetermined, and it may not have a precise solution. For the solution method presented later, we require an under- ($M < N$) or fully ($M = N$) determined system, in which all rows of B are linearly independent. An overdetermined system ($M > N$) occurs when more constraints are specified than the degree p or the number N of affected control points supports. Clearly, no more than $p+1$ constraints may be specified on each knot span. If necessary, knot insertion and/or degree elevation can be used to ensure that a set of constraints produces an under- or fully determined system. If $M \leq N$, then linear dependence of the rows of B is rarely a problem – see [Scho66; Good81; Fowl93] for mathematical details – but a robust algorithm should check for this during the first step of the solution process;
- generally the designer is not expected to specify the parameters $\{u_r\}$ at which the constraints are applied; theoretically these can also be unknowns, but this gives rise to nonlinear equations. Thus, as usual the system should choose the parameters, based on curve points picked by the designer.

Considering these remarks, we now write Eq. (11.69) as

$$B\,[\Delta\mathbf{P}_{i_\ell}] = [\Delta\mathbf{D}_j] \qquad \ell = 1,\ldots,N \quad j = 1,\ldots,M \tag{11.70}$$

or for brevity

$$B\Delta\mathbf{P} = \Delta\mathbf{D} \tag{11.71}$$

For the moment, assume $M < N$. There exist infinitely many solutions to Eq. (11.70). We want the *minimum length solution*, i.e., the one which minimizes the combined movements of all control points (the one which minimizes the length of the N-dimensional vector, $\Delta\mathbf{P}$). From linear algebra, we can write the N vector, $\Delta\mathbf{P}$, in terms of two mutually orthogonal components, one in the row space of B and the other in the null space of B, that is

$$\Delta\mathbf{P} = B^T\lambda + \mathbf{z} \tag{11.72}$$

where λ is some M vector, $B\mathbf{z} = 0$, and $(\lambda^T B)\cdot\mathbf{z} = 0$. It follows that

$$\begin{aligned} |\Delta\mathbf{P}|^2 &= \left(B^T\lambda + \mathbf{z}\right)^T \cdot \left(B^T\lambda + \mathbf{z}\right) \\ &= \left(B^T\lambda\right)^T \cdot \left(B^T\lambda\right) + \mathbf{z}^T\cdot\mathbf{z} \\ &= \left|B^T\lambda\right|^2 + |\mathbf{z}|^2 \end{aligned} \tag{11.73}$$

From Eqs. (11.71) and (11.72) we obtain

$$\begin{aligned} \Delta\mathbf{D} &= B\Delta\mathbf{P} \\ &= B\left(B^T\lambda + \mathbf{z}\right) \\ &= BB^T\lambda + B\mathbf{z} \\ &= BB^T\lambda \end{aligned} \tag{11.74}$$

and therefore

$$\lambda = (BB^T)^{-1}\Delta\mathbf{D} \tag{11.75}$$

Since the components of $\mathbf{z}$ are irrelevant to the system, yet they contribute to the length of $\Delta\mathbf{P}$ (by Eq. [11.73]), we obtain the minimum-length solution to Eq. (11.70) by setting $\mathbf{z} = 0$. Substituting Eq. (11.75) into (11.72) leads to

$$\Delta\mathbf{P} = A\Delta\mathbf{D} \qquad \text{where } A = B^T(BB^T)^{-1} \tag{11.76}$$

The matrix BB^T is nonsingular if the rows of B are linearly independent. Finally, if $M = N$ there is a unique solution to Eq. (11.70) given by

$$\Delta\mathbf{P} = A\Delta\mathbf{D} \qquad \text{where } A = B^{-1} \tag{11.77}$$

We remark that it is also possible (and quite useful) to restrict certain control points from changing. This is accomplished (mathematically) simply by setting the corresponding column of B to zero. Algorithmically, this means that the corresponding $\Delta\mathbf{P}_{i_\ell}$ does not appear in Eq. (11.70), and N is reduced by 1. Normally, a constraint affects $p+1$ control points, and up to $p+1$ constraints can be specified in any span. If a control point is restricted not to move, then this reduces to p the number of constraints that can be specified in any span influenced by this control point. We give examples of this below.

Equation (11.76) (or Eq. [11.77]) represents the first step in the solution process. Before proceeding to the second step, we summarize the sequence of events which brought us to this point:

1. the designer picks points on the curve, yielding parameters $\{u_r\}$ (determined by the system) at which constraints are to be imposed; he simultaneously indicates what types of constraints are to be imposed at each parameter, but actual constraints need not be specified at this stage;
2. the system sets up Eq. (11.70); if it is overdetermined, the system either informs the designer so that he can modify his selections, or it adds control points by means of knot insertion and/or degree elevation in such a way that the resulting equation system is under- or fully determined;
3. the matrix A is then computed, either by Eq. (11.76) or (11.77).

The designer can now interactively specify constraints $[\Delta\mathbf{D}_j]$. With each change in constraints, the matrix A must be multiplied with the new vector $[\Delta\mathbf{D}_j]$ to yield the changes in control point locations, $[\Delta\mathbf{P}_{i_\ell}]$. The relevant control points are then updated by

$$\hat{\mathbf{P}}_{i_\ell} = \mathbf{P}_{i_\ell} + \Delta\mathbf{P}_{i_\ell} \qquad \ell = 1,\ldots,N \tag{11.78}$$

Example

Ex11.1 Figure 11.35a shows a cubic curve defined by $\mathbf{P}_0,\ldots,\mathbf{P}_7$, and $U = \{0,0,0,0,{}^1\!/_5,{}^2\!/_5,{}^3\!/_5,{}^4\!/_5,1,1,1,1\}$. We apply one point constraint at $u = {}^1\!/_2$

$$\Delta\mathbf{D}_1 = \Delta\mathbf{D}_1^{(0)} = (0.04, -0.08, 0)$$

Then
$$B\Delta\mathbf{P} = [\Delta\mathbf{D}_1]$$

where B is the 1×4 matrix

$$B = \left[R_{2,3}\left(\frac{1}{2}\right), R_{3,3}\left(\frac{1}{2}\right), R_{4,3}\left(\frac{1}{2}\right), R_{5,3}\left(\frac{1}{2}\right)\right]$$

and $\Delta\mathbf{P}$ is the 4×1 column vector with components

$$\Delta\mathbf{P}_2, \Delta\mathbf{P}_3, \Delta\mathbf{P}_4, \Delta\mathbf{P}_5$$

Hence, BB^T is a 1×1 matrix, and $A = B^T(BB^T)^{-1}$ is a 4×1 matrix

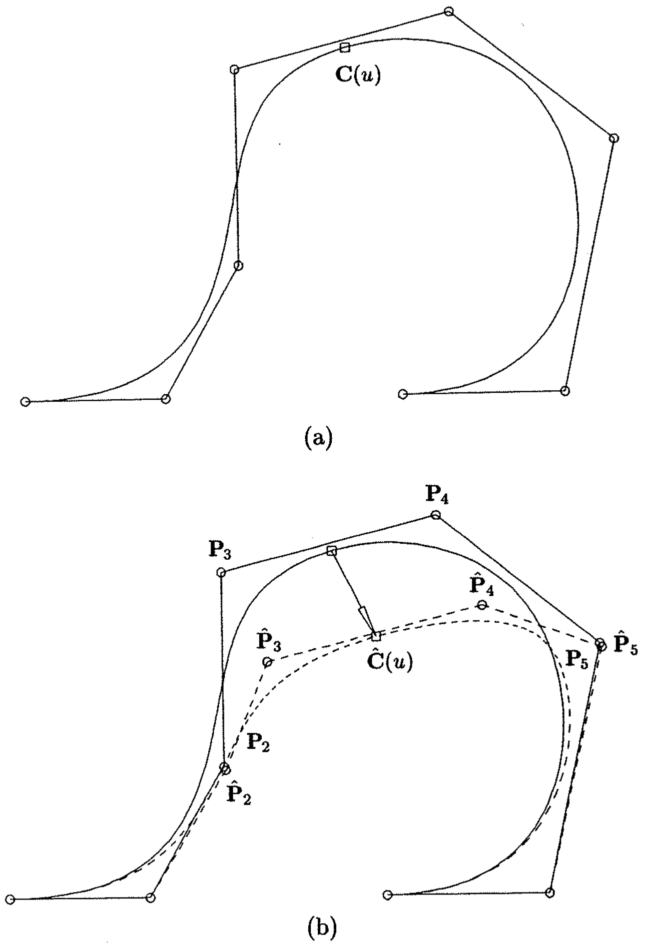


Figure 11.35. Constraint-based curve modification using positional constraints. (a) The original cubic curve with point to move; (b) all four local control points are allowed to move; (c) only $\mathbf{P}_3$ and $\mathbf{P}_4$ are allowed to move; (d) only $\mathbf{P}_3$ is allowed to move.

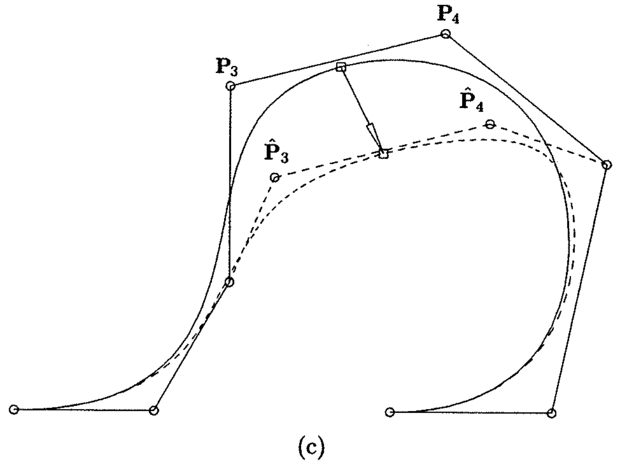
with components

$$a_0, a_1, a_2, a_3$$

It follows that

$$\Delta\mathbf{P}_i = a_{i-2}\Delta\mathbf{D}_1 \qquad i = 2,3,4,5$$

The result of applying the $\Delta\mathbf{P}_i$ is shown in Figure 11.35b. Note that $\mathbf{P}_3$ and $\mathbf{P}_4$ move the most, because these control points have the greatest



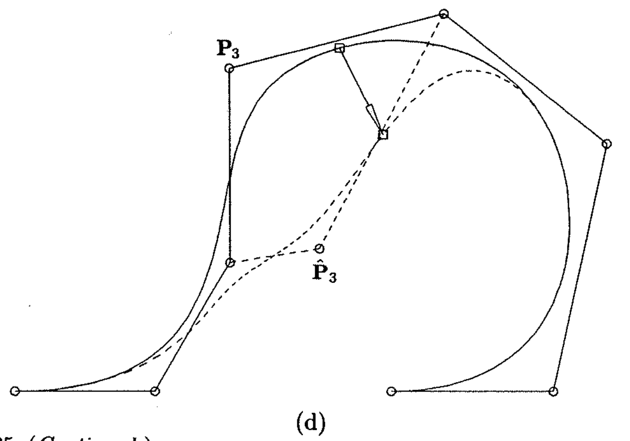


Figure 11.35. (*Continued.*)

effect on the curve at $u = 1/2$. We can restrict $\mathbf{P}_2$ and $\mathbf{P}_5$ from moving. Then B is the 1×2 matrix

$$B = \left[R_{3,3}\left(\frac{1}{2}\right), R_{4,3}\left(\frac{1}{2}\right)\right]$$

and $\Delta\mathbf{P}$ has components $\Delta\mathbf{P}_3$ and $\Delta\mathbf{P}_4$. A has components a_0 and a_1 (not the same as a_0, a_1 above), and

$$\Delta\mathbf{P}_i = a_{i-3}\Delta\mathbf{D}_1 \qquad i = 3, 4$$

Figure 11.35c shows the result of allowing only $\mathbf{P}_3$ and $\mathbf{P}_4$ to change, and Figure 11.35d illustrates the case in which only $\mathbf{P}_3$ is allowed to move.

This example clearly shows that for a given set of point and derivative constraints there is often flexibility in specifying how many and which control points should change, and this choice affects both the extent and the appearance (e.g., symmetry) of the corresponding curve change. However, dealing with control points can be less intuitive to a designer than imposing pure point and derivative constraints on a curve. A good user interface should package both a reasonable set of defaults and convenient handles for the more sophisticated designer to restrict control point movement. Finally, notice that any control point restrictions must be worked into Eq. (11.70) and into the first stage of the solution process.

In Figures 11.36a–11.36c point and derivative constraints are illustrated. Figure 11.36a shows the application of a derivative constraint only, while allowing $\mathbf{P}_3$, $\mathbf{P}_4$, and $\mathbf{P}_5$ to move. The magnitude of the first derivative vector is increased at $u = 0.556$, resulting in a local tension effect. It is interesting to observe what happens if only one control point is allowed to move, as shown in Figure 11.36b. The curve assumes the new derivative, but it suffers a positional displacement due to the absence of a positional constraint. In Figure 11.36c this problem is remedied by constraining the curve not to move at $u = 0.556$, and to assume a new derivative obtained by decreasing the magnitude of the old one. Two control points are allowed to move in order to obtain a system of equations which is not overdetermined.

In closing, we remark that Fowler and Bartels [Fowl93] give a different (but equivalent) method to represent matrix A in Eqs. (11.76) and (11.77). It is somewhat more complicated mathematically but is more efficient in some cases.

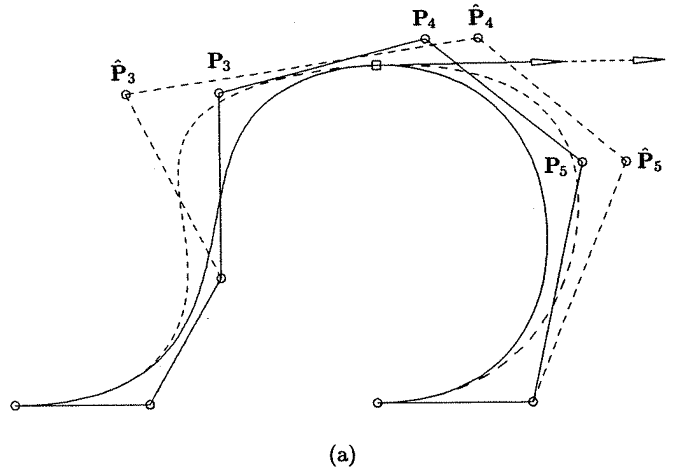
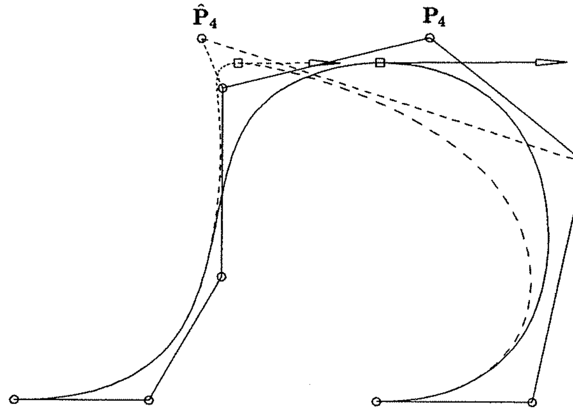
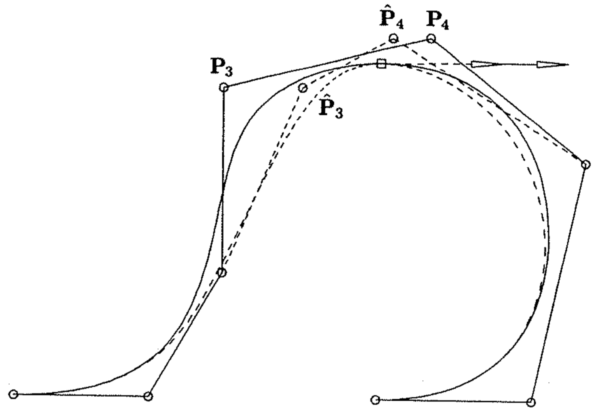


Figure 11.36. Constraint-based curve modification using positional and derivative constraints. (a) Derivative constraints are applied and $\mathbf{P}_3, \mathbf{P}_4$, and $\mathbf{P}_5$ are allowed to move; (b) derivative constraints are applied and only $\mathbf{P}_4$ is allowed to move; (c) derivative and positional constraints are applied, and $\mathbf{P}_3$ and $\mathbf{P}_4$ are allowed to move.



(b)



(c)

Figure 11.36. (*Continued.*)

11.5.2 CONSTRAINT-BASED SURFACE MODIFICATION

Let $\mathbf{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,p;j,q}(u,v)\mathbf{P}_{i,j}$ be a (p,q)th degree NURBS surface. In this section we choose to write $\mathbf{S}(u,v)$ in the form

$$\mathbf{S}(u,v) = \sum_{s=0}^{(n+1)(m+1)-1} R_s(u,v)\mathbf{P}_s \tag{11.79}$$

where

$$s = j(n+1) + i \tag{11.80}$$

We call s the *absolute address* corresponding to (i,j). Notice that

$$\mathbf{P}_s = \mathbf{P}_{i,j} \qquad R_s(u,v) = R_{i,p;j,q}(u,v) \tag{11.81}$$

where

$$i = s \bmod n+1 \qquad j = \frac{s}{n+1} \tag{11.82}$$

If $\mathbf{S}(u,v)$ is nonrational, then $R_s(u,v) = N_{i,p}(u)N_{j,q}(v)$, where i and j are defined by Eq. (11.82). Now suppose we are given a set of point and partial derivative constraints, $\left\{\Delta\mathbf{D}_r^{(k,\ell)}\right\}$, $k,\ell \geq 0$, specified at the parameter locations $\{(u,v)_r\}$, $r = 1,\ldots,R$. Then for each r we have

$$\sum_{s=0}^{(n+1)(m+1)-1} R_s^{(k,\ell)}\big((u,v)_r\big)\Delta\mathbf{P}_s = \Delta\mathbf{D}_r^{(k,\ell)} \tag{11.83}$$

The partial derivatives $R_s^{(k,\ell)}\big((u,v)_r\big)$ can be computed in exactly the same way as the $\mathbf{S}^{(k,\ell)}(u,v)$ described in Section 4.5. Then, as was the case for curves, we can set up the $M \times N$ system of linear equations

$$B\,[\Delta\mathbf{P}_{s_\alpha}] = \left[\Delta\mathbf{D}_r^{(k,\ell)}\right] \qquad \alpha = 1,\ldots,N \tag{11.84}$$

M is the number of constraints ($M \geq R$), and N is the number of control points which are free to move. In general, a constraint involves $(p+1)\times(q+1)$ control points, but a control point can be held fixed simply by not including it in Eq. (11.84). Equation (11.84) can be under-, fully, or overdetermined. If overdetermined, then knot insertion and/or degree elevation can be used to obtain an under- or fully determined system.

Equation (11.84) is solved in precisely the same manner as for curves, i.e.

$$[\Delta\mathbf{P}_{s_\alpha}] = A\left[\Delta\mathbf{D}_r^{(k,\ell)}\right] \tag{11.85}$$

where

$$A = B^T(BB^T)^{-1} \tag{11.86}$$

(or $A = B^{-1}$ if Eq. [11.84] is fully determined). This yields a minimum combined movement of control points. The final step is

$$\hat{\mathbf{P}}_{s_\alpha} = \mathbf{P}_{s_\alpha} + \Delta\mathbf{P}_{s_\alpha} \qquad \alpha = 1,\ldots,N \tag{11.87}$$

Although the mathematics for curve and surface constraint-based modifications is virtually the same, the software required to set up the matrix A is more complicated for surfaces. Algorithm `A11.1` illustrates one possible solution. In addition to data defining the surface, a list (`CList`) of structures is input. The list contains R nodes; each node contains a parameter pair $(u,v)_r$ and a list of the types of constraints to be applied at that point (actual constraints are not input). Output consists of:

`M`	: the number of constraints (and columns of `A`);
`N`	: the number of control points which change (and rows of `A`);
`A[][]`	: the matrix A;
`CpInd[]`	: the absolute addresses, s_α, of the control points which change.

Furthermore, one of the following integer values is returned:

0 : no error;

1 : the system is overdetermined;

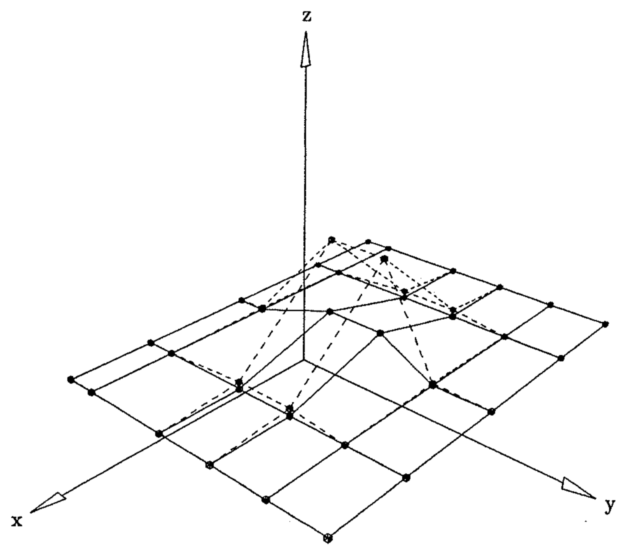
2 : the matrix is singular (rows of B are not linearly independent).

In order to build `A` and `CpInd`, three local two-dimensional arrays, `B[M][Nmax]`, `FreeCp[p+1][q+1]`, and `Map[n+1][m+1]`, and one four-dimensional array, `Funs`, and two new routines, `FreeCtrlPts(u,v,FreeCp)` and `RatDersBasisFuns(u,v,Funs)`, are used. `B` contains the matrix B of Eq. (11.84). `Map[i][j]`$= -1$ if $\mathbf{P}_{i,j}$ is not a part of the system defined by Eq. (11.84); otherwise, `Map[i][j]`$= \alpha \geq 0$ is an index for `CpInd`, and `CpInd[`α`]`$= s$ is the absolute address corresponding to (i,j). For a given (u,v), `FreeCtrlPts()` returns the Boolean array `FreeCp` that indicates which of the possible $(p+1) \times (q+1)$ control points influencing the surface at (u,v) are free to move. This routine can be implemented in a number of meaningful ways, thus we refrain from elaborating on it here. The routine `RatDersBasisFuns(u,v,Funs)` returns the basis functions and their partial derivatives. `Funs[k][l][i][j]` is the (k,ℓ)th partial derivative of the (i,j)th basis function.

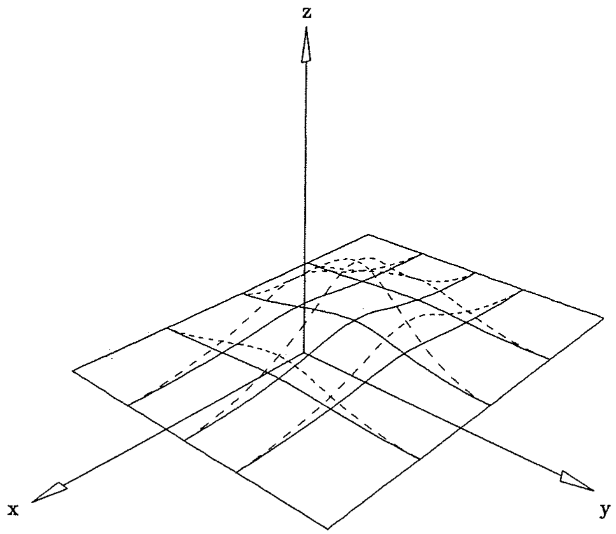
```
ALGORITHM A11.1
ConstBasedSurfMod1(n,m,p,q,U,V,Pw,CList,R,M,N,A,CpInd)
  { /*  Build the A matrix for surface constraints  */
    /*  Input:  n,m,p,q,U,V,Pw,CList,R  */
    /*  Output: M,N,A,CpInd  */
  Loop through the list CList and count constraints
    to obtain M.
  Nmax = Max((n+1)*(m+1),R*(p+1)*(q+1)); /* Max possible N */
  for (i=0; i<M; i++)
    for (j=0; j<Nmax; j++)   B[i][j] = 0.0;
  for (i=0; i<=n; i++)
    for (j=0; j<=m; j++)   Map[i][j] = -1;
  brow = 0;   N = 0;   /* N is computed in this loop */
  for (r=1; r<=R; r++)
    {
    Extract (u,v) value from the rth list node.
    FreeCtrlPts(u,v,FreeCp);
    uspan = FindSpan(n,p,u,U);
    vspan = FindSpan(m,q,v,V);
    for (i=0; i<=p; i++)
```

```
      for (j=0; j<=q; j++)
        if (FreeCp[i][j])
          { /* This ctrl pt is free to move */
          iup = i+uspan-p;   jvq = j+vspan-q;
          if (Map[iup][jvq] == -1)
            { /* Ctrl pt not yet in Eq. system */
            Map[iup][jvq] = N;
            CpInd[N] = jvq*(n+1)+iup;
            N = N+1;
            }
          }
    /* Compute required functions */
    RatDersBasisFuns(u,v,Funs);
    Loop through each type of constraint for this (u,v) and do:
      {
      Set $k$ and $\ell$ indicating which derivative.
      for (i=0; i<=p; i++)
        for (j=0; j<=q; j++)
          {
          alf = Map[i+uspan-p][j+vspan-q];
          if (alf >= 0)  B[brow][alf] = Funs[k][l][i][j];
          }
      brow = brow+1;
      }  /* End of loop through each type of constraint */
    }  /* End of for-loop: r=1,...,R */
  if (M > N)  return(1);   /* system overdetermined */
  if (M == N)
    if ($B$ singular)   return(2);
      else
      {
      A = $B^{-1}$;   return(0);
      }
  if ($BB^T$ singular)   return(2);
    else
    {
    A = $B^T(BB^T)^{-1}$;   return(0);
    }
  }
```

Figures 11.37–11.39 show examples of applying surface constraints. In Figures 11.37a and 11.37b a point constraint is applied at $u = v = 1/2$, allowing all control points affecting the surface at this point to move. Figure 11.37a shows the new control net, and Figure 11.37b illustrates the change in surface shape. Note that the control points around the periphery of the local domain move very little. Figures 11.38a and 11.38b show what happens if only $\mathbf{P}_{1,2}$, $\mathbf{P}_{2,2}$, and $\mathbf{P}_{4,2}$ are

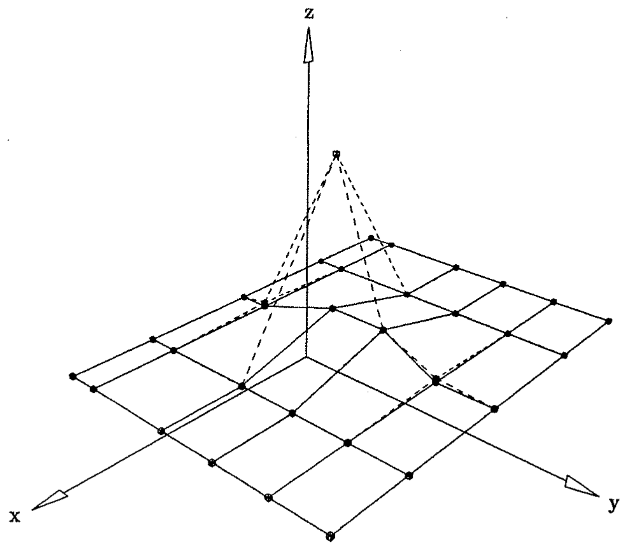


(a)

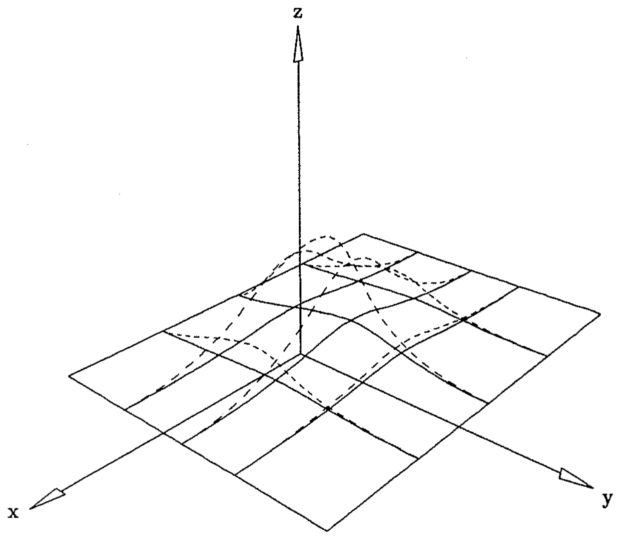


(b)

Figure 11.37. Constraint-based surface modification with a positional constraint. (a) A surface net showing all local control points that move; (b) a surface with change in shape.

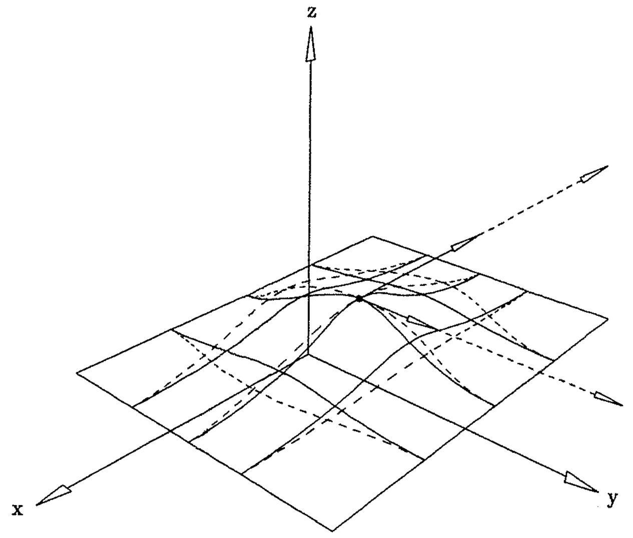


(a)

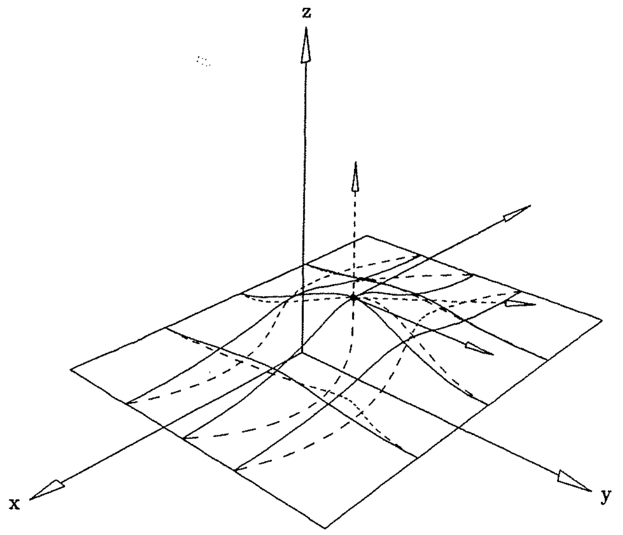


(b)

Figure 11.38. Constraint-based surface modification with a positional constraint. (a) A surface net showing that only $\mathbf{P}_{1,2}, \mathbf{P}_{2,2}$, and $\mathbf{P}_{4,2}$ are allowed to move; (b) a surface with change in shape.



(a)



(b)

Figure 11.39. Constraint-based surface modification with positional and derivative constraints. (a) An increase in partial derivative magnitudes induces a local tension effect; (b) a change in partial derivative directions induces a local twist effect.

allowed to move. Finally, in Figures 11.39a and 11.39b partial derivative and positional constraints are applied. In Figure 11.39a the magnitudes of the partials are increased, resulting in a tension effect, whereas in Figure 11.39b the partials are rotated about the point $\mathbf{S}(^1/_2, ^1/_2)$, yielding a twist in the surface shape.

In closing, we remark that Fowler [Fowl92] gives an alternative (equivalent) solution for solving for the $\Delta\mathbf{P}_{i,j}$, which makes use of the tensor product structure of $\mathbf{S}(u,v)$. It is less general than this method, but in some cases it is more efficient.