CHAPTER

TEN

Advanced Surface Construction Techniques

10.1 Introduction

In this chapter we cover several advanced surface construction techniques, namely swung, skinned, swept, Gordon, and Coons surfaces. Roughly speaking, the idea in this chapter is to take one or two curves, or sets of curves, and to create a NURBS surface which interpolates these curves; i.e., the given curves are isoparametric curves in the NURBS surface. Notice that this is fundamentally different from the interpolation to discrete point and derivative data presented in Chapter 9. In some of the constructions, it may not be obvious to the reader that the surfaces created do indeed satisfy the desired constraints. We leave it as an exercise for the reader to convince himself of this, either by mathematical proof or by software implementation.

10.2 Swung Surfaces

A swung surface is a generalization of a surface of revolution. Let

$$\mathbf{P}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i \tag{10.1}$$

be a *profile curve* defined in the xz plane, and let

$$\mathbf{T}(v) = \sum_{j=0}^{m} R_{j,q}(v)\mathbf{T}_j \tag{10.2}$$

be a *trajectory curve* defined in the xy plane (see Figure 10.1). Denoting the nonzero coordinate functions of $\mathbf{P}(u)$ and $\mathbf{T}(v)$ by $P_x(u)$, $P_z(u)$, $T_x(v)$, and $T_y(v)$, we define the swung surface by [Wood87]

$$\mathbf{S}(u,v) = \big(\alpha P_x(u)T_x(v), \alpha P_x(u)T_y(v), P_z(u)\big) \tag{10.3}$$

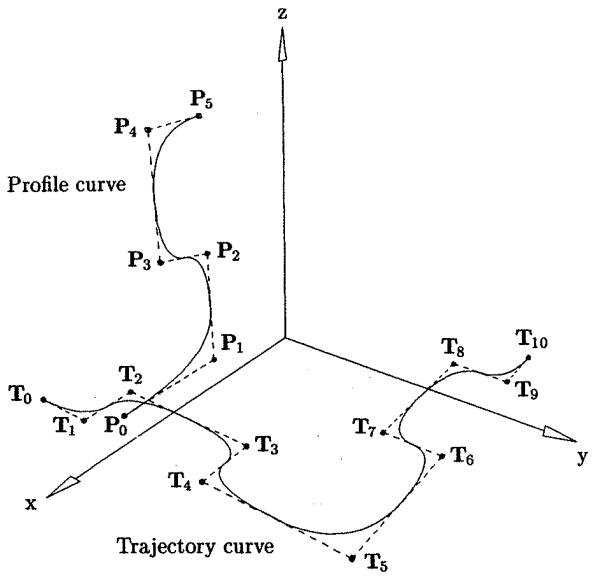


Figure 10.1. Profile and trajectory curves to define a swung surface.

Geometrically, $\mathbf{S}(u,v)$ is obtained by swinging $\mathbf{P}(u)$ about the z-axis and simultaneously scaling it according to $\mathbf{T}(v)$. α is an arbitrary scaling factor. Fixing u yields curves having the shape of $\mathbf{T}(v)$ but scaled in the x and y directions. Fixing $v = v_0$, the isoparametric curve, $\mathbf{C}_{v_0}(u)$, is obtained by rotating $\mathbf{P}(u)$ into the plane containing the vector $\big(T_x(v_0), T_y(v_0), 0\big)$, and scaling the x and y coordinates of the rotated curve with the factor $\alpha|\mathbf{T}(v_0)|$. The z coordinate remains unscaled. It follows from the transformation invariancy property of NURBS (P4.24) that $\mathbf{S}(u,v)$ has a NURBS representation given by

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,p;j,q}(u,v)\,\mathbf{Q}_{i,j} \tag{10.4}$$

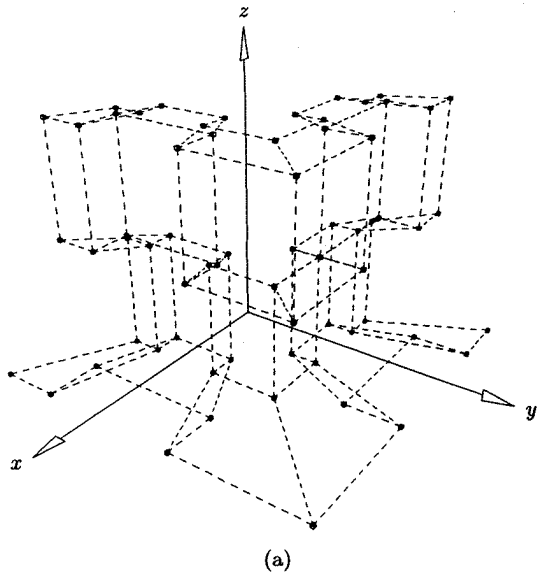
where
$$\mathbf{Q}_{i,j} = (\alpha P_{i,x} T_{j,x}, \alpha P_{i,x} T_{j,y}, P_{i,z}) \tag{10.5}$$

and
$$w_{i,j} = w_i w_j \tag{10.6}$$

The U and V knot vectors for $\mathbf{S}(u,v)$ are those defining $\mathbf{P}(u)$ and $\mathbf{T}(v)$. Figures 10.2a–10.2c show examples of swinging. Figure 10.2a shows the control points of the swung surface defined by the curves in Figure 10.1 using $\alpha = 0.12$. The surface in Figure 10.2b is nonrational, whereas the surface of Figure 10.2c was obtained by applying the weights

$$w_1^P = w_4^P = 5 \qquad w_1^T = w_4^T = w_6^T = w_9^T = 5$$

where superscripts P and T refer to profile and trajectory curves, respectively.



(a)

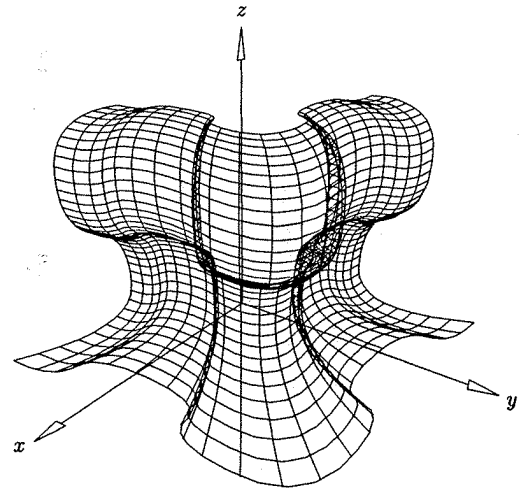
Figure 10.2. Swung surfaces. (a) Control net; (b) swung surface with no weights applied; (c) swung surface with some weights applied.

Note that both $\mathbf{P}(u)$ and $\mathbf{T}(v)$ can be either open or closed; correspondingly, $\mathbf{S}(u,v)$ can be open or closed in both u and v directions. Choosing $\alpha = 1$ and $\mathbf{T}(v)$ to be the circle with radius 1, centered at the origin, yields the surface of revolution given in Chapter 8.

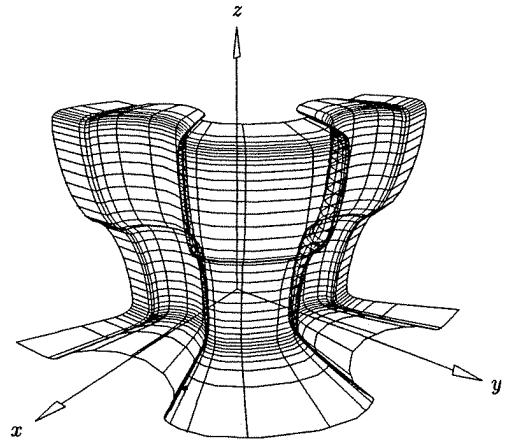
10.3 Skinned Surfaces

Let $\{\mathbf{C}_k(u)\}$, $k = 0, \ldots, K$, be a set of curves. We call the $\mathbf{C}_k(u)$ *section curves.* In practice, they are usually planar cross sections, in the u direction, of the surface to be constructed. However, they can be three-dimensional. Furthermore, we do not assume planarity in this section. Skinning is a process of blending the section curves together to form a surface. The blend direction is the v direction, sometimes called the *longitudinal direction.* Although approximation across the section curves can be used, skinning methods usually interpolate through the $\mathbf{C}_k(u)$, with the result that the $\mathbf{C}_k(u)$ are isoparametric curves on the resulting skinned surface.

Skinning is simply a newer term for *lofting*, which dates back many decades, before computers. It was, and still is, widely used in the shipbuilding, automotive, and aircraft industries. One of the earliest computerized systems incorporating a lofting procedure was CONSURF, developed by A. Ball at the British Aircraft Corporation [Ball74, 75, 77]. That system was based on rational cubic



(b)



(c)

Figure 10.2. (*Continued.*)

curves. Filip and T. Ball [Fili89] describe a procedural method for lofting based on arbitrary parametric section curves. Detailed descriptions of skinning using B-splines are given in references [Till83; Wood87, 88; Hohm91].

Based on B-splines, we define skinning as follows. Let

$$\mathbf{C}_k^w(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_{i,k}^w \qquad k = 0, \ldots, K \tag{10.7}$$

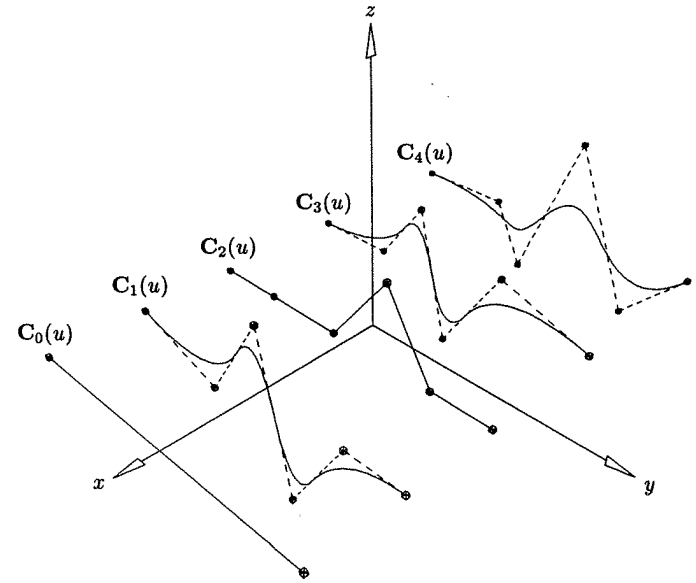
be the rational or nonrational section curves. We assume that all $\mathbf{C}_k^w(u)$ are defined on the same knot vector, U, and have common degree p. If necessary, the curves can be brought to a common p and U, as described in Section 8.4 (in fact, the ruled surface is a special case of a skinned surface). Then, for the v direction a degree q is chosen, and parameters $\{\bar{v}_k\}$, $k = 0, \ldots, K$, and a knot vector, V, are computed. These are then used to do $n+1$ curve interpolations across the control points of the section curves, yielding the control points $\mathbf{Q}_{i,j}^w$ of the skinned surface. Therefore, $\mathbf{Q}_{i,j}^w$ is the jth control point of the interpolating curve through $\mathbf{P}_{i,0}^w, \ldots, \mathbf{P}_{i,K}^w$. Note that if even one of the $\mathbf{C}_k^w(u)$ is rational, then the v direction interpolations through the $\mathbf{P}_{i,k}^w$ are carried out in four-dimensional space; otherwise, the three-dimensional points $\mathbf{P}_{i,k}$ are interpolated.

Figures 10.3a–10.3d show the skinning process. The original section curves are shown in Figure 10.3a and are defined on the knot vectors

$$\mathbf{C}_0(u): \quad \{0,0,1,1\} \qquad (p=1)$$

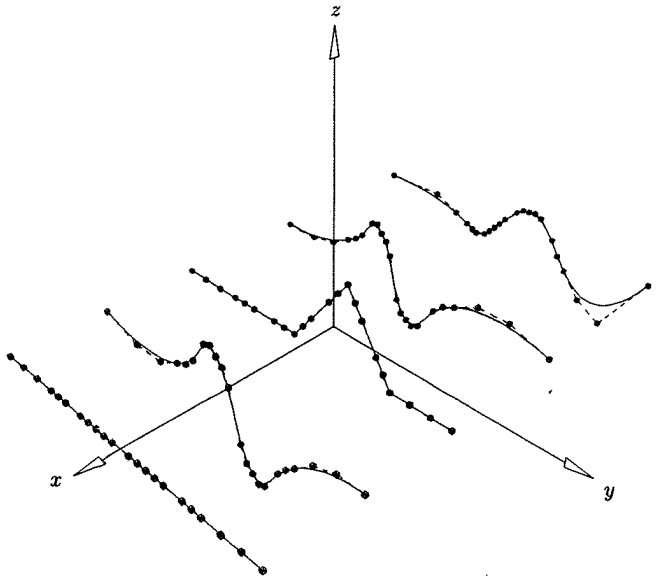
$$\mathbf{C}_1(u): \quad \left\{0,0,0,\tfrac{3}{10},\tfrac{1}{2},\tfrac{7}{10},1,1,1\right\} \qquad (p=2)$$

$$\mathbf{C}_2(u): \quad \left\{0,0,\tfrac{1}{5},\tfrac{2}{5},\tfrac{3}{5},\tfrac{4}{5},1,1\right\} \qquad (p=1)$$

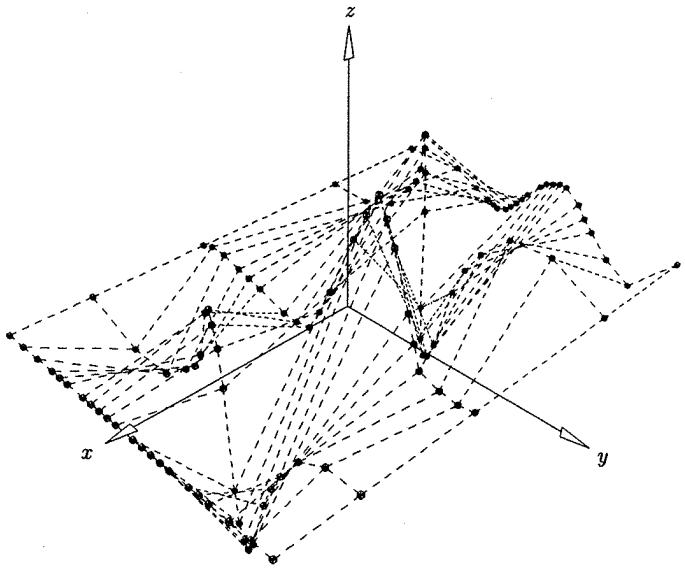


(a)

Figure 10.3. The process of surface skinning. (a) Cross-sectional curves; (b) cross-sectional curves made compatible; (c) control points; (d) skinned surface.

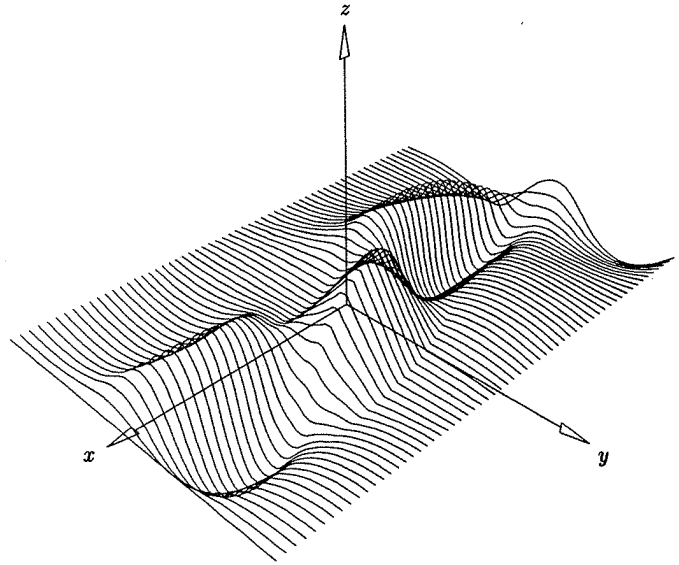


(b)



(c)

Figure 10.3. (*Continued.*)



(d)

Figure 10.3. (*Continued.*)

$$\mathbf{C}_3(u): \quad \left\{0,0,0,\frac{3}{10},\frac{1}{2},\frac{7}{10},1,1,1\right\} \quad (p=2)$$

$$\mathbf{C}_4(u): \quad \left\{0,0,0,0,\frac{3}{10},\frac{7}{10},1,1,1,1\right\} \quad (p=3)$$

After degree raising and knot refinement, the common knot vector is

$$\left\{0,0,0,0,\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{3}{10},\frac{3}{10},\frac{2}{5},\frac{2}{5},\frac{2}{5},\frac{1}{2},\frac{1}{2},\right.$$
$$\left.\frac{3}{5},\frac{3}{5},\frac{3}{5},\frac{7}{10},\frac{7}{10},\frac{4}{5},\frac{4}{5},\frac{4}{5},1,1,1,1\right\}$$

This raises the number of control points in the u direction to 22. The compatible curves are shown in Figure 10.3b. Note the significant number of new control points, the price one pays for flexibility. Figure 10.3c shows the control net of the skinned surface that is depicted in Figure 10.3d.

Although simple to formulate, it is quite difficult to implement a robust and usable skinning capability; and interactively designing skinned surfaces is often a tedious and iterative process. Surface shape is difficult to control; it depends on the number, shape, and positioning of the section curves (all of which the interactive designer usually controls), as well as on the method used for v-directional interpolations (which is usually embedded in the software and the designer cannot control). Often the design process consists of starting with a small number

of section curves and then iteratively skinning, taking plane cuts to obtain additional section curves, modifying the new curves to obtain more desirable shapes, and reskinning. Finally, a skinned surface can exhibit unwanted self-intersections and twisting. However, in spite of these difficulties, skinning is a powerful and widely used surface design technique.

We turn our attention now to the v-directional interpolations. We assume for the moment that all section curves are nonrational; skinning across rational sections presents a host of problems, which we return to at the end of this section. Three items must be determined: the degree q, the parameters $\{\bar{v}_k\}$, and knots $V = \{v_i\}$. The degree q is arbitrary, the only restriction being $q \leq K$. The $\bar{v}_k$ are computed by averaging, for example

$$\bar{v}_0 = 0 \qquad \bar{v}_K = 1$$

$$\bar{v}_k = \bar{v}_{k-1} + \frac{1}{n+1} \sum_{i=0}^{n} \frac{|\mathbf{P}_{i,k} - \mathbf{P}_{i,k-1}|}{d_i} \qquad k = 1, \ldots, K-1 \tag{10.8}$$

where d_i denotes the total chord length of $\mathbf{P}_{i,0}, \ldots, \mathbf{P}_{i,K}$. The knots $\{v_i\}$ are then computed using Eq. (9.8). The skinned surface of Figure 10.3d was constructed using this method of interpolation.

Another common method is to use a so-called *spine curve* (also called a *path curve*), that is

$$\mathbf{C}_s(v) = \sum_{i=0}^{n_s} R_{i,p_s}(v)\,\mathbf{R}_i \tag{10.9}$$

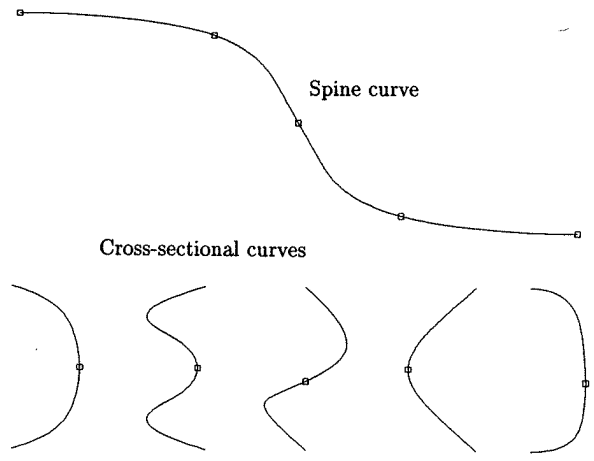


Figure 10.4. Spine curve and cross-sectional curves for spine-based skinning.

$\mathbf{C}_s(v)$ can be an arbitrary rational or nonrational, planar or nonplanar curve. The designer uses the spine curve to position and orient the section curves in space. Figure 10.4 shows the spine curve and five planar section curves to be positioned in three-dimensional space. One possible positioning is illustrated in Figure 10.5. The section curves are positioned at spine parameters

$$\hat{v}_k = \{0, 0.25, 0.5, 0.78, 1\}$$

and at section curve parameters

$$\hat{u}_k = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}$$

Each section curve is oriented so that its plane is perpendicular to the tangent vector of the spine curve at the intersection of the spine curve and the section's plane. The spine curve can also aid in v-directional shape control by providing additional constraints for use in the v-directional interpolations. For example, spine tangent directions can be used: Let $\{\hat{v}_k\}$, $k = 0, \ldots, K$, be the v values of the points on $\mathbf{C}_s(v)$ where the section curves are positioned; for planar sections these are the v values corresponding to intersections of the spine with the section planes. These values are generally known, and they can be input to the skinning routine. Notice that, in general, $\hat{v}_k \neq \bar{v}_k$ if the $\bar{v}_k$ are computed using Eq. (10.8).

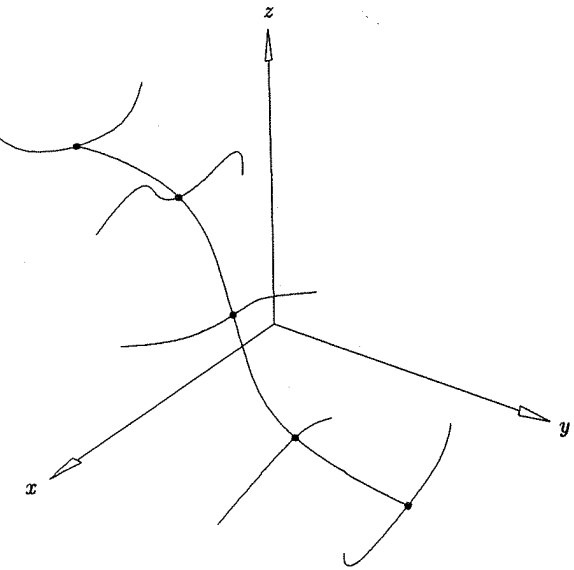


Figure 10.5. Section curves positioned along a spine curve.

Of course, one can set $\bar{v}_k = \hat{v}_k$, but this may not be a good idea if $\mathbf{C}_s(v)$ is poorly parameterized. Now let

$$\mathbf{P}_k = \mathbf{C}_s(\hat{v}_k) \qquad \mathbf{D}_k = \mathbf{C}_s'(\hat{v}_k) \qquad k = 0, \ldots, K$$

Then for each i and k we compute a derivative vector, $\mathbf{D}_{i,k}$. For each i this yields $2(K+1)$ interpolation constraints, $\mathbf{P}_{i,0}, \ldots, \mathbf{P}_{i,K}$ and $\mathbf{D}_{i,0}, \ldots, \mathbf{D}_{i,K}$. The $\mathbf{D}_{i,k}$ are computed by letting

$$d_k = |\mathbf{P}_k - \mathbf{P}_{k-1}| \qquad k = 1, \ldots, K$$

(see Figure 10.6). Then for $k = 1, \ldots, K-1$, set

$$\mathbf{D}_{i,k} = \frac{|\mathbf{P}_{i,k+1} - \mathbf{P}_{i,k}| + |\mathbf{P}_{i,k} - \mathbf{P}_{i,k-1}|}{d_{k+1} + d_k} \mathbf{D}_k \tag{10.10}$$

and at the ends

$$\mathbf{D}_{i,0} = \frac{|\mathbf{P}_{i,1} - \mathbf{P}_{i,0}|}{d_1} \mathbf{D}_0$$

$$\mathbf{D}_{i,K} = \frac{|\mathbf{P}_{i,K} - \mathbf{P}_{i,K-1}|}{d_K} \mathbf{D}_K \tag{10.11}$$

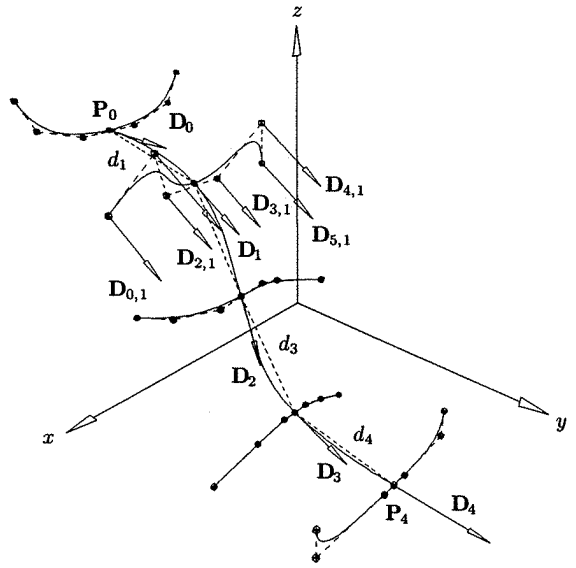


Figure 10.6. Computation of derivatives for spine skinning.

Hence, $\mathbf{D}_{i,k}$ is parallel to $\mathbf{D}_k$, but its magnitude is adjusted to reflect relative distances. Equations (10.10) and (10.11) ensure that

$$\mathbf{S}_v(u, \bar{v}_k) \parallel \mathbf{D}_k \qquad \text{for all } u \text{ and } k = 0, \ldots, K \tag{10.12}$$

that is, all partial derivatives with respect to v on the kth isoparametric curve are parallel to $\mathbf{D}_k$. This follows from the fact that these partial derivatives are linear combinations of the $\mathbf{D}_{i,k}$. Letting $m = 2K+1$, we can now set up $m+1$ linear equations for each fixed i (see Eqs. [9.19] and [9.20]), i.e.

$$\mathbf{P}_{i,k} = \sum_{j=0}^{m} N_{j,q}(\bar{v}_k)\mathbf{Q}_{i,j} \qquad \mathbf{D}_{i,k} = \sum_{j=0}^{m} N'_{j,q}(\bar{v}_k)\mathbf{Q}_{i,j} \tag{10.13}$$

The knot vector V must contain $m+q+2$ knots; Eqs. (9.22) and (9.23) give suitable choices for the cases $q=2$ and $q=3$, respectively.

Figure 10.7a shows the control points of the skinned surface defined by the data in Figure 10.5, using cubic interpolation with specified tangents as shown in Figure 10.6; the surface is depicted in Figure 10.7b. The same data is interpolated in Figures 10.8a and 10.8b, without specified derivatives. Note the significant difference in surface shapes (Figures 10.7b and 10.8b). The positioning of section curves has a definite effect on surface shape, just as the positioning of data points in curve and surface interpolation; Figures 10.9a and 10.9b show two examples. In Figure 10.9a the spine and section curve parameters are

$$\hat{v}_k = \{0, 0.35, 0.5, 0.62, 1\} \qquad \hat{u}_k = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}$$

and in Figure 10.9b they are

$$\hat{v}_k = \{0, 0.25, 0.5, 0.78, 1\} \qquad \hat{u}_k = \left\{\frac{1}{2}, \frac{3}{10}, \frac{1}{2}, \frac{9}{10}, \frac{1}{2}\right\}$$

We now consider the case in which some of the section curves are rational (Eq. [10.7]). The v interpolations must now be carried out in homogeneous space. Analogous to Eq. (10.8), the $\bar{v}_k$ are computed by averaging four-dimensional distances, that is

$$\bar{v}_0 = 0 \qquad \bar{v}_K = 1$$

$$\bar{v}_k = \bar{v}_{k-1} + \frac{1}{n+1} \sum_{i=0}^{n} \frac{|\mathbf{P}^w_{i,k} - \mathbf{P}^w_{i,k-1}|}{d^w_i} \qquad k = 1, \ldots, K-1 \tag{10.14}$$

where d^w_i denotes the total chord length of $\mathbf{P}^w_{i,0}, \ldots, \mathbf{P}^w_{i,K}$. The knots $\{v_i\}$ are again computed using Eq. (9.8). Interpolations through the $\mathbf{P}^w_{i,k}$ yield the control points $\mathbf{Q}^w_{i,j}$, $i = 0, \ldots, n$, $j = 0, \ldots, K$, of the skinned surface $\mathbf{S}^w(u,v)$. The isoparametric curves, $\mathbf{C}^w_{\bar{v}_k}(u) = \mathbf{S}^w(u, \bar{v}_k)$, are the original section curves, $\mathbf{C}^w_k(u)$. Although the designer may still use a spine curve to position and orient the

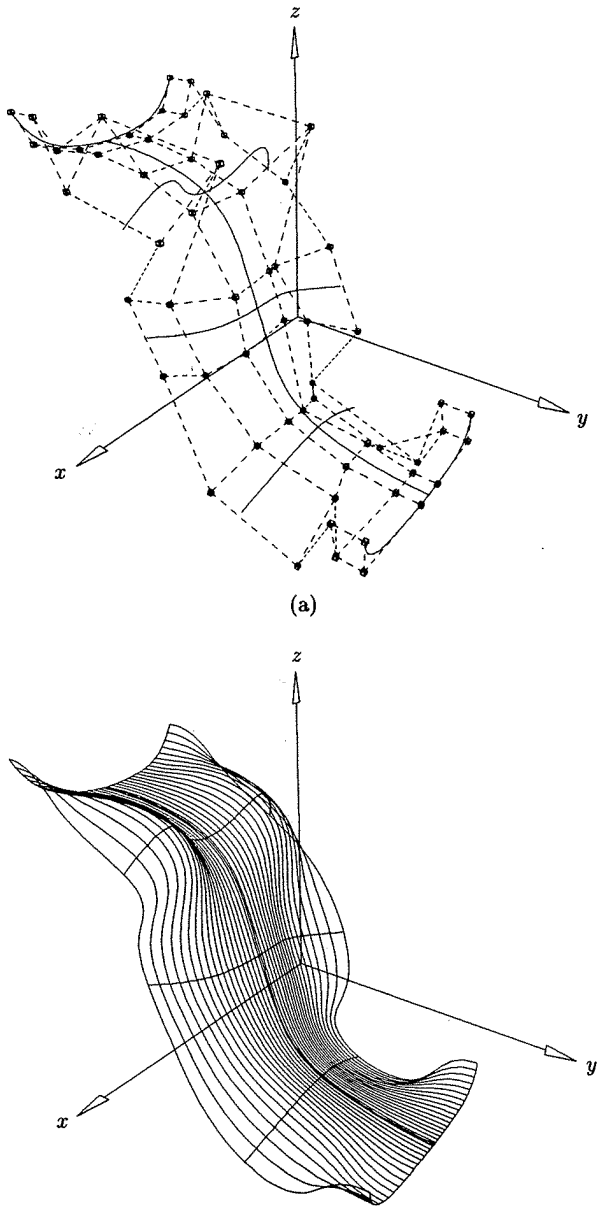


Figure 10.7. Spine skinning with spine controlled interpolation. (a) Control net; (b) skinned surface.

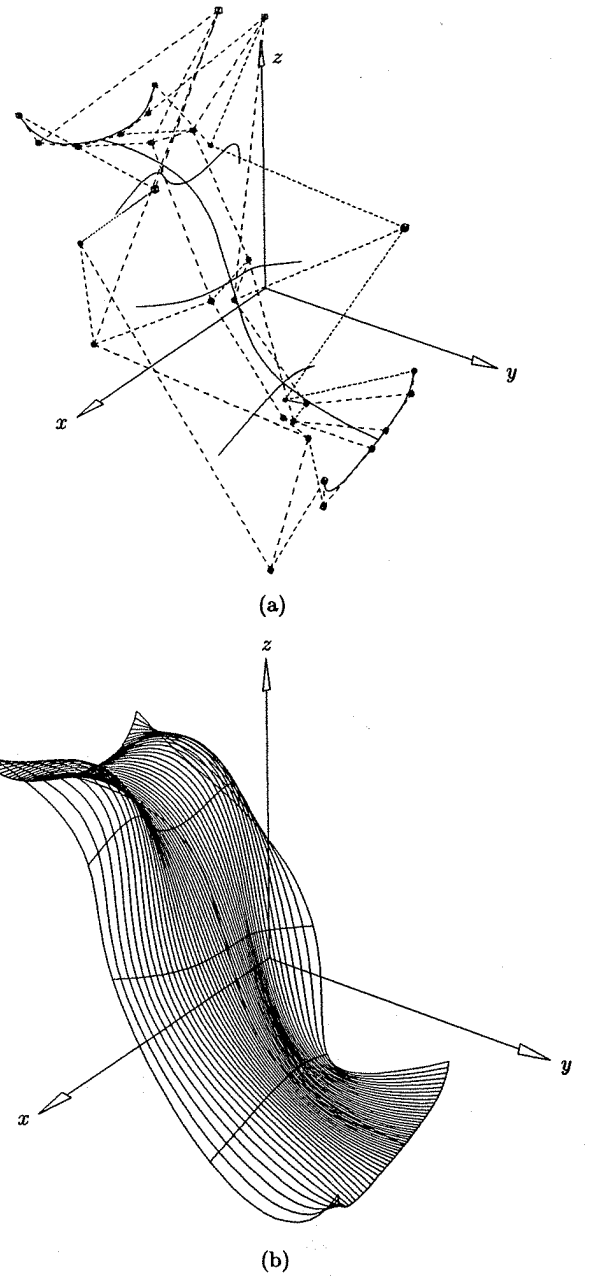
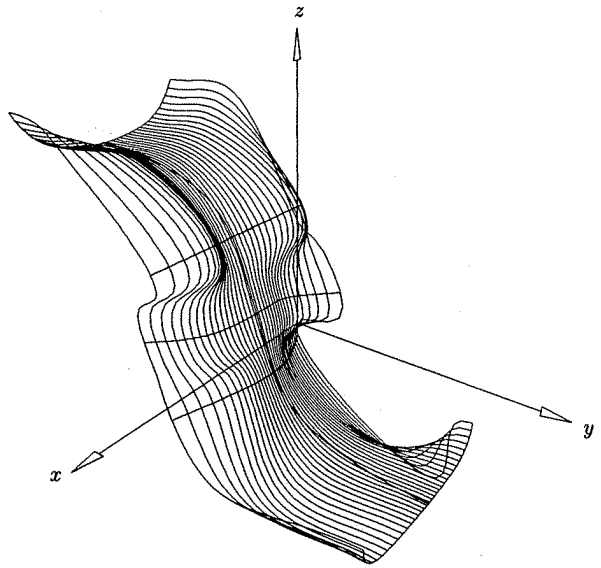
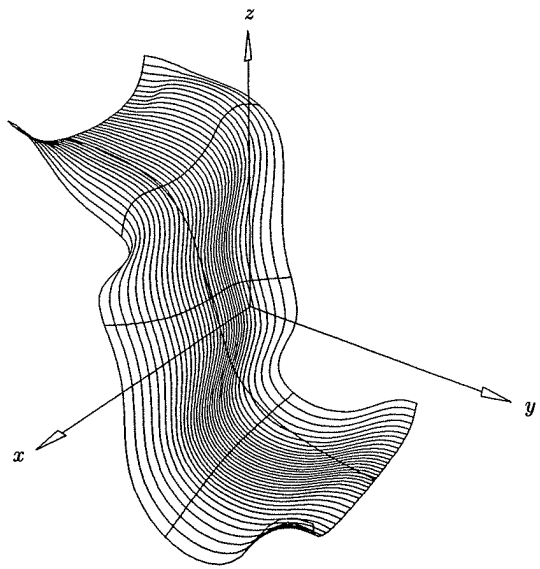


Figure 10.8. Spine skinning with no spine interpolation constraint. (a) Control net; (b) skinned surface.



(a)



(b)

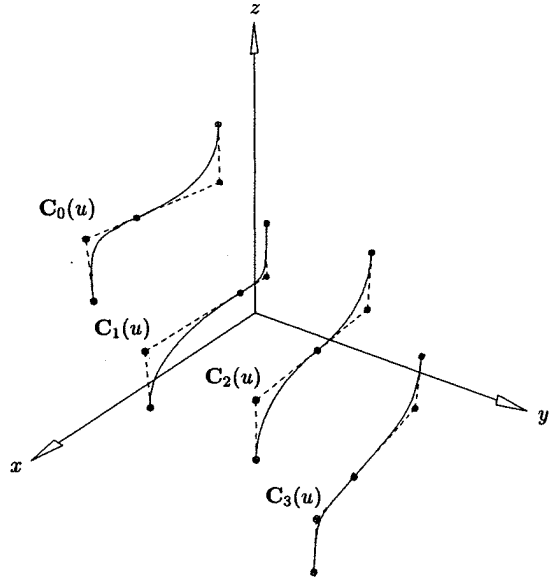
Figure 10.9. Spine skinning. (a) Section curves are positioned at different spine parameters; (b) section curves are positioned at different section curve parameters.

section curves, there is no obvious way to impose (linear) derivative constraints for the v direction interpolations to ensure that Eq. (10.12) is satisfied for all u and k.

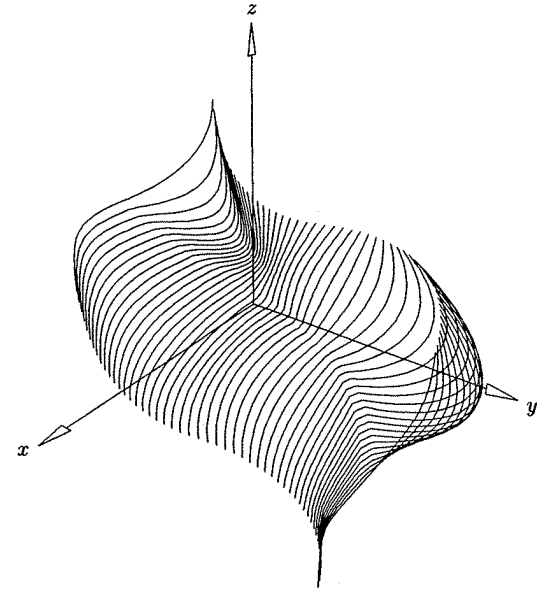
Most often, skinning through rational section curves yields satisfactory results, but the reader should be aware that there exist several potential hazards, some of which do not seem to have very satisfying practical solutions. These problems all arise as a result of the fact that surface construction is taking place in homogeneous space, and corresponding three-dimensional and four-dimensional geometry can be radically different. For example, two three-dimensional curves can be smooth, similar in shape, and close to one another, while their four-dimensional counterparts may have cusps, have totally different shapes, and be far from one another. More specifically, some problems that can arise are:

- the v-directional surface parameterization may be rather poor. As a simple example, take two section curves, and multiply all the control points of one of them by a large weight. This does not change the three-dimensional curve geometry. The simple ruled (skinned) surface between the two sections is poorly parameterized in the v direction. The three-dimensional surface parameterization can be improved by using the projected control points, $\mathbf{P}_{i,k}$, to compute the $\bar{v}_k$ (as in Eq. [10.8]), but this can produce wild four-dimensional v-directional interpolations, resulting in undesirable geometry both in four-dimensional and three-dimensional space;
- for fixed i, the surface weights $w_{i,0}, \ldots, w_{i,K}$ result from interpolating the weights of the section control points, $\mathbf{P}^w_{i,0}, \ldots, \mathbf{P}^w_{i,K}$. Clearly, the interpolation can produce a negative or zero $w_{i,k}$, even though all the section weights are positive (e.g., see the explanation to Figure 10.10b below). This can cause problems, for example crash a system, downstream in a system not designed to handle such weights;
- the surface may not be as smooth as the section curves indicate. For example, let one section curve be the full circle given in Example `Ex7.2`. As shown in that example, the circle is C^1 continuous, although its homogeneous counterpart is only C^0 continuous at the double knots. Assuming that all other section curves are C^1, one expects the resulting skinned surface, $\mathbf{S}(u,v)$, to be C^1 continuous in the u direction. But $\mathbf{S}^w(u,v)$ is only C^0 continuous in the u direction, since it is formed by blending C^0 curves; in general, arbitrary u-directional isoparametric curves on $\mathbf{S}^w(u,v)$ are only C^0, and their projections to three-dimensional space may be only C^0 continuous. Figures 10.10a and 10.10b show an example of this. The four rational section curves are all C^1 continuous in three-dimensional space. They are quadratics, defined with a double knot at $u = 1/2$. The respective weights of the section curves are

$$\begin{aligned}
\mathbf{C}_0 &: \{1,2,1,1,1\} \\
\mathbf{C}_1 &: \{1,1,1,3,1\} \\
\mathbf{C}_2 &: \{1,1,1,1,1\} \\
\mathbf{C}_3 &: \{1,4,1,2,1\}
\end{aligned}$$



(a)



(b)

Figure 10.10. Rational skinning. (a) C^1 continuous section curves; (b) skinned surface.

$\mathbf{C}_0$, $\mathbf{C}_1$, and $\mathbf{C}_3$ are only C^0 continuous in four-dimensional space. After v-directional interpolations, the surface weights are:

$$w_{i,j} = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \\ 1 & 0.94 & 1 & 4.43 & 1 \\ 1 & 0.71 & 1 & -0.75 & 1 \\ 1 & 4 & 1 & 2 & 1 \end{bmatrix}$$

(note the negative weight at $\mathbf{P}_{3,2}$). The surface shown in Figure 10.10b exhibits only C^0 continuity in the u direction. To the authors' knowledge, the only solutions to this dilemma involve the use of higher degree section curves. For example, a full circle can be represented as a degree 5 NURBS curve with positive weights and no interior knots (Example E7.10), thereby eliminating the problem. Several solutions to this problem, all effectively raising the degree of the skinned surface, are discussed in [Hohm91]. A general solution is to use so-called *smoothing functions*. Suppose $\mathbf{C}_k^w(u)$ is a C^r continuous section curve on the knot vector U, whose projection $\mathbf{C}_k(u)$ is C^{r+s} continuous, $s > 0$. Let $f(u)$ be a B-spline scalar-valued function on U, such that

$$\hat{\mathbf{C}}_k^w(u) = f(u)\mathbf{C}_k^w(u) \tag{10.15}$$

is C^{r+s} continuous. $f(u)$ is called a smoothing function. For fixed u_0, $f(u_0)$ simply scales $\mathbf{C}_k^w(u_0)$ up or down the four-dimensional cone defined by $\mathbf{C}_k(u)$, that is, $\hat{\mathbf{C}}_k(u) = \mathbf{C}_k(u)$ for all u. Clearly, if all $\mathbf{C}_k^w(u)$ can be smoothed, then skinning through the corresponding $\hat{\mathbf{C}}_k^w(u)$ yields a C^{r+s} continuous surface which interpolates the three-dimensional section curves. However, we know of no general algorithm for determining $f(u)$. Differentiating Eq. (10.15) from the left and right yields a set of linear equations which express the desired continuity constraints at the knots. However, other considerations are:

- if $\mathbf{C}_k^w(u)$ is closed, then $\hat{\mathbf{C}}_k^w(u)$ should be closed;
- all weights of $\hat{\mathbf{C}}_k^w(u)$ should be positive;
- the weights of $\hat{\mathbf{C}}_k^w(u)$ should not vary wildly or become exceedingly large.

A degree $r + s$ function, $f(u)$, suffices to satisfy just the continuity constraints at the knots, but the other constraints generally force $f(u)$ to be of higher degree. Finally, once $f(u)$ is determined it must be multiplied with $\mathbf{C}_k^w(u)$. Multiplication of B-splines is not a trivial task. An algorithm similar to degree elevation (Algorithm A5.9) of Chapter 5 can be developed, i.e.,

1. decomposition of $f(u)$ and $\mathbf{C}_k^w(u)$ into Bézier segments;
2. multiplication of the segments;
3. recomposition of $\hat{\mathbf{C}}_k^w(u)$ into a NURBS curve (knot removal).

10.4 Swept Surfaces

Next we address the topic of sweeping a section curve along an arbitrary trajectory curve. Denote the trajectory by $\mathbf{T}(v)$ and the section curve by $\mathbf{C}(u)$. A general form of the swept surface is given by

$$\mathbf{S}(u,v) = \mathbf{T}(v) + M(v)\mathbf{C}(u) \tag{10.16}$$

where $M(v)$ is a 3×3 matrix incorporating rotation and nonuniform scaling of $\mathbf{C}(u)$ as a function of v. $\mathbf{T}(v)$ and $\mathbf{C}(u)$ may be arbitrary: nonrational or rational, planar or nonplanar, open or closed.

In general, Eq. (10.16) can produce nonsensical and unwanted surfaces with self-intersections, degeneracies, and discontinuities. Furthermore, in many cases $\mathbf{S}(u,v)$ is not precisely representable as a NURBS surface. In practice, most swept surfaces are one of two specific types:

1. $M(v)$ is the identity matrix for all v, that is, for each v, $\mathbf{C}(u)$ is just translated by $\mathbf{T}(v)$;
2. $M(v)$ is not the identity matrix.

Case 1 is described by the equation

$$\mathbf{S}(u,v) = \mathbf{T}(v) + \mathbf{C}(u) \tag{10.17}$$

which has a precise NURBS representation. We call this surface a *generalized translational sweep*. The general cylinder of Section 8.3 is a special case, with $\mathbf{T}(v)$ being a straight line. $\mathbf{S}(u,v)$ is constructed as follows. Let

$$\mathbf{T}(v) = \frac{\sum_{j=0}^{m} N_{j,q}(v) w_j^T \mathbf{T}_j}{\sum_{j=0}^{m} N_{j,q}(v) w_j^T} \qquad V = \{v_0, \ldots, v_s\}$$

and

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i^C \mathbf{Q}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i^C} \qquad U = \{u_0, \ldots, u_r\}$$

Then

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

is defined on knots U and V, and has control points

$$\mathbf{P}_{i,j} = \mathbf{T}_j + \mathbf{Q}_i \qquad i = 0, \ldots, n \quad j = 0, \ldots, m \tag{10.18}$$

and weights

$$w_{i,j} = w_i^C w_j^T \qquad i = 0, \ldots, n \quad j = 0, \ldots, m \tag{10.19}$$

Figures 10.11a and 10.11b show an example of translational sweep.

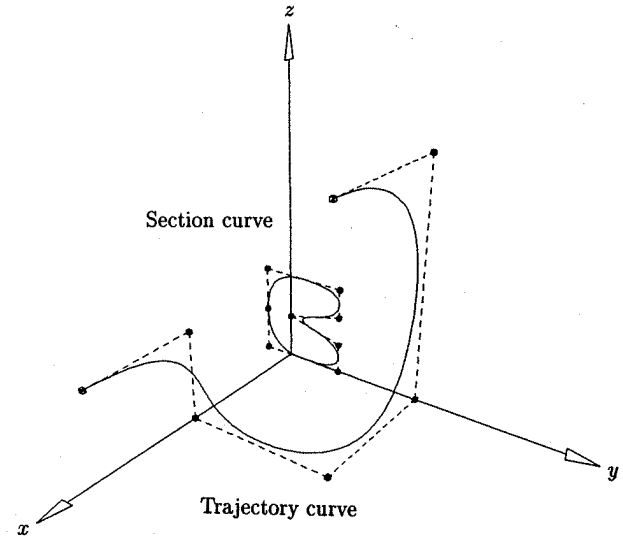
Now consider Case 2. Let $\{\mathbf{O}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ denote our global coordinate system. To simplify matters, we introduce a local orthonormal coordinate system $\{\mathbf{o}(v), \mathbf{x}(v), \mathbf{y}(v), \mathbf{z}(v)\}$, which moves along $\mathbf{T}(v)$. For each v, $v_0 \le v \le v_s$, set

$$\mathbf{o}(v) = \mathbf{T}(v) \qquad \mathbf{x}(v) = \frac{\mathbf{T}'(v)}{|\mathbf{T}'(v)|} \tag{10.20}$$

Additionally, let $\mathbf{B}(v)$ be a vector-valued function satisfying $\mathbf{B}(v) \cdot \mathbf{x}(v) = 0$ for all v, and set

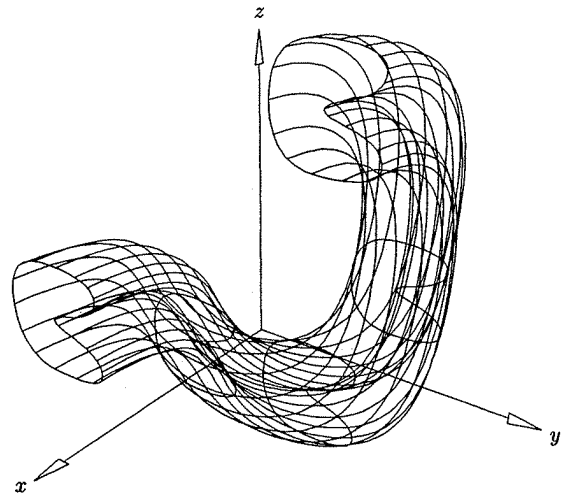
$$\mathbf{z}(v) = \frac{\mathbf{B}(v)}{|\mathbf{B}(v)|} \tag{10.21}$$

The determination of this function is a critical and nontrivial part of the sweeping



(a)

Figure 10.11. Translational sweeping. (a) Trajectory and section curves; (b) swept surface.



(b)

Figure 10.11. (*Continued.*)

process. We return to the question of how to determine $\mathbf{B}(v)$ at the end of this section; for now, we simply assume it is given. $\mathbf{y}(v)$ is then defined by

$$\mathbf{y}(v) = \mathbf{z}(v) \times \mathbf{x}(v) \tag{10.22}$$

In order to incorporate scaling we introduce a three-dimensional vector function

$$\mathbf{s}(v) = \big(s_x(v), s_y(v), s_z(v)\big) \tag{10.23}$$

The swept surface is now defined by

$$\mathbf{S}(u,v) = \mathbf{T}(v) + A(v)S(v)\mathbf{C}(u) \tag{10.24}$$

where $S(v)$ is a 3×3 diagonal matrix with diagonal elements $\big(s_x(v), s_y(v), s_z(v)\big)$, and $A(v)$ is the general transformation matrix taking $\{\mathbf{O},\mathbf{X},\mathbf{Y},\mathbf{Z}\}$ into $\{\mathbf{o}(v), \mathbf{x}(v), \mathbf{y}(v), \mathbf{z}(v)\}$. The isoparametric curves on $\mathbf{S}(u,v)$ at fixed v values are instances of $\mathbf{C}(u)$, transformed by $A(v)S(v)$ and translated by $\mathbf{T}(v)$. If $\mathbf{C}(u)$ passes through the global origin at $u = \bar{u}$, then $\mathbf{T}(v)$ is the isoparametric curve $\mathbf{S}(u = \bar{u}, v)$.

$\mathbf{S}(u,v)$ may not be representable in NURBS form. Clearly, if $A(v)$ and $S(v)$ can be precisely represented as rational B-spline functions, then so can $\mathbf{S}(u,v)$. $S(v)$ is generally no problem. If scaling is applied at all, then it is common to use a linear function or a higher degree B-spline approximation to a function which is linear with respect to the arc length parameter. However, $A(v)$ is generally

not representable in NURBS form, even in the simple case that $\mathbf{z}(v) = \mathbf{Z}$ for all $v_0 \le v \le v_s$. Hence, we must construct a NURBS approximation to Eq. (10.24). Denote this approximation by $\hat{\mathbf{S}}(u,v)$. We present three methods:

- The first method is to use Eq. (10.24) to evaluate an $(n \times m)$ grid of points lying precisely on $\mathbf{S}(u,v)$, and then use either the interpolation or approximation techniques of Chapter 9 to construct a nonrational B-spline approximation, $\hat{\mathbf{S}}(u,v)$. A disadvantage with this method is that none of the u or v isoparametric curves on $\hat{\mathbf{S}}(u,v)$ are precise;
- Another method is given by Bloomenthal and Riesenfeld [Blom91] and by Coquillart [Coqu87a]; their method involves approximating the three-dimensional offset curves of the trajectory. Their algorithm interpolates $\mathbf{T}(v)$ as an isoparametric curve if $\mathbf{C}(u)$ passes through the origin; however, instances of $\mathbf{C}(u)$ are not interpolated, except at $v = v_0$ and $v = v_s$. Bloomenthal and Riesenfeld give error bounds for their method, based on error bounds for the offsetting process;
- A third method, which we present in more detail here, is to use skinning. We transform and place $K+1$ instances of $\mathbf{C}(u)$ along $\mathbf{T}(v)$, and then skin across these section curves. The accuracy of the resulting surface, $\hat{\mathbf{S}}(u,v)$, is increased by increasing K. Each instance of $\mathbf{C}(u)$ is interpolated as an isoparametric curve on $\hat{\mathbf{S}}(u,v)$. There are two strategies for the v-directional fitting. The first makes use of the B-spline structure of $\mathbf{T}(v)$. The trajectory's degree q and knot vector V are used for the fitting, hence they are inherited by $\hat{\mathbf{S}}(u,v)$. Furthermore, the parameters $\bar{v}_k$ for the fitting must be taken from $\mathbf{T}(v)$; if $\mathbf{T}(v)$ is rational, its weights must be factored in prior to the v-directional fitting. This method is required if $\mathbf{C}(u)$ passes through the global origin and it is desired that $\mathbf{T}(v)$ be an isoparametric curve in $\hat{\mathbf{S}}(u,v)$, but the method can be used in any case. The other method does not make use of $\mathbf{T}(v)$ other than to position the section curve; that is, the v-degree of $\hat{\mathbf{S}}(u,v)$ is input, parameters and knots are computed in some manner, and $\mathbf{T}(v)$'s weights need not be factored in.

Corresponding to the two methods of v-directional fitting, two sweeping algorithms follow. In both cases, the swept surface, $\hat{\mathbf{S}}(u,v)$, inherits its u-degree p and the knot vector U from $\mathbf{C}(u)$. `T` and `C` are the trajectory and section curves, and `Bv` and `sv` are input functions given by Eqs. (10.21) and (10.23), respectively. In Algorithm `A10.1` the v-degree `q` is not required; the number `K` is to be viewed as a minimum, because more instances of the section curve may be required depending on the number of knots defining $\mathbf{T}(v)$.

```
ALGORITHM A10.1
  SweepSurface1(T,C,Bv,sv,K,V,Pw)
    { /*  Swept surface.  Trajectory interpolated.  */
      /*  Input:  T,C,Bv,sv,K  */
      /*  Output: V,Pw  */
    q = degree of T(v).
```

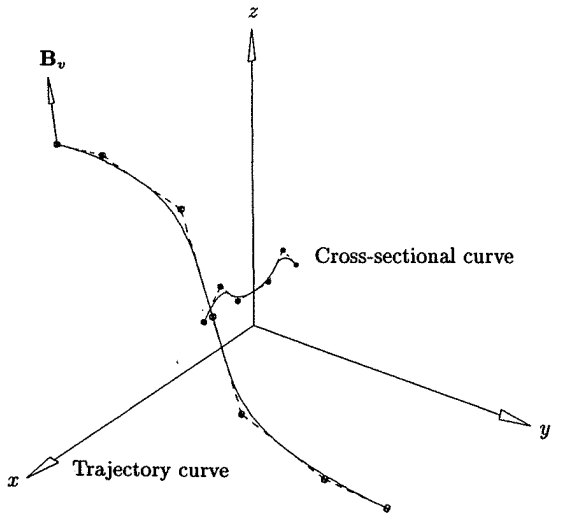
```
ktv = number of knots of T(v).
nsect = K+1;
if (ktv <= nsect+q)
  {  /* Must refine T(v)'s knot vector */
  m = nsect+q-ktv+1;
  Insert m more knots into T(v)'s knot vector.  Locations
    are not critical, recursively insert at the midpoint of
    the longest span will do.  New control points do not
    have to be computed, as we only require a refined
    set of knots.
  The resulting knot vector V is inherited by Ŝ(u,v).
  }
  else
  {  /* T(v)'s knot vector will do */
  V = T(v)'s knot vector.
  if (ktv > nsect+q+1)    /* Must increase number of */
    nsect = ktv-q-1;      /* instances of C(u) */
  }
v̄_0 = T(v)'s minimum parameter value.
v̄_nsect-1 = T(v)'s maximum parameter value.
for (k=1; k<nsect-1; k++)    /* Compute parameters by */
  v̄_k = (v_k+1 + ··· + v_k+q)/q;    /* averaging knots */
for (k=0; k<nsect; k++)
  {  /* Transform and position section control points  */
  Let Q_i and w_i be the control points and weights
    of C(u), i = 0,...,n.
  Scale the control points Q_i by sv.
  Compute {o,x,y,z}(v̄_k) from Eqs.(10.20)-(10.22).
  Compute the transformation matrix A(v̄_k) transforming the
    global system into {o,x,y,z}(v̄_k).
  Apply A(v̄_k) to the scaled Q_i.
  Reapply the weights w_i, and denote the resulting
    weighted control points by Q^w_k,i.
  if (T(v) is rational)
    {  /* w(v) is the weight function of T(v) */
       /* (4th coordinate) */
    Q^w_k,i = w(v̄_k) Q^w_k,i;
    }
  }
for (i=0; i<=n; i++)
  {
  Interpolate across Q^w_0,i,..., Q^w_nsect-1,i to obtain
    P^w_0,i,..., P^w_nsect-1,i.
  }
}
```



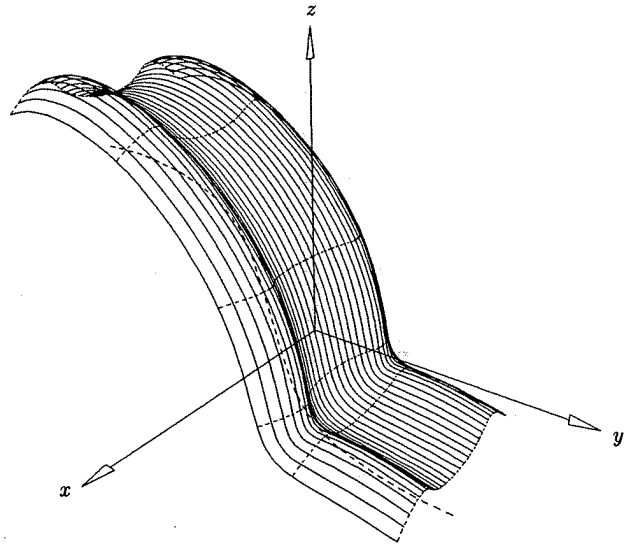
```
ALGORITHM A10.2
  SweepSurface2(T,C,Bv,sv,q,K,V,Pw)
    {  /*  Swept surface.  Trajectory not interpolated.  */
       /*  Input:  T,C,Bv,sv,q,K  */
       /*  Output: V,Pw  */
    Determine values v_0,...,v_K at which to place the
      instances of C(u).  A reasonable choice is to select the
      v_k so that the T(v_k) are approximately evenly
      spaced.
    for (k=0; k<=K; k++)
      {  /* Transform and position section control points */
      Let Q_i and w_i be the control points and weights
        of C(u), i = 0,...,n.
      Scale the control points Q_i by sv.
      Compute {o,x,y,z}(v_k) from Eqs.(10.20)-(10.22).
      Compute the transformation matrix A(v_k) transforming the
        global system into {o,x,y,z}(v_k).
      Apply A(v_k) to the scaled Q_i.
      Reapply the weights w_i, and denote the resulting
        weighted control points by Q^w_k,i.
      }
    Determine v̄_0,...,v̄_K, the v-parameters for the
      v-directional interpolations (Eqs.[10.8] or [10.14]).
    Given q and the v̄_k, compute the knot vector V by averaging
      (Eq.[9.8]).
    for (i=0; i<=n; i++)
      {
      Interpolate across Q^w_0,i,..., Q^w_K,i to obtain P^w_0,i,..., P^w_K,i.
      }
    }
```

Figure 10.12a shows the trajectory and section curves, along with a start vector, $\mathbf{B}(v_0)$. The section curve does not pass through the global origin. Figures 10.12b and 10.12c show the results of sweeping by Algorithms A10.2 and A10.1, respectively. Since the section curve does not pass through the origin the trajectory is not contained in the surface, even in Figure 10.12c. Also note the larger number of section curve placements in Figure 10.12c. Figures 10.13a–10.13c show sweeping examples where the section curve passes through the origin (Figure 10.13a). No trajectory control was used (Algorithm A10.2). Figure 10.13b used no scaling, while Figure 10.13c used a piecewise linear B-spline scaling function defined by control points

$$\left\{\left(\frac{3}{10},\frac{3}{10},1\right),\left(\frac{3}{2},\frac{3}{2},\frac{1}{2}\right),\left(\frac{3}{10},\frac{3}{10},1\right),(1,1,2)\right\}$$

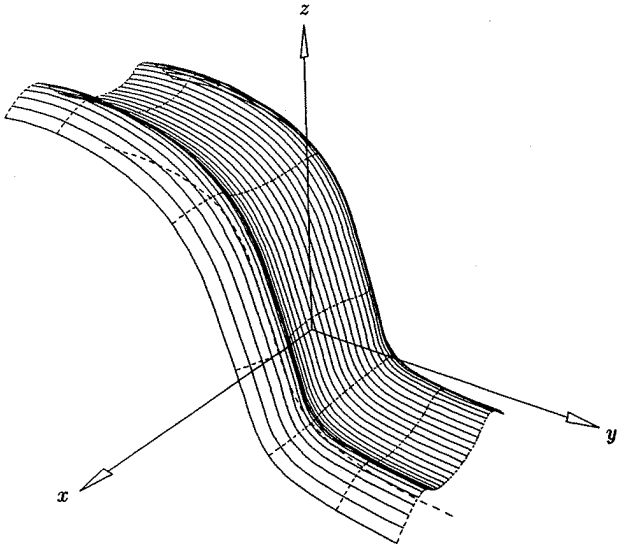


(a)



(b)

Figure 10.12. Sweeping examples. (a) Two-dimensional trajectory and the section curve not passing through the global origin; (b) swept surface with no trajectory controlled interpolation; (c) swept surface obtained by trajectory controlled interpolation.



(c)

Figure 10.12. (*Continued.*)

and knots

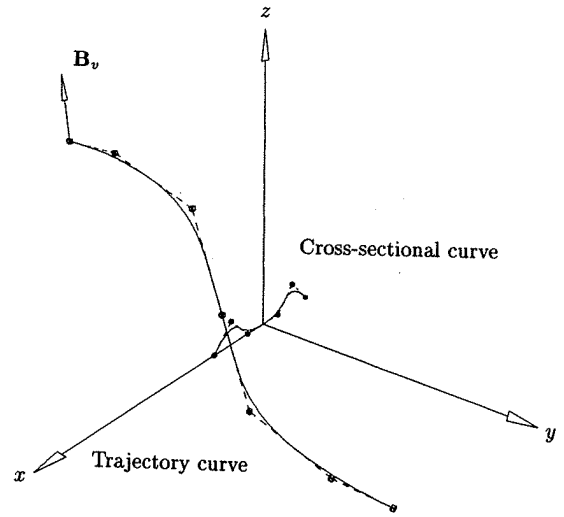
$$V = \left\{0, 0, \frac{3}{10}, \frac{7}{10}, 1, 1\right\}$$

The previous two examples used a planar trajectory curve. Figure 10.14a shows a three-dimensional (twisted) trajectory. The trajectory was not maintained in Figure 10.14b (Algorithm `A10.2`), whereas the trajectory is an isocurve in the surface of Figure 10.14c (Algorithm `A10.1`). In both examples all section instances were scaled with the constant vector $(3/5, 3/5, 1)$.

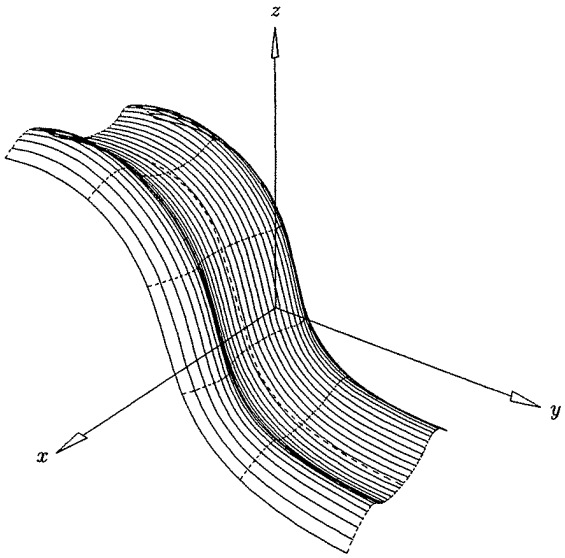
The positioning and size of the section curve, as well as the shape (curvature) of the trajectory, have a dramatic effect on surface shape. Two examples are shown in Figures 10.15a and 10.15b. The trajectory is not interpolated in Figure 10.15a, but it is in Figure 10.15b. Both surfaces exhibit the so-called *offsetting phenomenon*, i.e., they have self-intersections and unwanted oscillations.

We return now to the problem of determining a suitable orientation function, $\mathbf{B}(v)$. If $\mathbf{T}(v)$ is a planar curve, the solution is simple: Let $\mathbf{B}(v)$ be a constant function whose value is a vector normal to the plane of $\mathbf{T}(v)$. For arbitrary trajectories the situation is more difficult. A common solution is to use the so-called *Frenet frame* [DoCa76; Klok86; Bron92; Silt92]. Assume $\mathbf{T}(v)$ is twice differentiable. Define $\mathbf{B}(v)$ by

$$\mathbf{B}(v) = \frac{\mathbf{T}'(v) \times \mathbf{T}''(v)}{|\mathbf{T}'(v) \times \mathbf{T}''(v)|} \tag{10.25}$$

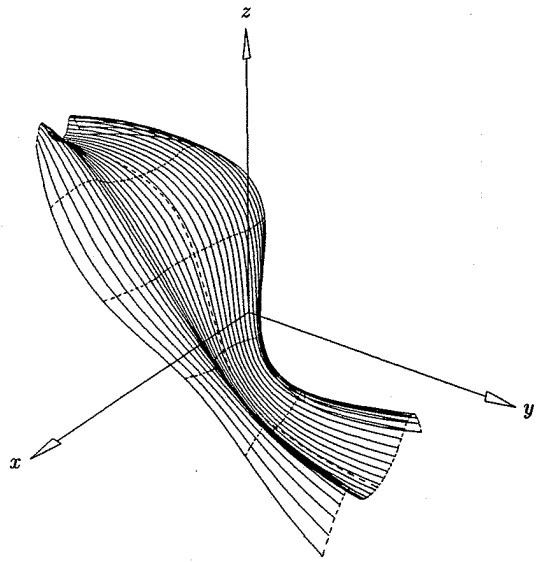


(a)



(b)

Figure 10.13. Sweeping examples. (a) Two-dimensional trajectory and section curve passing through the global origin; (b) swept surface with no scaling; (c) swept surface with scaled section curves.



(c)

Figure 10.13. (*Continued.*)

and $\mathbf{N}(v)$ by

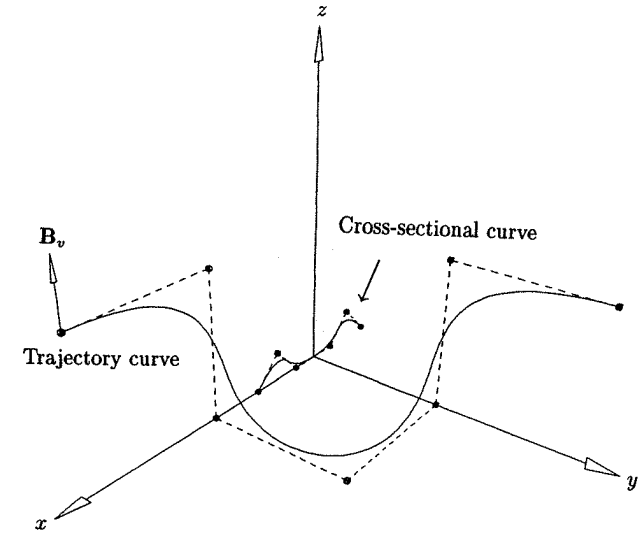
$$\mathbf{N}(v) = \mathbf{B}(v) \times \mathbf{T}'(v) \tag{10.26}$$

$\mathbf{T}'(v), \mathbf{N}(v)$, and $\mathbf{B}(v)$ are mutually orthogonal for all v, and when normalized they form a moving local coordinate system on $\mathbf{T}(v)$ called the Frenet frame. $\mathbf{B}(v)$ (or a NURBS approximation of it) can then be used in Eq. (10.21) and Algorithms `A10.1` and `A10.2`. There are several disadvantages to this method:

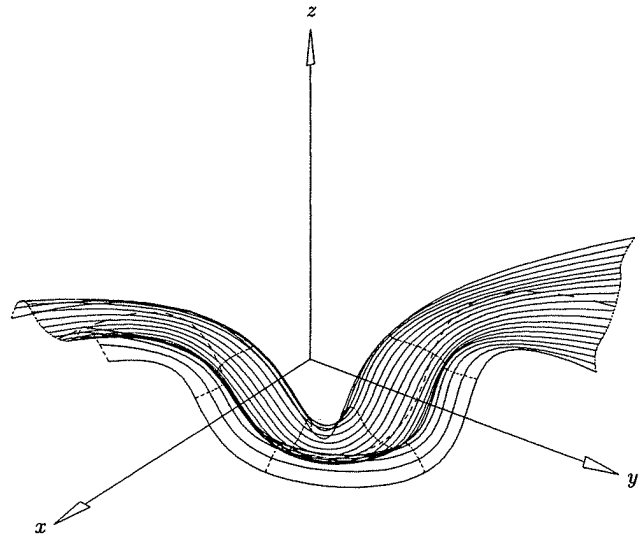
- $\mathbf{B}(v)$ is not defined by Eqs. (10.25) and (10.26) for linear segments or at inflection points (or more generally, where $\mathbf{T}''(v) \times \mathbf{T}'(v) = 0$);
- $\mathbf{B}(v)$ flips abruptly to the opposite direction at an inflection point;
- for three-dimensional trajectories, the vectors given by $\mathbf{B}(v)$ can rotate excessively around $\mathbf{T}(v)$, causing unwanted twisting of the resulting swept surface. Figure 10.16 shows this phenomenon; notice how the Frenet frame flips at the inflection point.

The first two problems can be dealt with rather easily [Klok86; Bron92; Silt92], but the rotation of $\mathbf{B}(v)$ depends solely on the shape of $\mathbf{T}(v)$, hence it cannot be avoided without changing the shape of $\mathbf{T}(v)$.

Siltanen and Woodward [Silt92] give an elegant method, called the *projection normal method*, that avoids these three problems. Let $v_0, \ldots, v_m$ be an increasing sequence of parameter values, with v_0 and v_m being the minimum and maximum values, respectively, in the domain of $\mathbf{T}(v)$. At each v_i we compute a vector, $\mathbf{B}_i$,

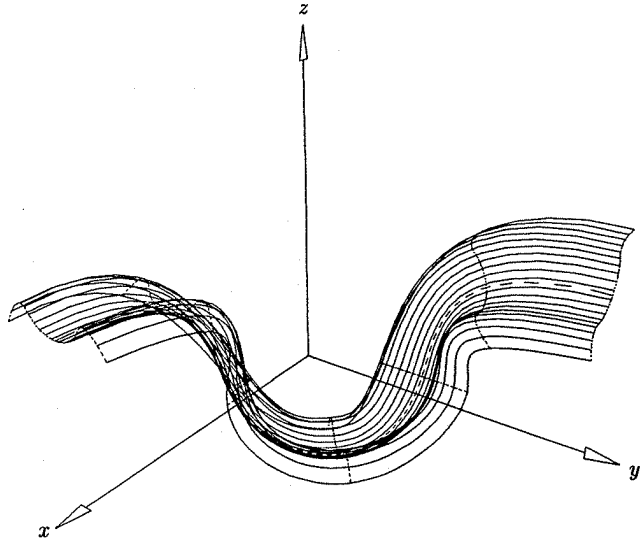


(a)



(b)

Figure 10.14. Sweeping examples with scaled section curves. (a) Three-dimensional trajectory and section curve; (b) trajectory is not interpolated; (c) trajectory is interpolated.



(c)

Figure 10.14. (*Continued.*)

and then use either interpolation or approximation to obtain the function $\mathbf{B}(v)$. We use Siltanen and Woodward's method to compute the $\mathbf{B}_i$. Set $\mathbf{B}_0$ to an arbitrary unit length vector orthogonal to $\mathbf{T}'(v_0)$. Then for $i = 1, \ldots, m$, compute

$$\mathbf{T}_i = \frac{\mathbf{T}'(v_i)}{|\mathbf{T}'(v_i)|}$$

and set

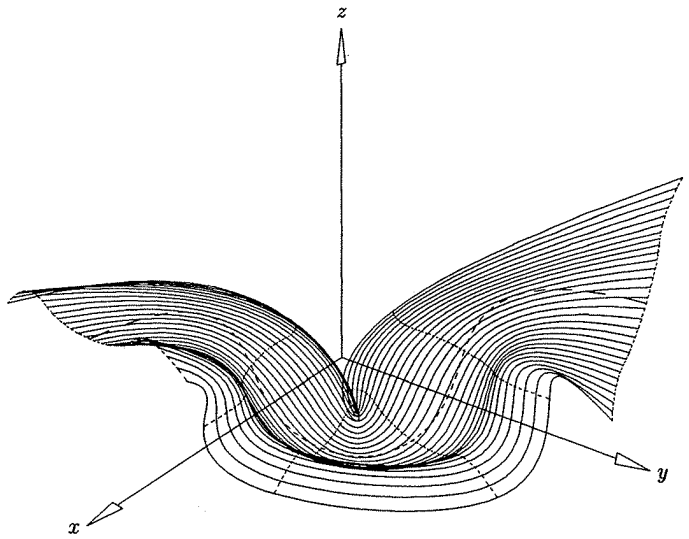
$$\mathbf{b}_i = \mathbf{B}_{i-1} - (\mathbf{B}_{i-1} \cdot \mathbf{T}_i)\mathbf{T}_i$$

$$\mathbf{B}_i = \frac{\mathbf{b}_i}{|\mathbf{b}_i|} \tag{10.27}$$

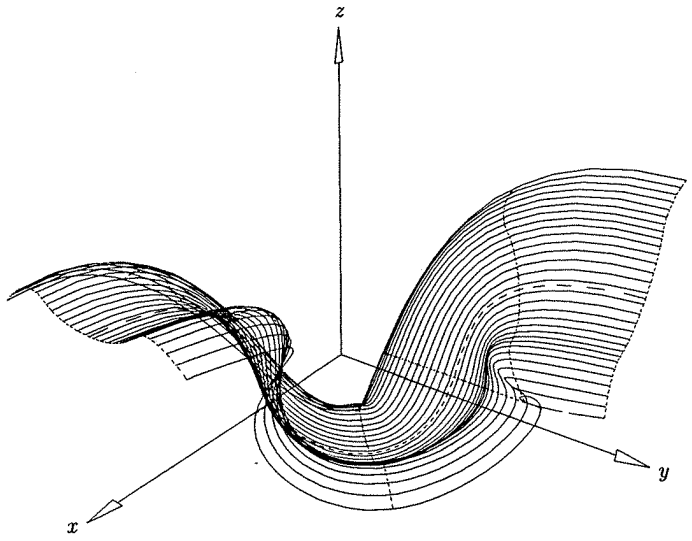
In words, $\mathbf{B}_i$ is obtained by projecting $\mathbf{B}_{i-1}$ onto the plane defined by $\mathbf{T}_i$. Care must be taken to avoid points where $\mathbf{T}_i \parallel \mathbf{B}_{i-1}$. Notice that if $\mathbf{T}(v)$ is a closed curve, then in general the computed $\mathbf{B}_m$ is not equal to $\mathbf{B}_0$ (which is required). The solution is to set $\mathbf{B}_m = \bar{\mathbf{B}}_m = \mathbf{B}_0$, use the reverse of the previous technique to compute $\bar{\mathbf{B}}_{m-1}, \ldots, \bar{\mathbf{B}}_1$, and then set

$$\mathbf{B}_i = \frac{1}{2}(\mathbf{B}_i + \bar{\mathbf{B}}_i) \qquad i = 1, \ldots, m-1$$

The swept surfaces of Figures 10.12 to 10.15 used the projection normal method to compute the $\mathbf{B}_i$ vectors. Figures 10.17a and 10.17b show examples of how the



(a)



(b)

Figure 10.15. Sweeping examples. (a) Trajectory is not interpolated; (b) trajectory is interpolated.

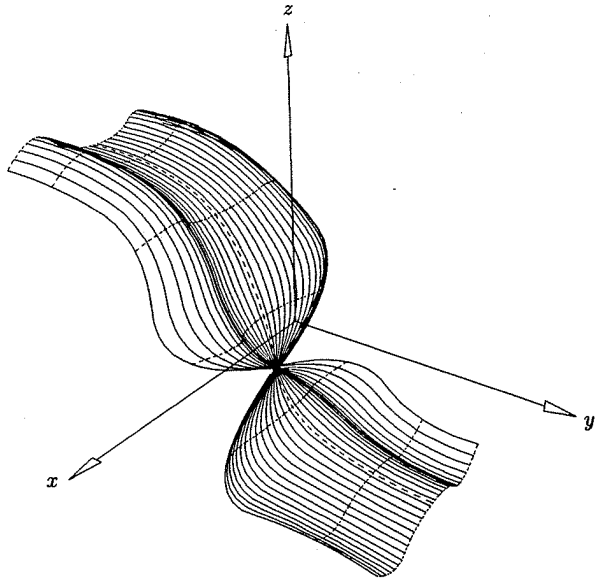


Figure 10.16. Sweeping example; section curve is placed by the Frenet frame.

$\mathbf{B}(v)$ vector (projection normal method) changes in comparison to the Frenet frame. In Figure 10.17a the trajectory is planar, whereas in Figure 10.17b it is twisted. In both cases the moving $\mathbf{B}(v)$ is drawn with a solid arrow, and the Frenet frame axes are plotted dashed.

Finally, Klok [Klok86] and Guggenheimer [Gugg89] present another method of obtaining a rotation minimizing $\mathbf{B}(v)$ as the solution to a differential equation with an initial condition. Additional reading on swept surfaces can be found in [Choi90; Akma92].

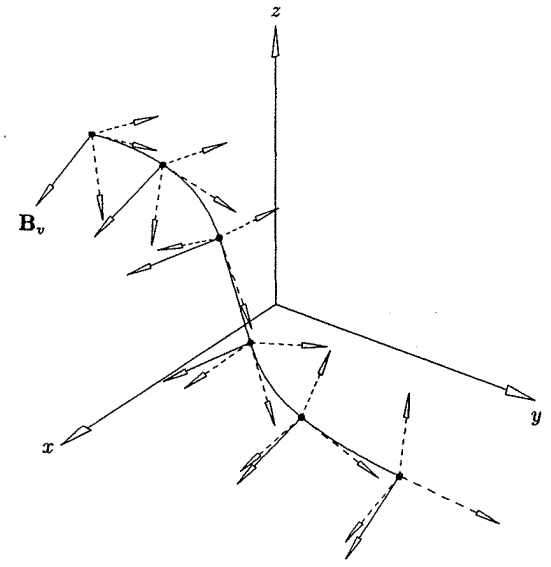
10.5 Interpolation of a Bidirectional Curve Network

Let

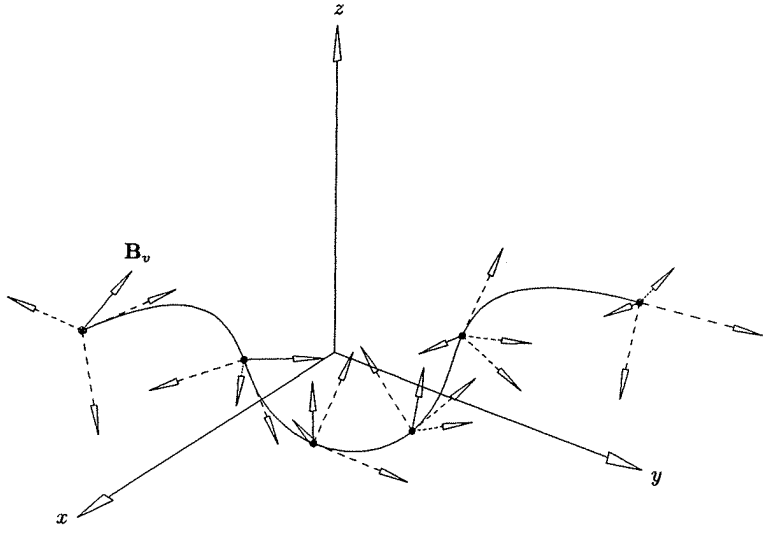
$$\mathbf{C}_k(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_{k,i} \qquad k = 0, \ldots, r \quad u \in [0,1]$$

$$\mathbf{C}_\ell(v) = \sum_{j=0}^{m} N_{j,q}(v)\mathbf{P}_{\ell,j} \qquad \ell = 0, \ldots, s \quad v \in [0,1] \tag{10.28}$$

be two sets of nonrational B-spline curves satisfying the compatibility conditions (see Figure 10.18):



(a)



(b)

Figure 10.17. The change of $\mathbf{B}(v)$ and the Frenet frame along the trajectory. (a) Planar trajectory; (b) three-dimensional trajectory.

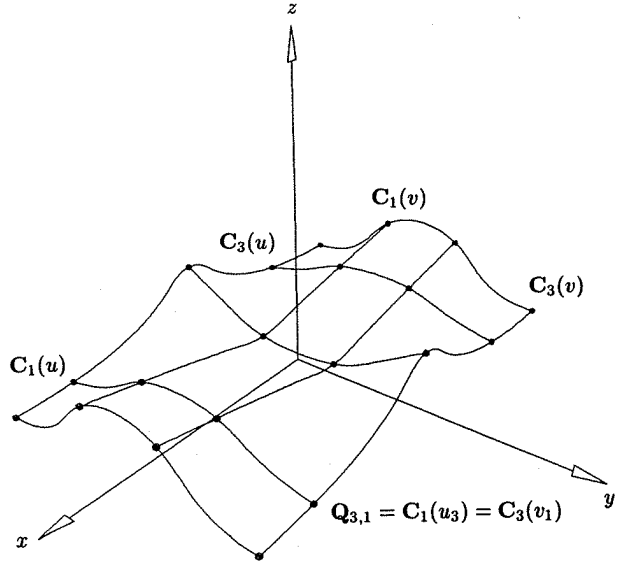


Figure 10.18. Compatible network of curves.

- as independent sets they are compatible in the B-spline sense, that is, all the $\mathbf{C}_k(u)$ are defined on a common knot vector, U^C, and all the $\mathbf{C}_\ell(v)$ are defined on a common knot vector, V^C;
- there exist parameters $0 = u_0 < u_1 < \cdots < u_{s-1} < u_s = 1$ and $0 = v_0 < v_1 < \cdots < v_{r-1} < v_r = 1$ such that

$$\mathbf{Q}_{\ell,k} = \mathbf{C}_k(u_\ell) = \mathbf{C}_\ell(v_k) \qquad k = 0, \ldots, r \quad \ell = 0, \ldots, s \tag{10.29}$$

We want to construct a NURBS surface, $\mathbf{S}(u, v)$, which interpolates the two sets of curves, that is

$$\begin{aligned} \mathbf{S}(u_\ell, v) &= \mathbf{C}_\ell(v) \qquad \ell = 0, \ldots, s \\ \mathbf{S}(u, v_k) &= \mathbf{C}_k(u) \qquad k = 0, \ldots, r \end{aligned} \tag{10.30}$$

Notice that we have restricted the curves to be nonrational. Theoretically, the construction given later can be carried out in homogeneous space for rational curves; but it is generally not practical to do so, and the three-dimensional results can be unpredictable. If rational curves are involved, we recommend using the constrained approximation techniques of Chapter 9 to obtain nonrational approximations of the rational curves to within necessary tolerances. We refer the reader to Lin and Hewitt [Lin94] for more detail on the case of rational curves.

Gordon developed surfacing techniques satisfying Eq. (10.30) at General Motors Corporation in the late 1960s; hence they bear his name, *Gordon surfaces* [Gord69, 71, 93]. In particular, he showed that if $\{\phi_\ell(u)\}_{\ell=0}^{s}$ and $\{\psi_k(v)\}_{k=0}^{r}$ are any two sets of blending functions satisfying

$$\phi_\ell(u_i) = \begin{cases} 0 & \text{if } \ell \neq i \\ 1 & \text{if } \ell = i \end{cases}$$
$$\psi_k(v_i) = \begin{cases} 0 & \text{if } k \neq i \\ 1 & \text{if } k = i \end{cases} \tag{10.31}$$

then the surface given by

$$\begin{aligned} \mathbf{S}(u,v) &= \sum_{\ell=0}^{s} \mathbf{C}_\ell(v)\phi_\ell(u) + \sum_{k=0}^{r} \mathbf{C}_k(u)\psi_k(v) - \sum_{\ell=0}^{s}\sum_{k=0}^{r} \mathbf{Q}_{\ell,k}\,\phi_\ell(u)\,\psi_k(v) \\ &= \mathbf{L}_1(u,v) + \mathbf{L}_2(u,v) - \mathbf{T}(u,v) \end{aligned} \tag{10.32}$$

satisfies the interpolation conditions of Eq. (10.30). Hence, the desired surface is composed of three simpler surfaces, namely two lofted surfaces, $\mathbf{L}_1(u,v)$ and $\mathbf{L}_2(u,v)$, and the tensor product, $\mathbf{T}(u,v)$. Gordon experimented with various sets of blending functions, including simple step functions, Lagrange polynomials, and cubic splines.

Our task is to produce a NURBS representation of $\mathbf{S}(u,v)$. Clearly, if $\mathbf{L}_1(u,v)$, $\mathbf{L}_2(u,v)$, and $\mathbf{T}(u,v)$ are all compatible in the B-spline sense (defined on the same knot vectors), then they can be added and subtracted by applying the corresponding operations to their control points, that is, the control points $\mathbf{P}_{i,j}$ of $\mathbf{S}(u,v)$ are computed by

$$\mathbf{P}_{i,j} = \mathbf{P}_{i,j}^{L_1} + \mathbf{P}_{i,j}^{L_2} - \mathbf{P}_{i,j}^{T} \tag{10.33}$$

where $\mathbf{P}_{i,j}^{L_1}$, $\mathbf{P}_{i,j}^{L_2}$, and $\mathbf{P}_{i,j}^{T}$ are the control points of $\mathbf{L}_1(u,v)$, $\mathbf{L}_2(u,v)$, and $\mathbf{T}(u,v)$, respectively. Furthermore, it is easy to obtain NURBS representations of $\mathbf{L}_1(u,v)$, $\mathbf{L}_2(u,v)$, and $\mathbf{T}(u,v)$; the first two are skinned surfaces, and the latter is obtained by interpolating the points $\mathbf{Q}_{\ell,k}$ (see Section 9.3). The choice of parameters and knots for the skinning and point interpolation processes is determined by the given parameters, $\{u_\ell\}$ and $\{v_k\}$, and the knots U^C and V^C of the $\{\mathbf{C}_k(u)\}$ and $\{\mathbf{C}_\ell(v)\}$. We illustrate with a simple example.

Example

Ex10.1 The data defining the network of curves in Figure 10.18 is

$$\mathbf{C}_k(u):\ n = 5, p = 3,\ U^C = \left\{0,0,0,0,\frac{3}{10},\frac{7}{10},1,1,1,1\right\}$$

$$\mathbf{C}_\ell(v):\ m = 4, q = 2,\ V^C = \left\{0,0,0,\frac{2}{5},\frac{3}{5},1,1,1\right\}$$

Figures 10.19a and 10.19b show the control points and the resulting surface of the u-directional lofting operation. The knot vectors are

$$U^{L_1} = \{0,0,0,0,1,1,1,1\}$$
$$V^{L_1} = \left\{0,0,0,\frac{2}{5},\frac{3}{5},1,1,1\right\}$$

The v-directional lofting is illustrated in Figures 10.20a and 10.20b. The knot vectors are

$$U^{L_2} = \left\{0,0,0,0,\frac{3}{10},\frac{7}{10},1,1,1,1\right\}$$

$$V^{L_2} = \{0,0,0,0.321,0.647,1,1,1\}$$

The tensor product surface is depicted in Figures 10.21a and 10.21b. The knots are

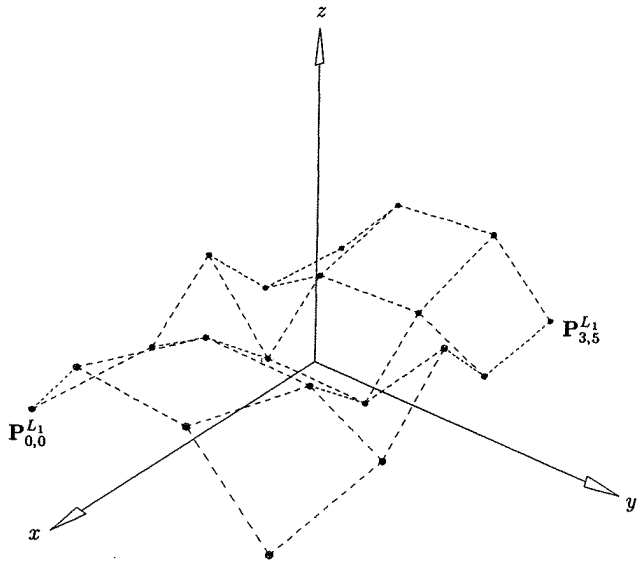
$$U^{T} = \{0,0,0,0,1,1,1,1\}$$
$$V^{T} = \{0,0,0,0.32,0.648,1,1,1\}$$

Notice the slight difference in the knot vectors V^{L_2} and V^T. This is because surface skinning and surface interpolation compute knot vectors differently (but meaningfully for their respective operations). The former uses the curve control points, while the latter utilizes the intersection points of the curve network. However, knots which are very close together are unnecessary and undesirable. A solution is to pass all previously computed knot vectors into the skinning and interpolation routines, allow them to compute new knots using their respective algorithms, compare these new knots with existing knots, and replace new knots with existing ones where the difference is small. Degree elevation is not required in this example, as lofting and interpolation use the curve degrees. However, knot refinement is required, and is depicted in Figures 10.22a–10.22c. The Gordon surface construction is shown in Figure 10.23a, and the surface is illustrated in Figure 10.23b. The final knots are

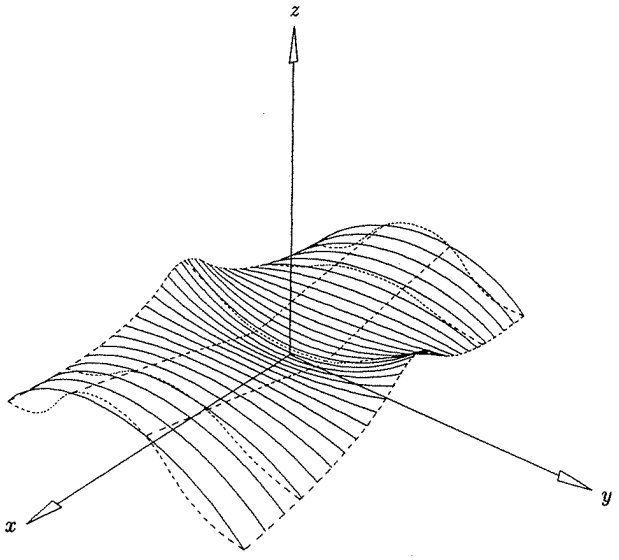
$$U = \left\{0,0,0,0,\frac{3}{10},\frac{7}{10},1,1,1,1\right\}$$

$$V = \{0,0,0,0.32,0.321,0.4,0.6,0.647,0.648,1,1,1\}$$

As alluded to earlier, the degrees of the input curves, those degrees used for lofting, and those used to construct $\mathbf{T}(u,v)$ can all be different. The final degree (p,q) of $\mathbf{S}(u,v)$ is simply the maximum of those used. The degrees of $\mathbf{L}_1(u,v)$, $\mathbf{L}_2(u,v)$, and $\mathbf{T}(u,v)$ must be raised to (p,q) before knot refinement. A sketch of the complete algorithm follows. `Ck[]` and `Cl[]` are arrays of input curves, and `ul[]` and `vk[]` are the parameters of the curve intersection points. To be

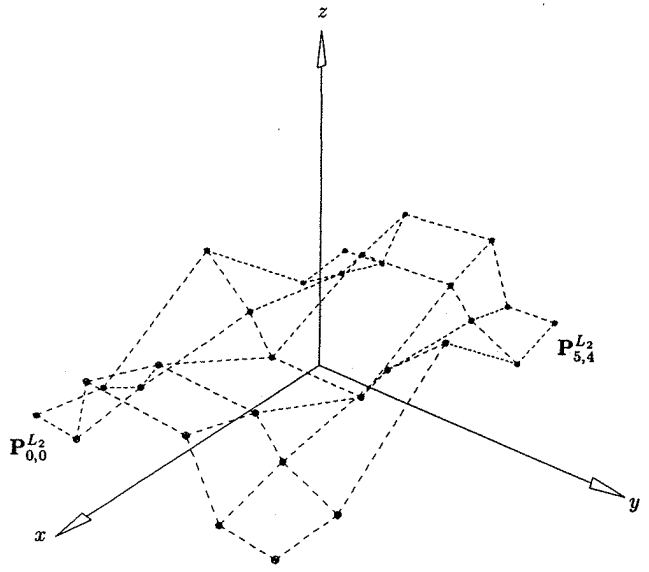


(a)

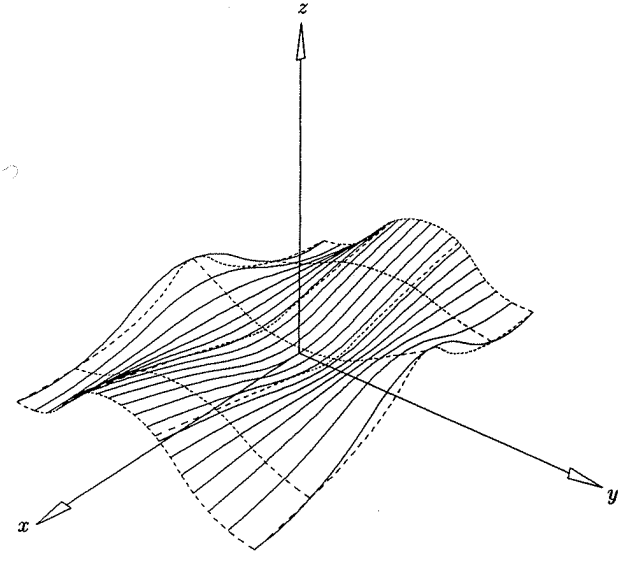


(b)

Figure 10.19. u-directional lofted surface. (a) Control net; (b) surface interpolating v curves.

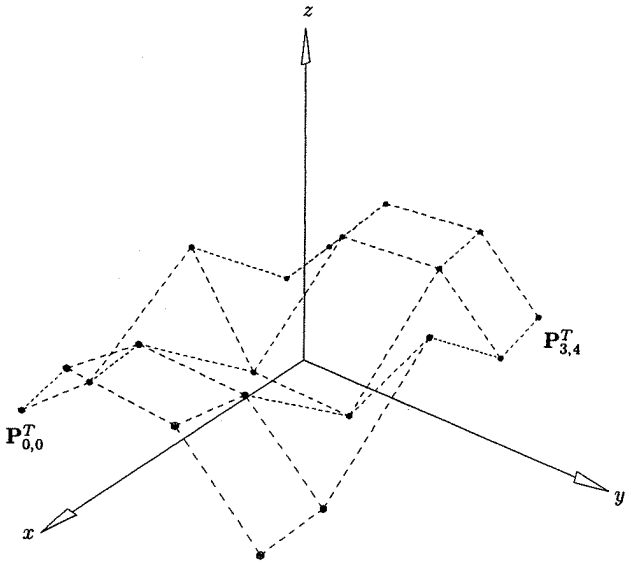


(a)

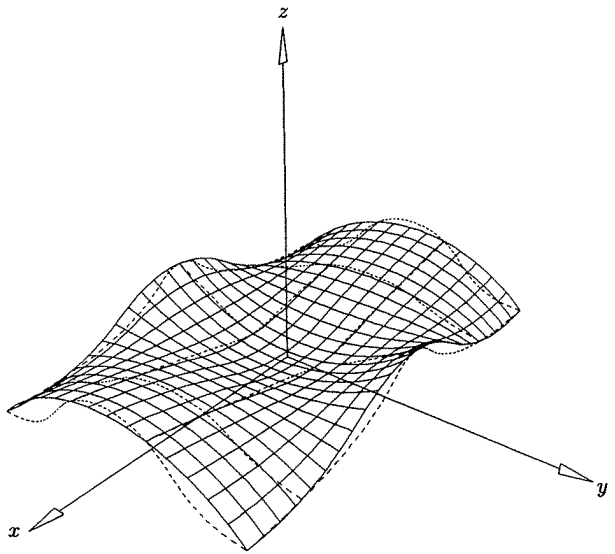


(b)

Figure 10.20. v-directional lofted surface. (a) Control net; (b) surface interpolating u curves.

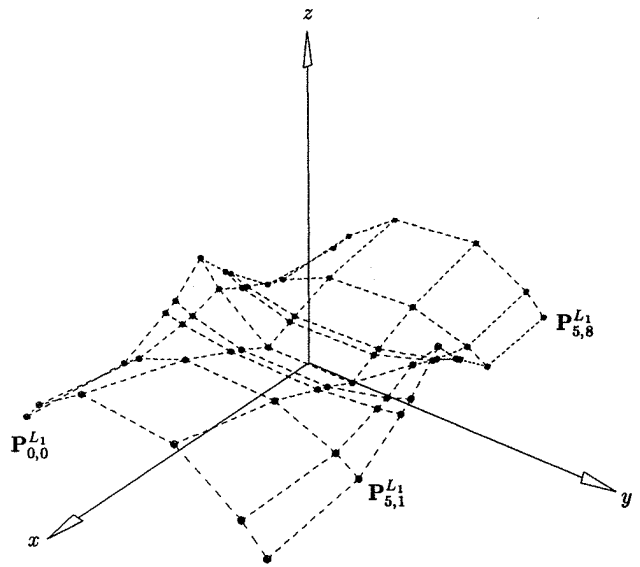


(a)

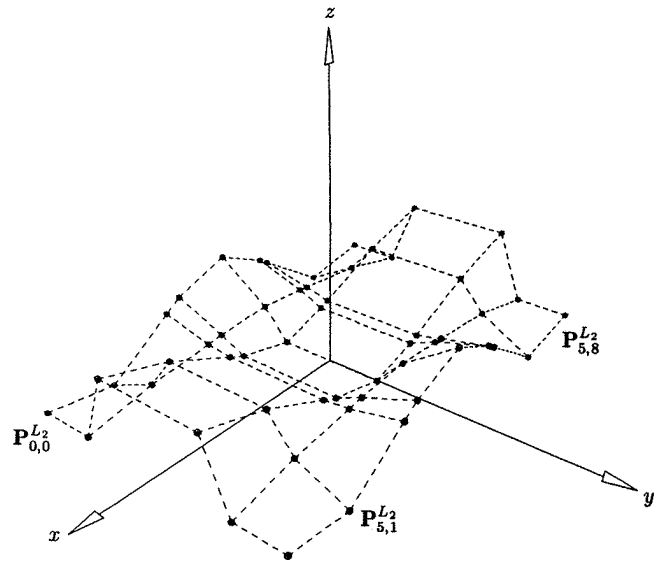


(b)

Figure 10.21. Tensor product surface. (a) Control net; (b) surface interpolating intersection points.

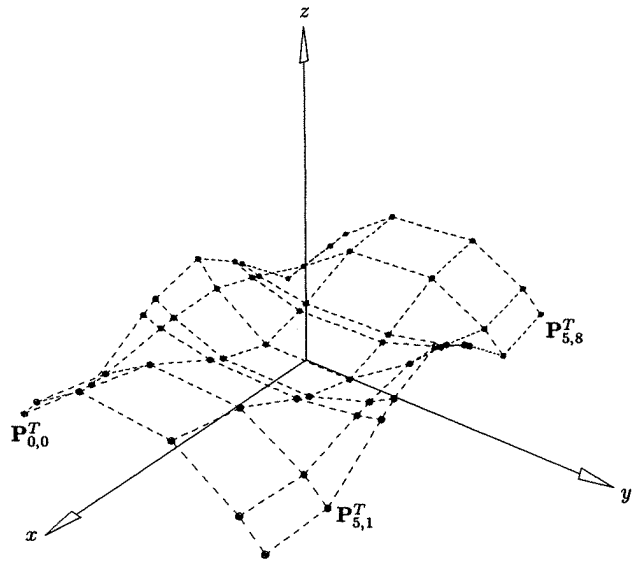


(a)



(b)

Figure 10.22. Knot refinement to construct a Gordon surface. (a) Ruled surface in the u direction refined; (b) ruled surface in the v direction refined.

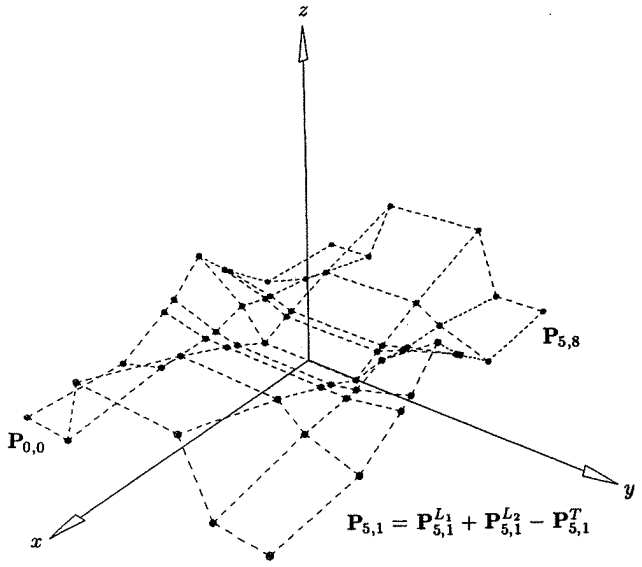


(c)

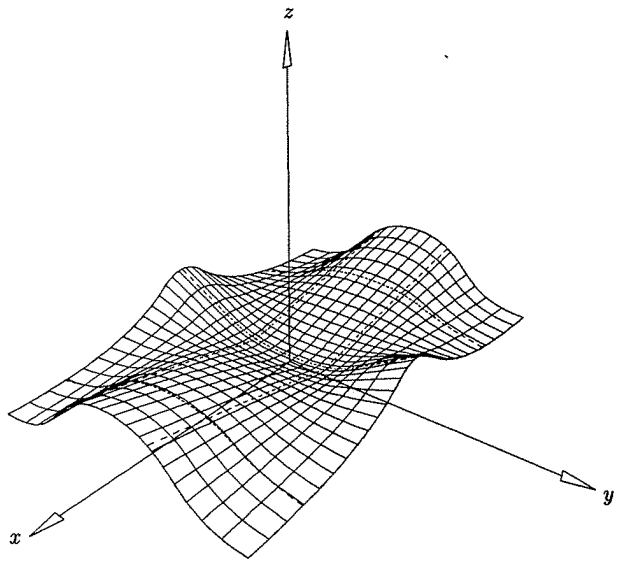
Figure 10.22. (*Continued.*) (c) tensor product in both directions refined.

completely general, the degrees `pl,ql,pt,qt` used for lofting and interpolation are also input. `n,m,p,q,U,V,Pij` define the output Gordon surface.

```
ALGORITHM A10.3
  GordonSurface(Ck,Cl,ul,vk,r,s,pl,ql,pt,qt,n,m,p,q,U,V,Pij)
    { /*  Create Gordon surface.  */
      /*  Input:  Ck,Cl,ul,vk,r,s,pl,ql,pt,qt  */
      /*  Output: n,m,p,q,U,V,Pij  */
    pc = degree of the Ck curves.
    qc = degree of the Cl curves.
    Choose a suitable U1 and loft to obtain L1(u,v)
      (degree (pl,qc)).
    Choose a suitable V2 and loft to obtain L2(u,v)
      (degree (pc,ql)).
    Choose suitable UT and VT, and interpolate to obtain T(u,v)
      (degree (pt,qt)).
    p = max(pc,pl,pt).
    q = max(qc,ql,qt).
    Degree elevate L1(u,v), L2(u,v) and T(u,v) to (p,q)
      as necessary.
    U = Merge(U1,U2,UT).
    V = Merge(V1,V2,VT).
```

(a)



(b)

Figure 10.23. A Gordon surface. (a) Control net; (b) surface interpolating u- and v-directional curves.

```
Refine knot vectors of L1(u,v), L2(u,v) and T(u,v)
  as necessary.
n = (number of u-knots)-p-2.
m = (number of v-knots)-q-2.
Compute surface control points Pij by Eq.(10.33).
}
```

10.6 Coons Surfaces

Assume now that we have four curves

$$\mathbf{C}_k(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_{k,i} \qquad k = 0,1 \quad u \in [0,1]$$

$$\mathbf{C}_\ell(v) = \sum_{j=0}^{m} N_{j,q}(v)\mathbf{P}_{\ell,j} \qquad \ell = 0,1 \quad v \in [0,1] \tag{10.34}$$

and that we want to construct a surface having these four curves as its boundaries. Furthermore, we assume that the curves satisfy the compatibility conditions

- as independent sets they are compatible in the B-spline sense, that is, the two $\mathbf{C}_k(u)$ are defined on a common knot vector, U, and the two $\mathbf{C}_\ell(v)$ are defined on a common knot vector, V;
- $$\begin{aligned} \mathbf{S}_{0,0} &= \mathbf{C}_{k=0}(u=0) = \mathbf{C}_{\ell=0}(v=0) \\ \mathbf{S}_{1,0} &= \mathbf{C}_{k=0}(u=1) = \mathbf{C}_{\ell=1}(v=0) \\ \mathbf{S}_{0,1} &= \mathbf{C}_{k=1}(u=0) = \mathbf{C}_{\ell=0}(v=1) \\ \mathbf{S}_{1,1} &= \mathbf{C}_{k=1}(u=1) = \mathbf{C}_{\ell=1}(v=1) \end{aligned} \tag{10.35}$$

(see Figure 10.24). We do not restrict the curves to be nonrational in this section. The constructions given here usually yield satisfactory surfaces, even when carried out in homogeneous space. However, this complicates the algorithms. For example, Eq. (10.35) must hold in homogeneous space. This requires corner weight compatibility, which can require rational reparameterization of the curves (Section 6.4).

The *bilinearly blended Coons patch* is defined as

$$\mathbf{S}(u,v) = \mathbf{R}_1(u,v) + \mathbf{R}_2(u,v) - \mathbf{T}(u,v) \tag{10.36}$$

where $\mathbf{R}_1(u,v)$ and $\mathbf{R}_2(u,v)$ are ruled surfaces between $\mathbf{C}_{k=0}(u)$ and $\mathbf{C}_{k=1}(u)$, and $\mathbf{C}_{\ell=0}(v)$ and $\mathbf{C}_{\ell=1}(v)$, respectively, and $\mathbf{T}(u,v)$ is the bilinear tensor product surface

$$\mathbf{T}(u,v) = \begin{bmatrix} 1 & u \end{bmatrix} \begin{bmatrix} \mathbf{S}_{0,0} & \mathbf{S}_{0,1} \\ \mathbf{S}_{1,0} & \mathbf{S}_{1,1} \end{bmatrix} \begin{bmatrix} 1 \\ v \end{bmatrix}$$

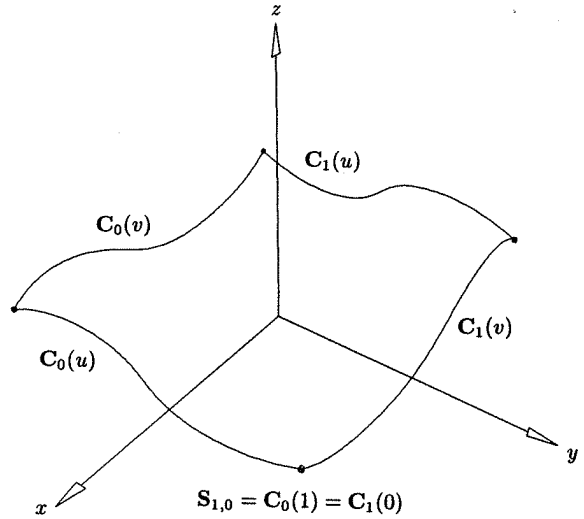
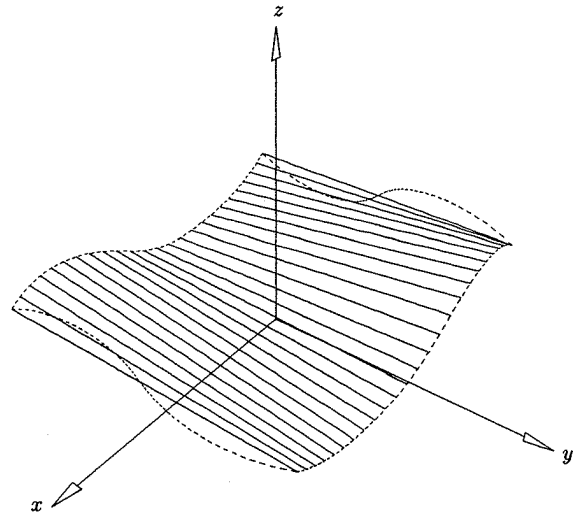


Figure 10.24. Compatible boundary curves to define a bilinearly blended Coons patch.

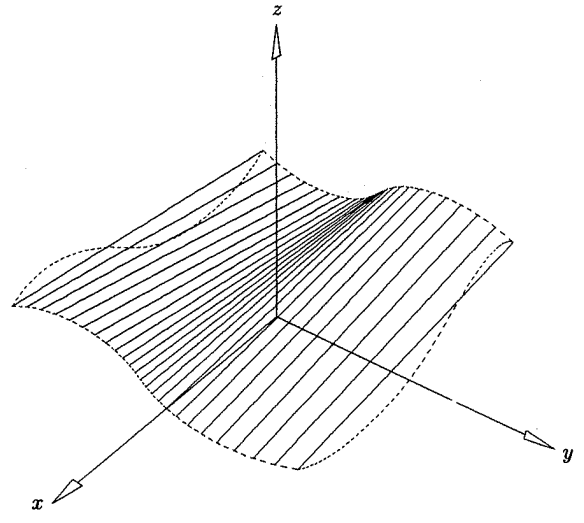
Clearly, this is a special case of the Gordon surface, and therefore Eq. (10.33) and Algorithm `A10.3` (simplified, and modified to handle rational curves) can be used to obtain a NURBS representation of the Coons surface given by Eq. (10.36). The input arguments `pl,ql,pt,qt` are set to 1 in this case. Figures 10.25a–10.25d show the construction of the bilinearly blended Coons patch defined by the curves in Figure 10.24. Figure 10.25a shows a ruled surface in the u direction, and Figure 10.25b illustrates the v-directional ruled surface. Figure 10.25c shows the bilinear surface defined by the four corner points. The bilinearly blended Coons patch is depicted in Figure 10.25d. It is important to note that the bilinearly blended Coons patch is, in general, not a bilinear surface.

Coons developed his patches in the mid 1960s [Coon67]. Although we present the bilinearly blended Coons patch as a special case of Gordon surfaces, we remark that Coons' work preceded and influenced the development of Gordon surfaces. Coons was interested in interpolation to position and tangent information in bidirectional networks of curves, as was Gordon later. Coons constructed a patch within each set of four curves, hence his was a local interpolation method. Gordon's method represents a global interpolation to a network of curves.

If two bilinearly blended Coons patches share a common boundary curve, they possess, in general, only C^0 continuity across that boundary; that is, the combined surface will have a crease along the common boundary. In order to obtain smooth, C^n continuous ($n > 0$) joins of his patches, Coons developed higher order patches using *cross-boundary derivative fields*. To illustrate, we present the *bicubically blended Coons patch*. Let $\mathbf{C}_k(u)$ and $\mathbf{C}_\ell(v)$ be four boundary

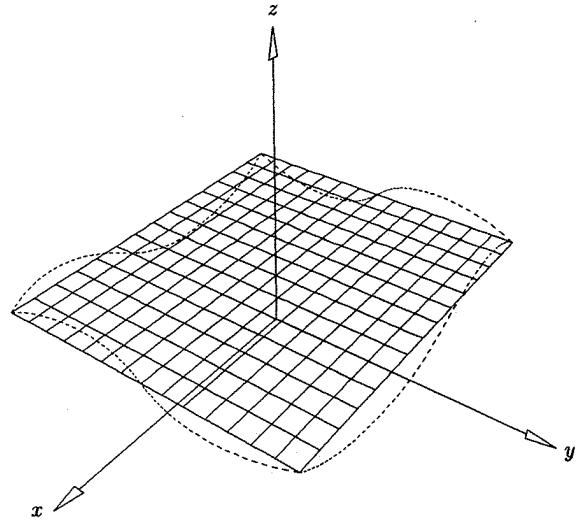


(a)



(b)

Figure 10.25. Bilinearly blended Coons patch. (a) A ruled surface in the u direction; (b) a ruled surface in the v direction; (c) a bilinear surface; (d) a bilinearly blended Coons patch.



(c)

Figure 10.25. (*Continued.*)

curves as in Eq. (10.34), satisfying the two compatibility conditions given. We also assume four cross-boundary derivative fields (see Figure 10.26)

$$\mathbf{D}_k(u) = \sum_{i=0}^{n} N_{i,p}(u)\,\mathbf{Q}_{k,i} \qquad k = 0,1 \quad u \in [0,1]$$

$$\mathbf{D}_\ell(v) = \sum_{j=0}^{m} N_{j,q}(v)\,\mathbf{Q}_{\ell,j} \qquad \ell = 0,1 \quad v \in [0,1] \tag{10.37}$$

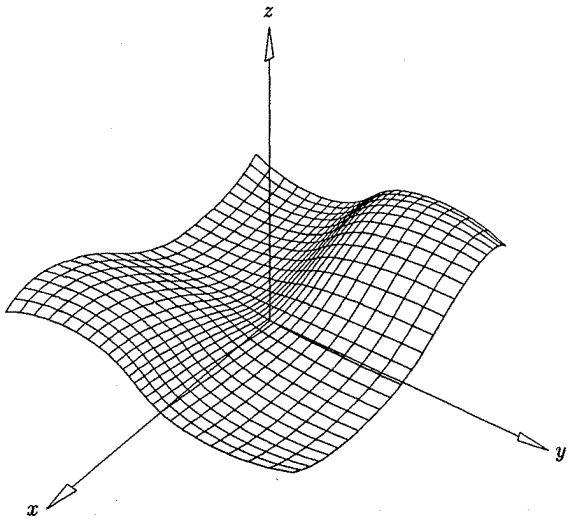
We want to construct a surface which has the $\mathbf{C}_k(u)$ and $\mathbf{C}_\ell(v)$ as its boundaries, and the $\mathbf{D}_k(u)$ and $\mathbf{D}_\ell(v)$ as its first partial derivatives along the boundaries, that is

$$\mathbf{D}_{k=0}(u) = \mathbf{S}_v(u,0) \qquad \mathbf{D}_{k=1}(u) = \mathbf{S}_v(u,1)$$

$$\mathbf{D}_{\ell=0}(v) = \mathbf{S}_u(0,v) \qquad \mathbf{D}_{\ell=1}(v) = \mathbf{S}_u(1,v)$$

The compatibility conditions

- the $\mathbf{C}_k(u)$ and the $\mathbf{D}_k(u)$ are compatible in the B-spline sense, as are the $\mathbf{C}_\ell(v)$ and $\mathbf{D}_\ell(v)$;
- $$\mathbf{T}_{0,0} = \frac{d\mathbf{D}_{k=0}(u=0)}{du} = \frac{d\mathbf{D}_{\ell=0}(v=0)}{dv}$$



(d)

Figure 10.25. (*Continued.*)

$$\mathbf{T}_{1,0} = \frac{d\mathbf{D}_{k=0}(u=1)}{du} = \frac{d\mathbf{D}_{\ell=1}(v=0)}{dv}$$

$$\mathbf{T}_{0,1} = \frac{d\mathbf{D}_{k=1}(u=0)}{du} = \frac{d\mathbf{D}_{\ell=0}(v=1)}{dv}$$

$$\mathbf{T}_{1,1} = \frac{d\mathbf{D}_{k=1}(u=1)}{du} = \frac{d\mathbf{D}_{\ell=1}(v=1)}{dv} \tag{10.38}$$

must hold. The $\mathbf{T}_{k,\ell}$ are the four *twist vectors.*

The bicubically blended Coons patch is defined by

$$\mathbf{S}(u,v) = \mathbf{S}_1(u,v) + \mathbf{S}_2(u,v) - \mathbf{T}(u,v) \tag{10.39}$$

where $\mathbf{S}_1(u,v)$ is a cubic blend of the data $\mathbf{C}_k(u)$, $\mathbf{D}_k(u)$, $k = 0,1$; $\mathbf{S}_2(u,v)$ is a cubic blend of the data $\mathbf{C}_\ell(v)$, $\mathbf{D}_\ell(v)$, $\ell = 0,1$; and $\mathbf{T}(u,v)$ is a bicubic tensor product surface (defined by 16 vector coefficients). Coons used cubic Hermite polynomials to blend the boundary data and to form $\mathbf{T}(u,v)$. Since our goal is to obtain a NURBS surface, we use cubic Bézier methods to blend and to form $\mathbf{T}(u,v)$. $\mathbf{S}_1(u,v)$ is a cubic Bézier in the v direction. Its four rows of control points are computed as $(i = 0,\ldots,n)$

$$\mathbf{P}^{S_1}_{i,0} = \mathbf{P}_{k=0,i}$$

$$\mathbf{P}^{S_1}_{i,1} = \mathbf{P}_{k=0,i} + \frac{1}{3}\mathbf{Q}_{k=0,i}$$

$$\mathbf{P}^{S_1}_{i,2} = \mathbf{P}_{k=1,i} - \frac{1}{3}\mathbf{Q}_{k=1,i}$$

$$\mathbf{P}^{S_1}_{i,3} = \mathbf{P}_{k=1,i} \tag{10.40}$$

Note that the two interior rows are derived using the Bézier derivative formulas (Eq. [1.10]). The control points of $\mathbf{S}_2(u,v)$ are computed analogously, i.e.

$$\mathbf{P}^{S_2}_{0,j} = \mathbf{P}_{\ell=0,j}$$

$$\mathbf{P}^{S_2}_{1,j} = \mathbf{P}_{\ell=0,j} + \frac{1}{3}\mathbf{Q}_{\ell=0,j}$$

$$\mathbf{P}^{S_2}_{2,j} = \mathbf{P}_{\ell=1,j} - \frac{1}{3}\mathbf{Q}_{\ell=1,j}$$

$$\mathbf{P}^{S_2}_{3,j} = \mathbf{P}_{\ell=1,j} \tag{10.41}$$

$\mathbf{T}(u,v)$ is a bicubic Bézier surface. Equations (1.10) and (3.24) are used to derive the 16 control points, $\mathbf{P}^T_{i,j}$, $i,j = 0,1,2,3$. As an example, we show how

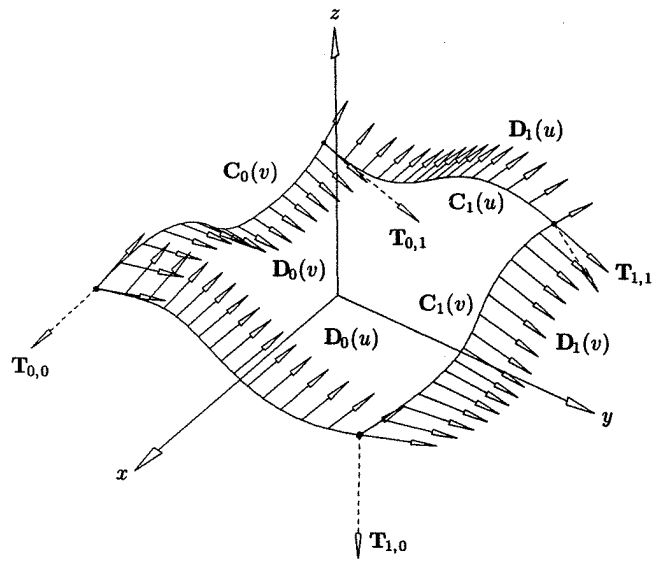


Figure 10.26. Compatible boundary curves along with cross-boundary tangent vector functions and twists.

to compute the four control points, $\mathbf{P}^T_{i,j}$, $i,j = 0,1$, adjacent to the corner $\mathbf{S}_{0,0}$. The other twelve are analogous.

$$\mathbf{P}^T_{0,0} = \mathbf{S}_{0,0} = \mathbf{C}_{k=0}(u=0) = \mathbf{C}_{\ell=0}(v=0)$$

$$\mathbf{P}^T_{1,0} = \frac{1}{3}\mathbf{D}_{\ell=0}(v=0) + \mathbf{P}^T_{0,0}$$

$$\mathbf{P}^T_{0,1} = \frac{1}{3}\mathbf{D}_{k=0}(u=0) + \mathbf{P}^T_{0,0}$$

$$\mathbf{P}^T_{1,1} = \frac{1}{9}\mathbf{T}_{0,0} + \mathbf{P}^T_{1,0} + \mathbf{P}^T_{0,1} - \mathbf{P}^T_{0,0} \tag{10.42}$$

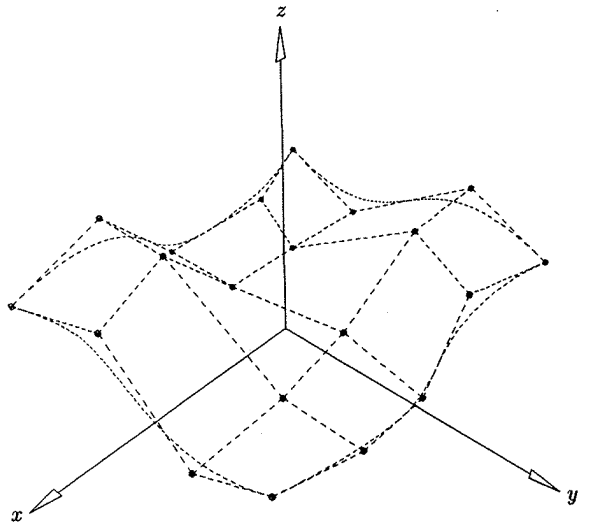
The surfaces $\mathbf{S}_1(u,v)$, $\mathbf{S}_2(u,v)$, and $\mathbf{T}(u,v)$ must then be made compatible using degree elevation and knot refinement. Subsequently, the control points $\mathbf{P}_{i,j}$ of the Coons surface are computed by the formula

$$\mathbf{P}_{i,j} = \mathbf{P}^{S_1}_{i,j} + \mathbf{P}^{S_2}_{i,j} - \mathbf{P}^T_{i,j} \tag{10.43}$$

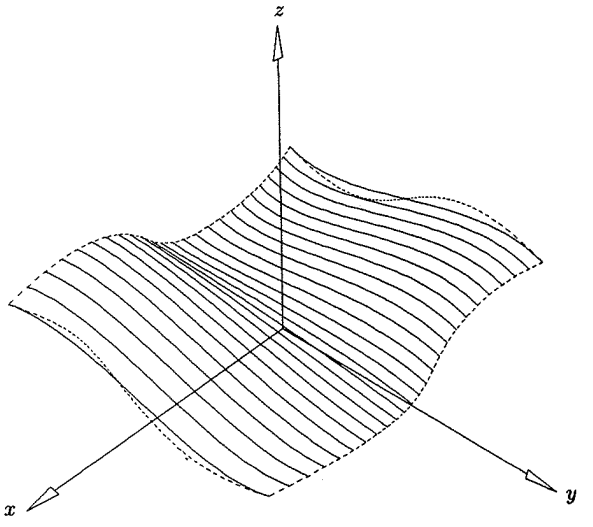
Figures 10.27–10.30 show the process of constructing a bicubically blended Coons patch. Figures 10.27a and 10.27b show a u-directional cubically blended surface, $\mathbf{S}_2(u,v)$, along with the defining control net. v-directional cubic blending is illustrated in Figures 10.28a and 10.28b $\left(\mathbf{S}_1(u,v)\right)$. A bicubic tensor product surface is depicted in Figures 10.29a and 10.29b. Finally, the Coons patch is illustrated in Figures 10.30a and 10.30b, after appropriate degree elevations and knot refinements are performed.

In Algorithm `A10.4`, `Ck[],Cl[]` and `Dk[],Dl[]` are arrays of boundary curves and cross-boundary derivatives, respectively. `n,m,p,q,U,V,Pij` define the output bicubically blended Coons surface.

```
ALGORITHM A10.4
  BicubicBlendCoons(Ck,Cl,Dk,Dl,n,m,p,q,U,V,Pij)
    { /*  Create bicubically blended Coons surface.  */
      /*  Input:  Ck,Cl,Dk,Dl  */
      /*  Output: n,m,p,q,U,V,Pij  */
    Compute the ctrl pts of S1(u,v) (Eq.10.40).
    Compute the ctrl pts of S2(u,v) (Eq.10.41).
    Compute the ctrl pts of T(u,v) (Eq.10.42).
    pcd = degree of the Ck curves and Dk derivatives.
    qcd = degree of the Cl curves and Dl derivatives.
    p = max(3,pcd);
    q = max(3,qcd);
    Degree elevate S1(u,v), S2(u,v) and T(u,v) to (p,q)
        if necessary.
    U = u-knot vector of S1(u,v).
    V = v-knot vector of S2(u,v).
```

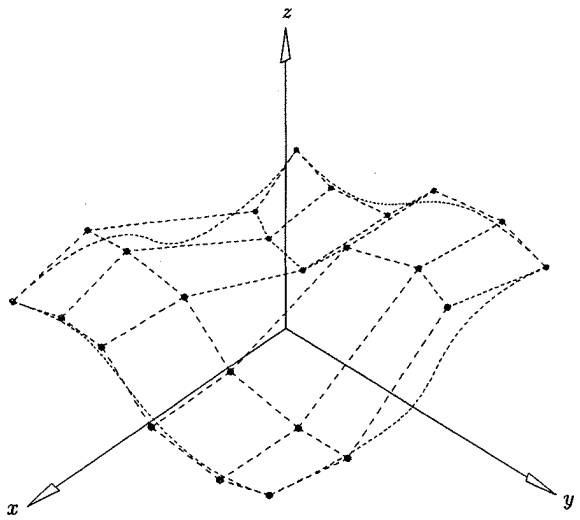



(a)

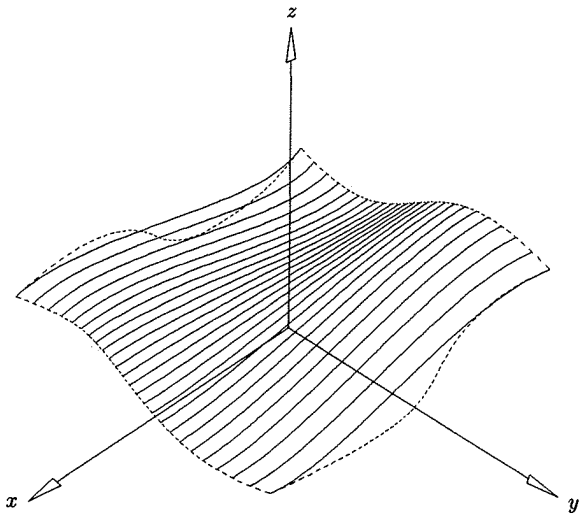


(b)

Figure 10.27. Cubically blended surface in the u direction. (a) Control net; (b) surface (Bézier in the u and B-spline in the v direction).

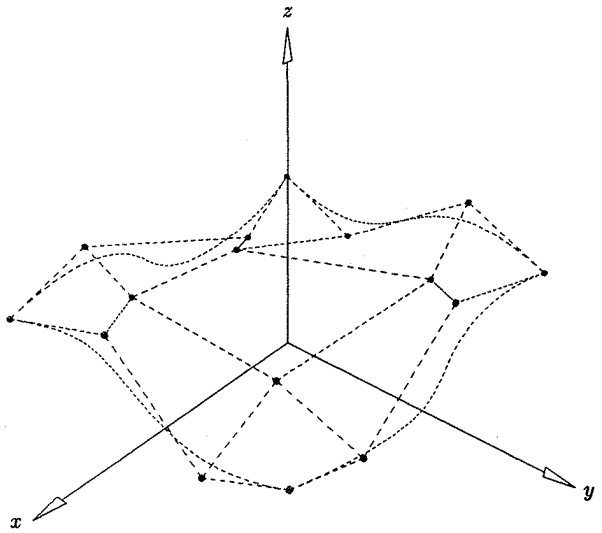


(a)

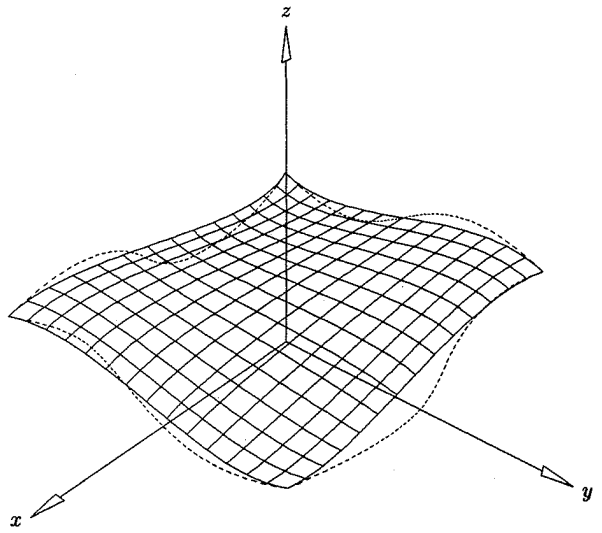


(b)

Figure 10.28. Cubically blended surface in the v direction. (a) Control net; (b) surface (B-spline in the u and Bézier in the v direction).

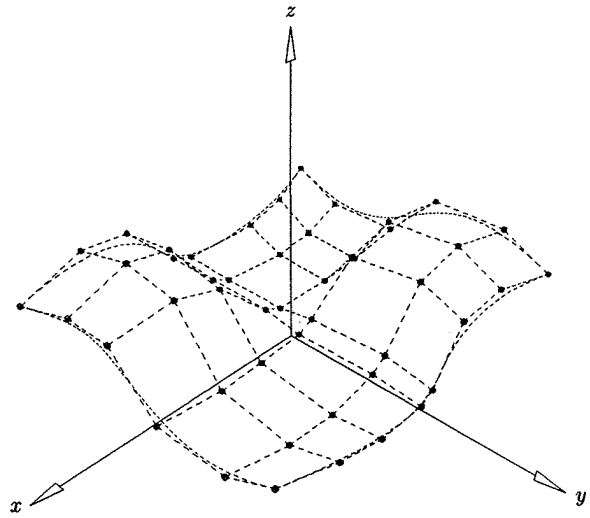


(a)

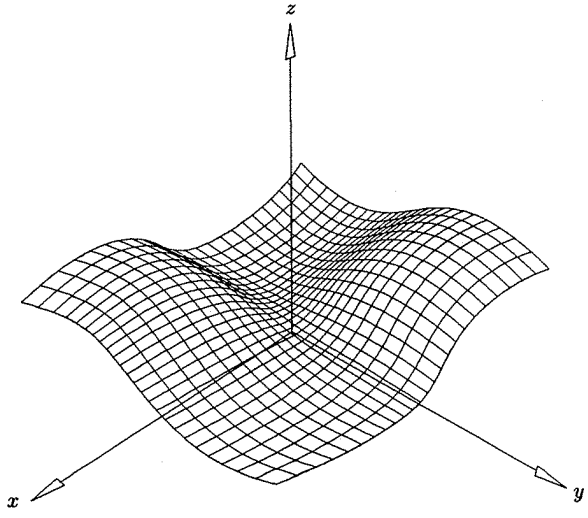


(b)

Figure 10.29. Bicubic tensor product surface. (a) Control net; (b) bicubic Bézier surface.



(a)



(b)

Figure 10.30. Bicubically blended Coons surface. (a) Control net after degree raising and knot refinement; (b) Coons surface.


```
Knot refine S1(u,v), S2(u,v) and T(u,v) to
    U and V if necessary.
n = (number of u-knots)-p-2.
m = (number of v-knots)-q-2.
Compute surface control points P_{i,j} by Eq.(10.43).
}
```

In practice, it is not easy to construct a bicubically blended Coons patch. First, the cross-boundary derivative fields are seldom available; even when they are available, the magnitudes of the derivatives may not be appropriate. Furthermore, one can almost never expect twist vector compatibility (Eq. [10.38]) to hold, that is, in general

$$\frac{d\mathbf{D}_{k=0}(u=0)}{du} \neq \frac{d\mathbf{D}_{\ell=0}(v=0)}{dv}$$

and likewise for the other three corners. The most common use of this patch is to attempt to fill a three- or four-sided hole in a patchwork of surfaces, smoothly in the G^1 sense. The boundary data is extracted from the neighboring surfaces. However, in this case derivative magnitudes are generally not meaningful, twist vectors are not compatible, and hence G^1 continuity between the patch and its neighboring surfaces is not possible. If the application can tolerate a small discontinuity in tangent direction across a boundary, then an angular tolerance can be specified and one of two approaches taken:

- use the iterative approximation techniques of Chapter 9 to obtain nonrational derivative fields which are twist compatible, and which approximate the vector directions of the original derivative fields to within the specified angular tolerance. If any one of the boundary curves is rational, then the problem becomes much more complex, as the twist vectors and derivative fields must be embedded in homogeneous space in such a way as to obtain four-dimensional compatibility while still maintaining the angular tolerance on the three-dimensional vector directions. Equations (4.7) and (4.26) are useful in this process;
- let

$$\mathbf{T}^k_{0,0} = \frac{d\mathbf{D}_{k=0}(u=0)}{du} \qquad \mathbf{T}^\ell_{0,0} = \frac{d\mathbf{D}_{\ell=0}(v=0)}{dv}$$

  $\mathbf{T}^k_{0,0}$ and $\mathbf{T}^\ell_{0,0}$ are computed from the following control points of $\mathbf{D}_{k=0}(u)$ and $\mathbf{D}_{\ell=0}(v)$: $\mathbf{Q}_{k=0,0}$, $\mathbf{Q}_{k=0,1}$ and $\mathbf{Q}_{\ell=0,0}$, $\mathbf{Q}_{\ell=0,1}$. Set

$$\mathbf{T}_{0,0} = \frac{1}{2}\left(\mathbf{T}^k_{0,0} + \mathbf{T}^\ell_{0,0}\right)$$

  Keeping $\mathbf{Q}_{k=0,0}$ and $\mathbf{Q}_{\ell=0,0}$ fixed, $\mathbf{Q}_{k=0,1}$ and $\mathbf{Q}_{\ell=0,1}$ can be modified to yield the common twist, $\mathbf{T}_{0,0}$. Knot insertion, applied prior to modifying $\mathbf{Q}_{k=0,1}$ and $\mathbf{Q}_{\ell=0,1}$, can be used to bound the angular error and restrict its domain of influence. The process is even more complicated if any of the curves or derivative fields are rational.

For a deeper and broader study of Coons patches, the reader should consult Barnhill's article [Barn93] and the references cited therein.