CHAPTER

6

A Parametric Triangular Patch Based on Generalized Conics

Bernd Hamann, Gerald Farin, and Gregory M. Nielson

6.1. Introduction

In graphics and computer aided design (CAD) applications, one is very often concerned with the question of how to generate a smooth surface (a surface that is G^1-continuous between adjacent triangular patches) when only point and normal (tangent plane) data are given in three-dimensional space. Examples are the methods described in Nielson [14] and Piper [16].

The surface scheme described in this paper is based on a simple curve scheme of degree-elevated conic sections. Using the concept of degree-elevated conics allows both modeling conics and rational cubic curves with inflection points. *Generalized conics* were introduced by Ball [1]–[3]; they are rational cubic curves with standard conics as a subset (see also Boehm [7]). Our principle of combining curves with and without inflection points is similar to Ball's idea. In a first step, boundary curves (degree-elevated conics) for all triangle edges are constructed in rational Bézier representation. In a second step, three different surface building blocks are obtained from calculating generalized conics emanating from a triangle vertex and ending at a point belonging to an opposite boundary curve. The three surface building blocks are finally blended together in a convex combination to obtain a triangular patch.

The construction of all patches is the same. Therefore, only the generation of a single building block patch will be described. A surface patch $\mathbf{s}(\mathbf{u})$ can be seen as a point-valued mapping from $I\!R^3$ (triples of barycentric coordinates) into $I\!R^3$ (three-dimensional points on the patch). The (intrinsic) domain for each triangular patch will be the set of all triples (u_1, u_2, u_3) of barycentric coordinates for which $\sum u_i = 1$ and $u_i \geq 0$, $i = 1, 2, 3$. Each point on a patch will then be the image of (u_1, u_2, u_3) using the concept of the projectors described below.

The patch $\mathbf{s}(\mathbf{u})$ with vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ will be a convex combination of three different building blocks, called $\mathbf{s}_1(\mathbf{u})$, $\mathbf{s}_2(\mathbf{u})$, and $\mathbf{s}_3(\mathbf{u})$. The barycentric coordinates u_1, u_2, and u_3 for a point in the domain triangle are represented by $\mathbf{u}$. The resulting surface patch can be written as

$$\mathbf{s}(\mathbf{u}) = \sum_{i=1}^{3} w_i(\mathbf{u})\mathbf{s}_i(\mathbf{u}). \tag{6.1}$$

In this sum, $\mathbf{s}_i(\mathbf{u})$ is the building block of the patch (emanating from vertex $\mathbf{v}_i$) that interpolates to the positional data along all three triangle edges and to the normal information along the opposite edge $\mathbf{e}_i$, $i = 1, 2, 3$. The weight functions $w_i(\mathbf{u})$ are chosen as in Nielson [14]. The final patch will interpolate to the positional and normal information prescribed along all three edges.

In the following, an interpolation theorem will be given for the discrete case (three vertices and three normals given only). The concept of using degree-elevated conics will be presented to obtain the single patch building blocks using a planar curve scheme.

6.2. An Interpolation Theorem for Discrete Data

The weight functions $w_i(\mathbf{u})$ and the patch building blocks $\mathbf{s}_i(\mathbf{u})$, $i = 1, 2, 3$, occurring in (6.1) must have certain properties in order to obtain a triangular patch that interpolates to both positional and normal information prescribed along its three edges. Here, the following weight functions will be used:

$$w_i(\mathbf{u}) = \frac{B^2_{(1,1,1)-\mathbf{e}_i}(\mathbf{u})}{B^2_{(0,1,1)}(\mathbf{u}) + B^2_{(1,0,1)}(\mathbf{u}) + B^2_{(1,1,0)}(\mathbf{u})},$$

where $i = 1, 2, 3$. Here, $\mathbf{e}_1 = (1, 0, 0)$, $\mathbf{e}_2 = (0, 1, 0)$, and $\mathbf{e}_3 = (0, 0, 1)$. The functions $B^2_{(i,j,k)}$, $i + j + k = 2$, are the Bernstein polynomials (in terms of barycentric coordinates) of degree 2, defined as

$$B^2_{(i,j,k)} = \frac{2}{i!j!k!} u_1^i u_2^j u_3^k,$$

where $\sum u_i = 1$ and $u_i \geq 0$, $i = 1, 2, 3$. The relevant properties of these weight functions are

$$\begin{array}{lll} \text{(i)} & \sum_{i=1}^{3} w_i(\mathbf{u}) & \equiv 1, \\ \text{(ii)} & w_i(e_k) & = \delta_{i,k}, \quad i, k \in \{1, 2, 3\}, \\ \text{(iii)} & D_{\mathbf{d}}(w_i(e_i)) & = 0, \quad i \in \{1, 2, 3\}. \end{array}$$

Here, $\sum u_i = 1$, $u_i \geq 0$, $i = 1, 2, 3$. Edge $\mathbf{e}_1$ is characterized by barycentric coordinates $(0, u_2, u_3)$, edge $\mathbf{e}_2$ by $(u_1, 0, u_3)$, and edge $\mathbf{e}_3$ by $(u_1, u_2, 0)$. The symbol $\delta_{i,k}$ is the Kronecker delta and $D_{\mathbf{d}}$ is a *directional derivative* in any direction $\mathbf{d}$, where $\mathbf{d}$ is a vector expressed in terms of barycentric coordinates (d_1, d_2, d_3) and $\sum d_i = 0$ (see Farin [8], [9]).

The individual patch building blocks will each interpolate to *all three boundary curves* of the triangular patch and to the *normals along one edge*, formalized as

$$\text{(iv)} \quad \mathbf{s}_i[\mathbf{p}](\overline{e}) = \mathbf{p}(\overline{e}),$$

and

$$\text{(v)} \quad \mathbf{n}[\mathbf{s}_i[\mathbf{p}]](e_i) = \mathbf{n}[\mathbf{p}](\overline{e}), \qquad i = 1,2,3.$$

The single patch building blocks $\mathbf{s}_i(\mathbf{u})$ and the normal $\mathbf{n}$ take the form of operators; $\mathbf{p}$ represents positional information, $\overline{\mathbf{e}}$ represents the union of all three edges, and each argument $\mathbf{e}_i$ is expressed in terms of barycentric coordinates for the corresponding edge.

Using the properties (i)–(v), it is possible to derive the following interpolation theorem for discrete data (see also Nielson [14]).

THEOREM 6.2.1. *The convex combination* $\mathbf{s}(\mathbf{u}) = \sum_{i=1}^{3} w_i(\mathbf{u}) * \mathbf{s_i}(\mathbf{u})$ *interpolates to all three boundary curves and the patch normal on the whole boundary provided the conditions* (i) – (v) *are satisfied.*

Proof. (a) Positional Interpolation. $\mathbf{s}_i[\mathbf{p}](\overline{e}) = \mathbf{p}(\overline{e})$ and $\sum_{i=1}^{3} w_i(\overline{e}) = 1$; therefore $\mathbf{s}[\mathbf{p}](\overline{e}) = \mathbf{p}(\overline{e})$.

(b) Normal Interpolation. To show interpolation of the boundary normals, one calculates two nonparallel tangent vectors in an arbitrary boundary point, calculates the cross product of those (determines normal at that point), and shows that this cross product (vector) is parallel to the prescribed patch normal. Let the directions in which tangent vectors are computed be $\mathbf{d}_1 = (-1,1,0)$, $\mathbf{d}_2 = (0,-1,1)$, and $\mathbf{d}_3 = (1,0,-1)$.

$\mathbf{D}_{\mathbf{d}_i}$ is the vector-valued derivative operator to calculate tangent vectors in direction $\mathbf{d}_i$. Boldface letters will denote vector-valued operators. Using the product rule one obtains, considering the properties (ii) and (iv):

$$\begin{aligned}
\mathbf{D}_{\mathbf{d}_i}\mathbf{s}[\mathbf{p}](e_i) &= w_1(e_i) * \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_1[\mathbf{p}](e_i) + D_{\mathbf{d}_i}w_1(e_i) * \mathbf{s}_1[\mathbf{p}](e_i) \\
&\quad + w_2(e_i) * \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_2[\mathbf{p}](e_i) + D_{\mathbf{d}_i}w_2(e_i) * \mathbf{s}_2[\mathbf{p}](e_i) \\
&\quad + w_3(e_i) * \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_3[\mathbf{p}](e_i) + D_{\mathbf{d}_i}w_3(e_i) * \mathbf{s}_3[\mathbf{p}](e_i) \\
&= \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_i[\mathbf{p}](e_i) + \mathbf{p}(e_i) * (D_{\mathbf{d}_i}(w_1(e_i) + w_2(e_i) + w_3(e_i))) \\
&= \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_i[\mathbf{p}](e_i)
\end{aligned}$$

Using the result from (a), the tangent vector along a boundary curve is given by

$$\mathbf{D}_{\mathbf{d}_{i+1}}\mathbf{s}_i[\mathbf{p}](e_i) = \mathbf{D}_{\mathbf{d}_{i+1}}\mathbf{s}[\mathbf{p}](e_i)$$

$i = 1,2,3$ and $\mathbf{d}_4 = \mathbf{d}_1$. The patch normal along a boundary is then determined by the following cross product (using property (v)) choosing the arbitrary directions $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$ (without loss of generality) :

$$\begin{aligned}
\mathbf{n}[\mathbf{p}](e_i) &= \mathbf{n}[\mathbf{s}_i[\mathbf{p}]](e_i) \\
&= \mathbf{D}_{\mathbf{d}_i}\mathbf{s}_i[\mathbf{p}](e_i) \times \mathbf{D}_{\mathbf{d}_{i+1}}\mathbf{s}_i[\mathbf{p}](e_i) \\
&= \mathbf{D}_{\mathbf{d}_i}\mathbf{s}[\mathbf{p}](e_i) \times \mathbf{D}_{\mathbf{d}_{i+1}}\mathbf{s}[\mathbf{p}](e_i) \\
&= \mathbf{n}[\mathbf{s}[\mathbf{p}]](e_i)
\end{aligned}$$

which proves normal interpolation along the boundaries.

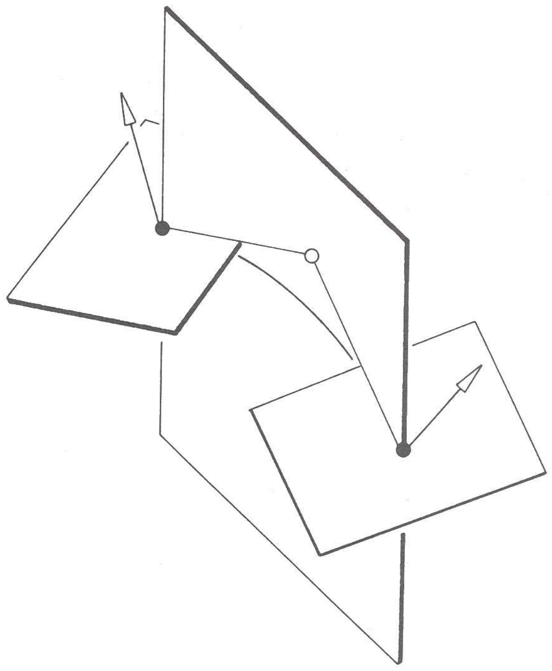

FIG. 6.1. *Obtaining a plane for a conic.*

6.3. The Planar Conic Curve Scheme

We will now describe how to make use of a planar curve scheme when two points and two unit outward normal vectors associated with those points are given in three-dimensional space (denoted as $\mathbf{b}_0$, $\mathbf{b}_3$, $\mathbf{n}_0$, and $\mathbf{n}_3$). For simplicity, we assume that points and normals are obtained from a convex surface. This constraint will be removed later. Generally, it is not possible to use a *planar* curve scheme for this situation, because *two points* and *one normal vector* determine a plane for the curve already. Therefore, one has to derive a plane for the curve and unit tangent vectors for its end points in a first step.

Referring to Fig. 6.1, our construction proceeds as follows:

1. Define a plane $\mathbf{P}$ through $\mathbf{b}_0$ and $\mathbf{b}_3$ that will contain the desired curve. This plane is specified by the requirement that the vector $(\frac{1}{2}\mathbf{n}_0 + \frac{1}{2}\mathbf{n}_3)$ lies in it (special care must be given to the case $\mathbf{n}_0 = -\mathbf{n}_3$).

2. Construct the intersection of the conic plane $\mathbf{P}$ with the tangent plane $\mathbf{P}_0$ at $\mathbf{b}_0$ and of $\mathbf{P}$ with the tangent plane $\mathbf{P}_3$ at $\mathbf{b}_3$. Each of the straight lines thus obtained defines the tangent of the desired curve at $\mathbf{b}_0$ and $\mathbf{b}_3$, respectively. The cross product between the normal to $\mathbf{P}$ and the two given normal vectors at the end points of the curve define (unit) tangent vectors for the end points, denoted by $\mathbf{t}_1$ and $\mathbf{t}_2$.

3. Assuming the data imply a convex curve, we describe the resulting conic in

terms of a degree-elevated rational Bézier curve of degree 3:

$$\mathbf{c}(t) = \frac{\sum_{i=0}^{3} \omega_i \mathbf{b}_i B_i^3(t)}{\sum_{i=0}^{3} \omega_i B_i^3(t)}, \tag{6.2}$$

where $t \in [0,1]$, $\omega_0 = \omega_3 = 1$, $\omega_1 = \omega_2 = \omega$ and $B_i^3(t) = \binom{3}{i}(1-t)^{3-i}t^i$, $i = 0 \cdots 3$. Referring to Fig. 6.3 the interior Bézier points lie on a line parallel to the line given by $\mathbf{b}_0$ and $\mathbf{b}_3$. Using the *law of sines* we have

$$D_1 = \frac{\sin\beta}{\sin\gamma} D.$$

Degree elevation requires the following ratio to hold:

$$\frac{d_1}{D_1 - d_1} = 2\omega.$$

Therefore,

$$d_1 = \frac{2\omega}{1+2\omega} \frac{\sin\beta}{\sin\gamma} D.$$

We obtain

$$\mathbf{b}_1 = \mathbf{b}_0 + d_1 \mathbf{t}_1.$$

The same construction is carried out for $\mathbf{b}_2$.

4. The weight ω should be chosen automatically such that (i) d_1 *is finite* for γ approaching 0 (parallel tangents at end points) and (ii) a *circular arc* (see [9]) is obtained for the case $\alpha = \beta$. The second goal is easily achieved by setting

$$\omega = \sin\frac{\gamma}{2} = \cos\alpha.$$

This choice also guarantees

$$\lim_{\gamma\to 0} d_1 = \lim_{\gamma\to 0} \frac{2\sin\frac{\gamma}{2}}{1+2\sin\frac{\gamma}{2}} \frac{\sin\beta}{\sin\gamma} D = \lim_{\gamma\to 0} \frac{2\frac{\gamma}{2}}{1+2\frac{\gamma}{2}} \frac{\sin\beta}{\gamma} D = D\sin\beta.$$

Therefore, d_1 is always finite.

In addition to the point and normal information, three *weights* associated with the edges $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ of a triangle can also be prescribed. In order to avoid consistency problems between triangles and to reduce input information, all weights associated with interior Bézier points should be chosen automatically as proposed above. Figure 6.2 illustrates the degree elevation process for curves.

6.4. Generating the Patch Building Blocks

Nielson [14] suggests computing a point $\mathbf{s}_i(\mathbf{u})$, $i = 1,2,3$, for given parameter values u_1, u_2, and u_3 on a particular building block by generating two separate cubic curves in Hermite form. The first cubic curve is associated with the edge

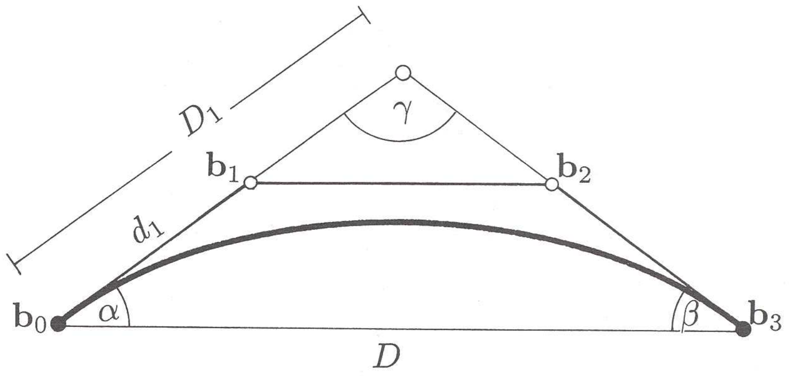

FIG. 6.2. *Generating conic along edge e_i.*

$\mathbf{e}_i$ of the domain triangle; it interpolates the vertices associated with this edge and the prescribed tangent vectors at those vertices. In a first step, this cubic is evaluated to obtain a point on the boundary curve along edge $\mathbf{e}_i$. The second cubic to be constructed is the result of blending from the vertex $\mathbf{v}_i$ to the point on the cubic along $\mathbf{e}_i$ just obtained. This curve then interpolates the vertex $\mathbf{v}_i$, the point on the cubic along $\mathbf{e}_i$, and the two tangent vectors prescribed for these points. In a second step, this cubic is evaluated to finally get the point on this building block of the surface patch.

In the new method presented here, *degree-elevated conics* will be used instead of cubic Hermite polynomials. The use of Hermite polynomials for constructing planar parametric curves can lead to undesired inflection points or even loops in the curve depending on the length of the prescribed tangent vectors at the two end points. This can no longer happen when degree-elevated conics are used instead. We will no show how to use the planar curve scheme from the last section in order to calculate points on the building blocks $\mathbf{s}_i(\mathbf{u})$.

(i) *Curve scheme for the boundary curves along triangle edges.* Using the planar curve scheme based on degree elevated conics, it is easy to generate the three boundary curves $\mathbf{c}_1(t)$, $\mathbf{c}_2(t)$, and $\mathbf{c}_3(t)$. When concerned with evaluating the patch building block $\mathbf{s}_1(\mathbf{u})$ at $\mathbf{u} = (u_1, u_2, u_3)$, one has to construct the curve $\mathbf{c}_1(t)$ with the parameter $t = u_3/(1-u_1) \in [0,1]$ along edge $\mathbf{e}_1$ first. The input for the conic scheme are the vertices $\mathbf{v}_2$ and $\mathbf{v}_3$, the normals $\mathbf{n}_2$ and $\mathbf{n}_3$, and the parameter t. The use of the conic scheme then yields a point on the patch boundary. The process for determining boundary points for the three patch building blocks always follows this general principle.

(ii) *Surface scheme for a point on* $\mathbf{s}_i(\mathbf{u})$. Having computed a point $\mathbf{c}_i(t)$ $(i = 1,2,3)$ on a boundary curve, the next question is how to estimate the *surface normal* of the final patch at a particular point on the curve $\mathbf{c}_i(t)$. It is clear that the surface normal $\mathbf{n}_i^S(t)$ along the boundary curve $\mathbf{c}_i(t)$ must also be perpendicular to this curve itself. This can be written as

$$\mathbf{n}_i^S \dot{\mathbf{c}}_i = 0,$$

$i = 1, 2, 3$, where $\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_i(t)$ denotes the tangent vector of the conic at the parameter value t. If we further require that

$$\mathbf{n}_i^S \mathbf{n}_i^C = \gamma(t), \tag{6.3}$$

where $\mathbf{n}_i^C$ denotes the unit normal vector to the conic in its plane at the parameter value t, then $\mathbf{n}_i^C$ is determined. We choose to select $\gamma(t)$ according to the following geometric interpretation: at $t = 0$,

$$\mathbf{n}_i^S(0)\mathbf{n}_i^C(0) = \gamma(0)$$

denotes the cosine of the angle formed by the (given) surface normal $\mathbf{n}_i^S(0)$ and the conic normal $\mathbf{n}_i^C(0)$. At $t = 1$, $\gamma(1)$ has the analogous interpretation. If we set

$$\gamma(t) = (1 - t)\gamma(0) + t\gamma(1),$$

then (6.3) assures that the cosine of the angle formed by $\mathbf{n}_i^S$ and $\mathbf{n}_i^C$ varies linearly along an edge.

This process guarantees that the surface normals derived by this method agree with the given ones at the vertices. Moreover, the surface normals along this edge will be the same for this triangular surface patch as for a potential neighbor sharing edge $\mathbf{e}_i$. Together with the fact that the choice of $\mathbf{n}_i^S(t)$ solely depends on given information along edge $\mathbf{e}_i$ and the special weight functions $w_i(\mathbf{u})$, tangent plane continuity between the actual patch and a neighbor along $\mathbf{e}_i$ is assured. We emphasize that our choice of boundary surface normals does consider the boundary curve — it is not simple linear interpolation of the given normal vectors at the two vertices. The process for determining the surface normal along a boundary conic $\mathbf{c}_i(t)$ is illustrated in Fig. 6.3.

Using the idea of radial projectors as introduced in Nielson [14], we next blend from a vertex $\mathbf{v}_1$ to the edge $\mathbf{e}_1$ with its corresponding point on the curve $\mathbf{c}_1(t)$. The generation of a point on a curve emanating from vertex $\mathbf{v}_1$ and ending at the previously computed point $\mathbf{c}_1(t)$ follows the same rules as the generation of the boundary conic $\mathbf{c}_1(t)$, $t = u_3/(1 - u_1) \in [0, 1]$. Therefore, only the input information for the planar conic scheme for calculating a point on building block $\mathbf{s}_1(\mathbf{u})$ will be given.

To obtain a point of the patch building block $\mathbf{s}_1(\mathbf{u})$ at $\mathbf{u} = (u_1, u_2, u_3)$, one has to construct a curve $\mathbf{c}(\bar{t})$ with $\bar{t} = (1 - u_1) \in [0, 1]$. The input data for the curve scheme are the vertices $\mathbf{v}_1$ and $\mathbf{c}_1(t)$, the normals $\mathbf{n}_1$, and the constructed surface normal $\mathbf{n}_1^S(t)$ along edge $\mathbf{e}_1$. Again, the use of the conic scheme with the parameter $\bar{t}$ yields a point of this building block of the whole patch. The final evaluation of a point $\mathbf{s}_1(\mathbf{u})$ is shown in Fig. 6.4.

Repeating this process for all three building blocks $\mathbf{s}_1(\mathbf{u})$, $\mathbf{s}_2(\mathbf{u})$, and $\mathbf{s}_3(\mathbf{u})$, one finally obtains a point $\mathbf{s}(\mathbf{u})$ on the surface. The weights for the middle Bézier point of each conic may be interpreted as a tension parameter.

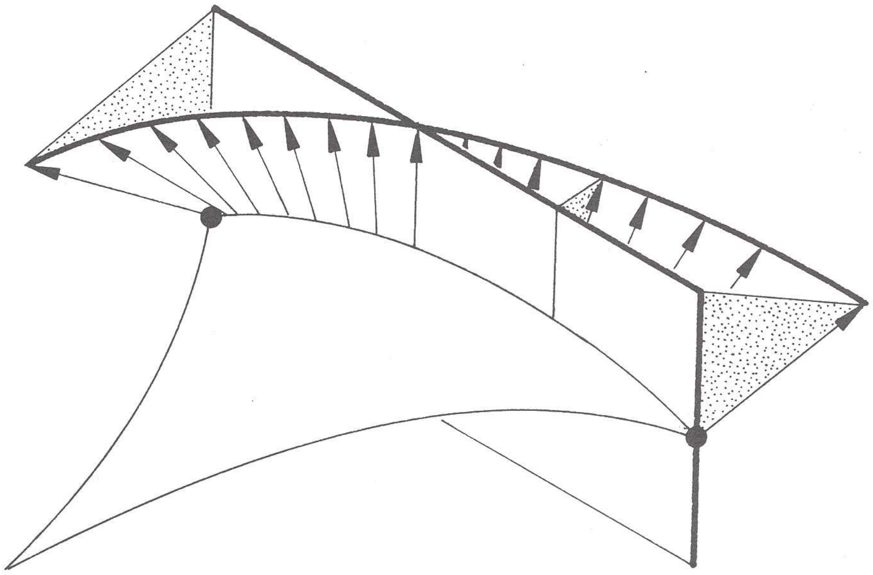

FIG. 6.3. *Generating surface normal along edge e_i.*

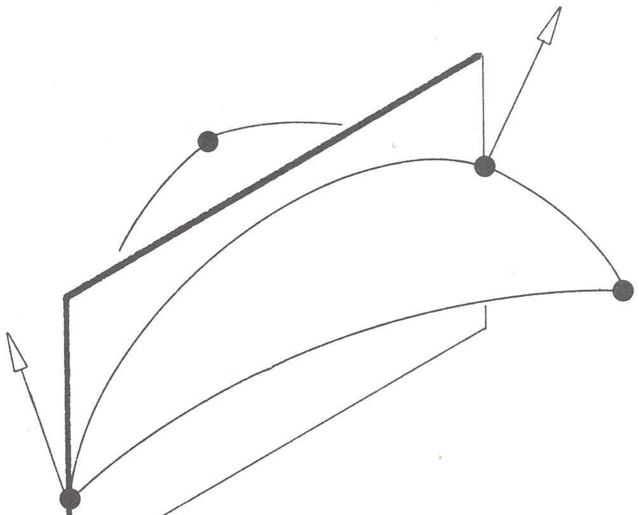

FIG. 6.4. *Evaluating surface building block $\mathbf{s}_i(\mathbf{u})$.*

6.5. Curve Scheme for Nonconvex Data

So far, only convex data configurations have been considered so that it is possible to make use of (degree-elevated) conics. Ball [1]–[3] introduced *generalized conics* as rational curves of degree 3. His scheme deals with data that do or do not imply curves with an inflection point. First, a criterion has to be presented that allows us to determine whether a conic scheme can be used for a planar data configuration (two vertices and two tangent vectors) or not.

The line through two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ subdivides the curve plane (specified by these two vertices and the normal vector of the plane) into two half spaces. If the two tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ do not point into the same half space, we have a *convex configuration*; otherwise we have a *nonconvex configuration.* If the data are convex, the planar scheme for degree-elevated conics can be used as described above. In the case of a nonconvex configuration, the two interior Bézier points for our curve scheme must be constructed in a way that a rational curve of degree three *with* an inflection point is obtained.

Assuming the tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ are directed into the *left* half space (with respect to the axis from $\mathbf{v}_1$ to $\mathbf{v}_2$), the tangent line given by $\mathbf{t}_2$ is reflected, the reflection being against the axis given by the end points. The construction for both interior Bézier points is then performed in the *left* half space as if dealing with a convex data configuration. Having computed both interior Bézier points in the regular fashion, $\mathbf{b}_2$ has to be reflected back in order to obtain the correct tangent direction at $\mathbf{v}_2$. The analogous construction has to be carried out in the case where both given tangent vectors are pointing into the *right* half space. The four different possibilities for generating the interior Bézier points are illustrated in Fig. 6.5.

Using the concept of degree-elevated conics allows us to handle both convex and nonconvex data. Continuity *inside* a single patch building block is guaranteed since the construction of the interior Bézier points is continuous with respect to the involved angles. If a given data set consists of a high percentage of convex data on certain triangles, the cost for determining a rational curve of degree 3 for nonconvex data is not very important. In the case that all given data belong to a convex surface, the scheme for degree-elevated conics can be used everywhere and the more complicated scheme for reflection of end tangents does not have to be considered.

6.6. Conclusions and Examples

A parametric triangular patch has been presented that makes use of given convex data, but is easily extended to nonconvex data by using Ball's idea of reflecting tangents and using rational curves of degree 3. For each triangle several degrees of freedom still remain — the weights for the middle Bézier points for all three boundary conics and for all the conics used in the radial blending from a vertex to the opposite boundary curve. The smaller these weights become, the closer the patch gets to its corresponding domain triangle

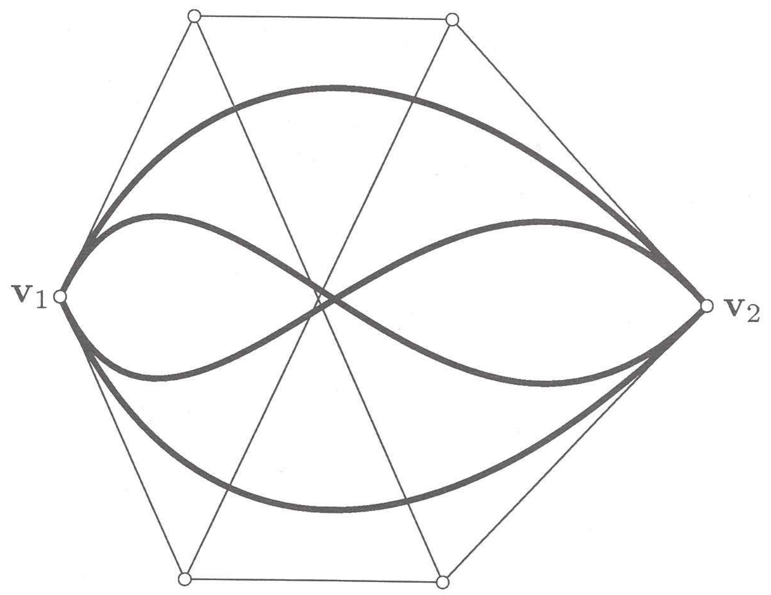

FIG. 6.5. *Construction of interior Bézier points for all data configurations.*

given by its three vertices. Choosing these weights automatically as proposed above yields circular arcs when the data configuration implies them. In this case the resulting surface will approximate the desired sphere-like surface quite well. However, the presented scheme does *not* have spherical precision even if the given data come from a sphere. This is a consequence of the used convex combination: each building block has spherical precision, but generates a *different* point on the sphere; therefore, when these points (all on a sphere!) are combined in a convex combination, the resulting point lies in the plane determined by the three points on the building blocks but no longer on the sphere.

Using degree-elevated conics instead of parametric cubic curves in a plane guarantees that one does not obtain inflection points or loops unless the prescribed normals at the two end points of a curve imply an inflection point. In Fig. 6.6 three different surfaces are shown for the case where four points and four unit normal vectors from a unit sphere are given. The first surface is obtained by choosing the weights for the interior Bézier points automatically (maximal distance to unit sphere approximately 0.01); the other two surfaces have successively lower weights associated with the interior Bézier points.

Acknowledgments

The work presented here was supported by the U.S. Department of Energy under contract DE-FG02-87ER25041 to Arizona State University and by National Science Foundation grant DCR-8502858. We appreciate the comments of the referees, which led us to develop the concept of degree-elevated conics.

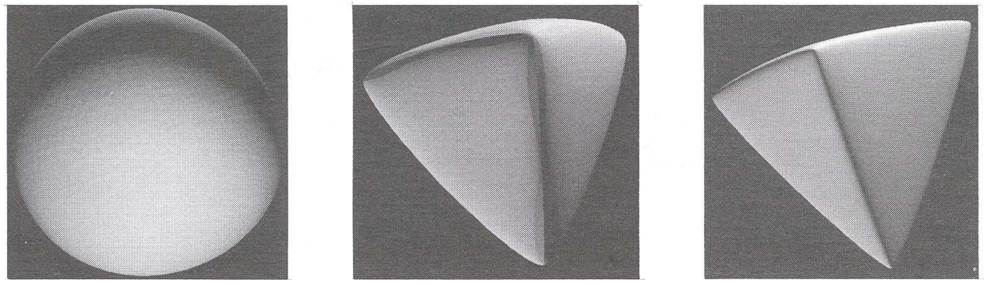

FIG. 6.6. *Surfaces obtained from spherical data.*

References

[1] A.A. Ball, *CONSURF* I, *Introduction of the conic lofting tile*, Comput. Aided Des., 6 (1974), pp. 243–249.

[2] ______, *CONSURF* II, *Description of the algorithms*, Comput. Aided Des. 7 (1975), pp. 237–242.

[3] ______ *CONSURF* III, 9 (1977), pp. 9–12.

[4] R.E. Barnhill, *Surfaces in computer aided geometric design: A survey with new results*, Comput. Aided Geom. Des., 1–3 (1985), pp. 1–17.

[5] R.E. Barnhill, G. Birkhoff, and W.J. Gordon, *Smooth interpolation in triangles*, J. Approx. Theory, 8 (1973), pp. 114–128.

[6] W. Boehm, G. Farin, and J. Kahmann, *A survey of curve and surface methods in CAGD*, Comput. Aided Geom. Des., 1 (1984), pp. 1–60.

[7] W. Boehm, *On cubics, A survey*, Computer Graphics and Image Processing, 19 (1982), pp. 201–226.

[8] G. Farin, *Triangular Bernstein–Bézier patches*, Comput. Aided Geom. Des., 3 (1986), pp. 83–127.

[9] ______, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, New York (1988).

[10] H. Hagen and H. Pottmann, *Curvature continuous triangular interpolants*, in Mathematical Methods in Computer Aided Geometric Design, T. Lyche and L.L. Schumaker, eds., Academic Press, New York, 1989, pp. 373–384.

[11] B. Hamann, *Visualization techniques for the representation of three dimensional data sets* (in German), CAD Computergraphik, to appear.

[12] J. Hoschek and D. Lasser, *Fundamentals of Geometric Data Processing* (in German), Teubner, Stuttgart, 1989.

[13] G.M. Nielson, *The side-vertex method for interpolation in triangles*, J. Approx. Theory, 25 (1979), pp. 318–336.

[14] ______, *A transfinite, visually continuous, triangular interpolant*, in Geometric Modeling: Algorithms and New Trends, G.E. Farin, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 235–246.

[15] G.M. Nielson and B. Hamann, *Techniques for the Interactive Visualization of Volumetric Data*, IEEE Conference, Visualization '90, 1990.

[16] B.R. Piper, *Visually smooth interpolation with triangular Bézier patches*, in Geometric Modeling: Algorithms and New Trends, G.E. Farin, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 221–233.

[17] V. Pratt, *Techniques for conic splines*, in Proceedings of SIGGRAPH 1985, ACM, New York, 1985, pp. 151–159.