Chapter 20

Surfaces with Arbitrary Topology

The surfaces that we have met so far are best suited for shapes that are the image of some part of the plane – of a rectangle in the case of B-spline or Bézier surfaces, of a triangulated region in the case of composite Bézier triangles. This limits the *topology* of these surfaces; for example, it is not possible to construct even a sphere without introducing degenerate patches while using a C^1 map of a part of the plane. Many shapes that have the topology of a planar region are too complex to model with one tensor product surface; just imagine to model a glove that way. The complexity issue may be tackled using the approach of hierarchical B-splines as proposed by Forsey and Bartels' [**?**].

In this chapter, we will investigate methods that are suited for the construction of shapes of arbitrary complexity and/or topology. We can only present a brief selection of methods; more literature on the topic: [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. The basic concepts of surface topology are nicely explained in [**?**].

20.1 Recursive Subdivision Curves

We already encountered Chaikin's algorithm in Section **??**. Starting with a polygon, it produces a sequence of refined polygons which ultimately converge to the uniform quadratic B-spline curve defined by the initial polygon.

This principle may be applied to higher degree uniform splines; we discuss the cubic case here. If we double a uniform knot sequence by inserting the midpoint of every knot, new control points $\mathbf{d}_i^1$ are generated from the existing ones by setting

$$\begin{aligned}\mathbf{d}_{2i}^{(1)} &= \frac{1}{8}\mathbf{d}_{i-1} + \frac{3}{4}\mathbf{d}_i + \frac{1}{8}\mathbf{d}_{i+1}\\ \mathbf{d}_{2i+1}^{(1)} &= \frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}_{i+1}\end{aligned} \tag{20.1}$$

See Figure 20.1 for an illustration. We may analyze the resulting curve by

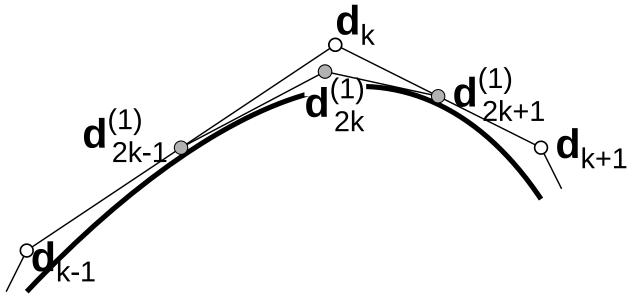

Figure 20.1: Subdivision curves: one step of cubic B-spline curve subdivision.

exploiting the fact that we are dealing with a cubic B-spline curve. However, such analysis may also be carried out without this knowledge. Let us investigate the limit of the sequence $\mathbf{d}_k, \mathbf{d}_{2k}^{(1)}, \mathbf{d}_{4k}^{(2)}, \ldots$ We may write the recursion in matrix form as

$$\begin{bmatrix}\mathbf{d}_{2k-1}^{(1)}\\ \mathbf{d}_{2k}^{(1)}\\ \mathbf{d}_{2k+1}^{(1)}\end{bmatrix} = \frac{1}{8}\begin{bmatrix}4 & 4 & 0\\ 1 & 6 & 1\\ 0 & 4 & 4\end{bmatrix}\begin{bmatrix}\mathbf{d}_{k-1}\\ \mathbf{d}_k\\ \mathbf{d}_{k+1}\end{bmatrix}$$

and abbreviate it as

$$\mathbf{D}^{(1)} = A\mathbf{D}. \tag{20.2}$$

The matrix A has eigenvalues $1, \frac{1}{2}, \frac{1}{4}$ and may be diagonalized[1] as

$$A = E\Lambda E^{-1},$$

where

$$E = \begin{bmatrix}1 & 1 & 2\\ 1 & 0 & -1\\ 1 & -1 & 2\end{bmatrix}, \quad E^{-1} = \frac{1}{6}\begin{bmatrix}1 & 4 & 1\\ 3 & 0 & -3\\ 1 & -2 & 1\end{bmatrix}, \quad \Lambda = \begin{bmatrix}1 & 0 & 0\\ 0 & \frac{1}{2} & 0\\ 0 & 0 & \frac{1}{4}\end{bmatrix}$$

[1]The process of diagonalization is a standard linear algebra procedure.

where E contains A's eigenvectors and the diagonal matrix Λ contains $A's$ eigenvalues.

For the next iteration, we have

$$\mathbf{D}^{(2)} = A\mathbf{D}^{(1)} = A^2\mathbf{D}^{(1)}. \tag{20.3}$$

Using our diagonalization for A, we get

$$A^2 = E\Lambda E^{-1}E^{-1}\Lambda E = E\Lambda^2 E^{-1}$$

and further

$$A^r = E\Lambda^r E^{-1}.$$

Taking the limit $r \to \infty$ yields

$$A^\infty = E\Lambda^\infty E^{-1}.$$

Since

$$\Lambda^\infty = \Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

this becomes

$$A^\infty = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 1 & 4 & 1 \\ 1 & 4 & 1 \end{bmatrix},$$

implying that all three points $\mathbf{d}_{k-1}, \mathbf{d}_k, \mathbf{d}_{k+1}$ converge to the same point $(\mathbf{d}_{k-1} + 4\mathbf{d}_k + \mathbf{d}_{k+1})/6$. This result is no surprise since we know we are dealing with B-spline curves. It does illustrate, however, a general principle which is ubiquitous in all of subdivision theory.

Subdivision may also be used to generate *interpolating* curves. The basic *four point* principle was developed by Dyn, Levin, and Gregory [**?**], [**?**]. Let a sequence of points $\mathbf{p}_i$ be given; then successively construct new sequences by setting

$$\begin{aligned} \mathbf{p}_{2i}^{(1)} &= \mathbf{p}_i, \\ \mathbf{p}_{2i+1}^{(1)} &= \frac{1}{16}[-\mathbf{p}_{i-1} + 9\mathbf{p}_i + 9\mathbf{p}_{i+1} - \mathbf{p}_{i+2}]. \end{aligned} \tag{20.4}$$

For an illustration, see Figure 20.2. The point $\mathbf{p}_{2i+1}^{(1)}$ is the result of applying cubic Lagrange interpolation at uniform knots, see [**?**].

At each level of the subdivision process, the points of the previous step are retained, this causes the scheme to interpolate to the initial set of points. The limit curve is C^1, but fails to be C^2.

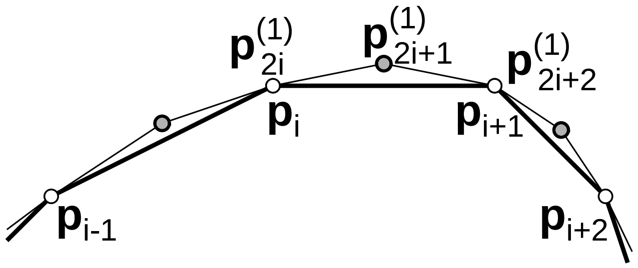

Figure 20.2: Subdivision curves: one step of the four point scheme.

20.2 Doo-Sabin Surfaces

The fundamental idea of this kind of surfaces goes back to Chaikin's algorithm, see section **??**. There, we started with a polygon, iteratively applied a refinement procedure to it, and observed that in the limit we ended up with a smooth curve. M. Sabin and D. Doo asked if this principle cannot be carried over to surfaces: start with a polyhedron, iteratively apply a refinement procedure to it, and see if maybe a smooth surface would result.

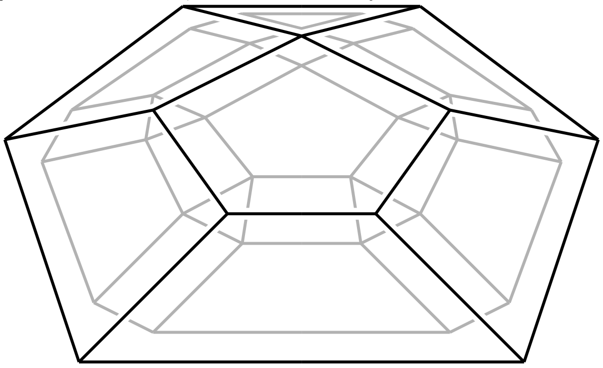

Figure 20.3: The Doo-Sabin algorithm: a new polyhedron is constructed by a refinement procedure.

They then came up with the following algorithm, illustrated in Fig. 20.3 and documented in [**?**]: Input: an arbitrary (open or closed) polyhedron with vertices $\mathbf{p}_i$. These vertices form (straight) edges and (not necessarily planar) faces, thus defining the topology of the polyhedron. The refinement step now becomes:

1. Find the centroid of the vertices of each face and the midpoints of each edge.
2. For each face, compute new vertices by averaging a vertex, its two adjacent edge midpoints, and the face centroid.
3. Construct a new polyhedron from these new vertices.
Step 3 needs some more explanation. The new polyhedron will have faces that are constructed according to three different rules:
3a The *F-faces* are found by connecting the midpoints of each original face,
3b The *E-faces* are found by considering any two original faces sharing a common edge: there are exactly four midpoints on the lines connecting each face centroid with the edge endpoints. These four points produce a four-sided face.
3c The *V-faces* are formed by considering all E-faces around an original vertex. They surround a face that is "centered" around that vertex.

As we keep repeating the algorithm, it produces mostly four-sided faces. The only non-four-sided faces are F-faces generated by those initial vertices whose valency[2] is not four. In fact, it is not hard to see that after the first step the valency of every new vertex is four. In this manner, large regions of the new polyhedra are covered with nets that have a tensor-product structure. Doo-Sabin surfaces are thus "mostly" biquadratic B-splines. See Figure 20.4 for an example.

Let us briefly analyze the structure of the Doo-Sabin algorithm. New face points $\mathbf{f}_i^{(1)}$ are created from previous ones in a linear fashion which lends itself to a matrix formulation. Let us consider a five-sided face such as the center face in Figure 20.3. We obtain

$$\begin{bmatrix} \mathbf{f}_1^{(1)} \\ \mathbf{f}_2^{(1)} \\ \mathbf{f}_3^{(1)} \\ \mathbf{f}_4^{(1)} \\ \mathbf{f}_5^{(1)} \end{bmatrix} = \frac{1}{8 \cdot 5} \begin{bmatrix} 22 & 7 & 2 & 2 & 7 \\ 7 & 22 & 7 & 2 & 2 \\ 2 & 7 & 22 & 7 & 2 \\ 2 & 2 & 7 & 22 & 7 \\ 7 & 2 & 2 & 7 & 22 \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \\ \mathbf{f}_4 \\ \mathbf{f}_5 \end{bmatrix} \tag{20.5}$$

We may write this in matrix form as

$$\mathbf{F}^{(1)} = A_5 \mathbf{F},$$

which becomes

$$\mathbf{F}^{(r+1)} = A_5^r \mathbf{F}$$

[2]The valency of a vertex is the number of edges emanating from it.

after r applications of the algorithm. In the general case, we have the same structure but have to replace A_5 by a matrix A_n for faces with n edges.

As r increases, the behavior of the subdivision process depends crucially on the eigenvalues of A_n since we may write A_n^r as

$$A_n^r = E_n \Lambda_n^r E_n^{-1}$$

following the approach of Section **??**. Since all rows of n sum to unity, one eigenvalue is $\lambda_1 = 1$. The remaining ones are all real since A is symmetric and are all between 0 and 1 since each $\mathbf{F}^{(k)}$ is contained in the convex hull of $\mathbf{F}^{(k-1)}$. An exact analysis of this process is fairly involved and is omitted here. It reveals, however, that Doo-Sabin surfaces are G^1 everywhere. For more details, see [**?**] or [**?**], [**?**], [**?**]. The matrices A_n are *circulant*, and their eigenstructure has been studied in detail by P. Davis [**?**]. A matrix is circulant if each row may be obtained from the previous row by a "right shift."

Figure 20.4 gives an example of several steps of the algorithm.

Figure 20.4: The Doo-Sabin algorithm: an example.

As a rather trivial observation, Doo-Sabin surfaces have the convex hull and local control properties, and their construction is affinely invariant. But they also do not need an underlying parametrization, which makes them more geometric than tensor product B-spline surfaces. A drawback of this nice feature is the problem of point evaluation: while we can evaluate as many points as close to the surface as we like, computation of just one point is not trivial.

One set of points on the surface is easy to identify: at every level of subdivision, the centroid of any face will be on the final surface. For a proof, we observe that any face will produce a sequence of F-faces converging to its centroid. In the limit, the centroid is thus on the surface.

20.3 Catmull-Clark Subdivision

The same issue of *Computer Aided Design* that included the Doo-Sabin algorithm also contained a competing method, invented by E. Catmull and J.

Clark[3], see [**?**]. While Doo-Sabin surfaces are a generalization of biquadratic B-splines to arbitrary topology, Catmull-Clark surfaces generalize bicubic B-spline surfaces to arbitrary topology.

These are the cubic analog to Doo-Sabin surfaces. We start with a polygonal mesh $\mathcal{M}^0$ consisting of vertices $\mathbf{v}_k^0$. We iteratively refine the mesh, resulting in finer and finer meshes $\mathcal{M}^i$, consisting of vertices $\mathbf{v}_k^i$. Each refinement step can be described by explaining what happens locally for the points adjacent to a vertex $\mathbf{v}^i$:

1) Form face points $\mathbf{f}_j^{i+1}$: for each face in the mesh, find the centroid of its vertices.

2) Form edge points $\mathbf{e}_j^{i+1}$: for each edge in the mesh, average the edge's endpoints and the two face points on either side of the edge.

3) Form a new vertex point $\mathbf{v}^{i+1}$: assuming there are n faces around $\mathbf{v}^i$, it is computed as follows:

$$\mathbf{v}^{i+1} = \mathbf{v}^i + \frac{1}{n}\sum_{j=1}^{n}(\mathbf{e}_j^i - \mathbf{v}) + \frac{1}{n}\sum_{j=1}^{n}(\mathbf{f}_j^{i+1} - \mathbf{v}) \tag{20.6}$$

4) Form new faces. Each new face consists of a loop of the form

$$\mathbf{f}^{i+1} - -\mathbf{e}^i - -\mathbf{v}^{i+1} - -\mathbf{e}^i - -\mathbf{f}^{i+1}$$

where the two $\mathbf{f}^{i+1}$'s refer to the same face point.

Note that all faces after the first level will be 4-sided. An example is shown in Figure **??**.

For the example of a vertex with valency $n = 4$, the relationship between new and old vertices may be expressed as a matrix equation:

$$\begin{bmatrix} \mathbf{v}^{i+1} \\ \mathbf{e}_1^{i+1} \\ \mathbf{e}_2^{i+1} \\ \mathbf{e}_4^{i+1} \\ \mathbf{e}_4^{i+1} \\ \mathbf{v}_1^{i+1} \\ \mathbf{v}_2^{i+1} \\ \mathbf{v}_3^{i+1} \\ \mathbf{v}_4^{i+1} \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 9 & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 6 & 6 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 6 & 1 & 6 & 1 & 0 & 1 & 1 & 0 & 0 \\ 6 & 0 & 1 & 6 & 1 & 0 & 1 & 1 & 0 \\ 6 & 1 & 0 & 1 & 6 & 0 & 0 & 1 & 1 \\ 4 & 4 & 4 & 0 & 0 & 4 & 0 & 0 & 0 \\ 4 & 0 & 4 & 4 & 0 & 0 & 4 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 & 0 & 0 & 4 & 0 \\ 4 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{v}^i \\ \mathbf{e}_1^i \\ \mathbf{e}_2^i \\ \mathbf{e}_4^i \\ \mathbf{e}_4^i \\ \mathbf{v}_1^i \\ \mathbf{v}_2^i \\ \mathbf{v}_3^i \\ \mathbf{v}_4^i \end{bmatrix}$$

[3]This is Jim Clark, founder of Silicon Graphics and Netscape.

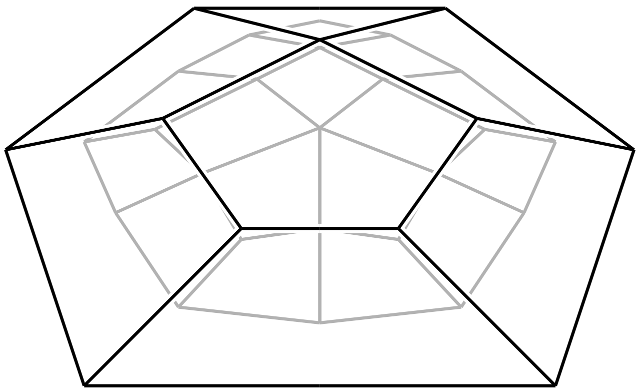

Figure 20.5: Catmull-Clark surfaces: the original control net (black) and one level of subdivision (gray).

For valencies other than four, similar matrix equations hold; see [**?**].

We may abbreviate (20.3) as

$$\mathbf{V}^{i+1} = A\mathbf{V}^i. \tag{20.7}$$

An equivalent form of (refeq:ccit) is given by

$$\mathbf{V}^{i+1} = A^i\mathbf{V}^1. \tag{20.8}$$

We note that the matrix A must be *stochastic*, i.e., the elements of each row must sum to one. This is a consequence of the fact that (20.12) represents a barycentric combination. We further note that our mesh refinement consists of taking *convex combinations* at each level: this implies that all meshes $\mathcal{M}^i$ lie in the convex hull of $\mathcal{M}^0$.

It follows that A has the dominant eigenvalue 1; the magnitudes of all other (possibly complex) eigenvalues are bounded by 1.

A careful eigenvalue analysis (see Ball and Storry [**?**]) can now be used to show that

$$\mathbf{v}^\infty = \frac{n^2\mathbf{v}^1 + 4\sum_j \mathbf{e}_j^1 + \sum_j \mathbf{f}_j^1}{n(n+5)}. \tag{20.9}$$

20.4 Midpoint Subdivision

This algorithm is simpler than either Catmull-Clark or Doo-Sabin and goes back to Peters and Reif [**?**] who call it “the simplest scheme.”

1) Form edge points $\mathbf{e}$: for each edge in the mesh, compute its midpoint.

2) Form faces of new level. There are two types: faces inscribed to the existing ones and faces whose vertices are the edge midpoints around old vertices. See Figure 20.6 for an illustration. In order to discuss convergence

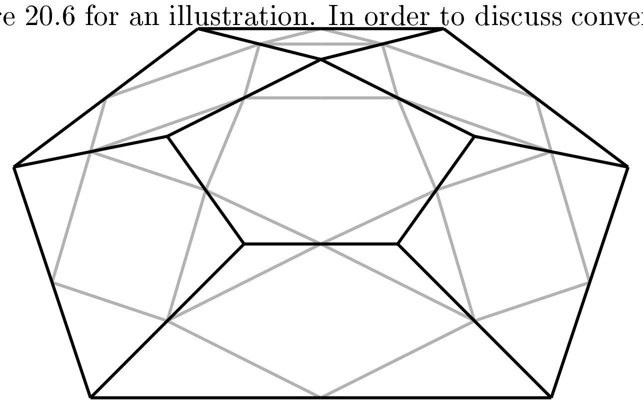

Figure 20.6: Midpoint subdivision: the original control net (black) and one level of subdivision (gray).

of the scheme, let $\mathbf{p}_1, \dots, \mathbf{p}_n$ be the vertices of a face. This face generates vertices $\mathbf{p}_i^1$ as edge midpoints

$$\begin{bmatrix} \mathbf{p}_1^1 \\ \vdots \\ \mathbf{p}_{n-1}^1 \\ \mathbf{p}_n^1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & & \\ & & \ddots & & \\ & & & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & & & & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_{n-1} \\ \mathbf{p}_n \end{bmatrix}. \tag{20.10}$$

We write this as

$$\mathbf{P}^1 = M\mathbf{P}.$$

Matrices of the form (20.10 are called *circulant.*

After k iterations, this becomes

$$\mathbf{P}^k = M^k\mathbf{P}.$$

The matrix M may be decomposed into a product

$$M = E\Lambda E^{-1}$$

where E is the matrix containing M's eigenvectors andΛ is the diagonal matrix containing M's eigenvalues. We then have, following the reasoning in Section **??**

$$M^k = E\Lambda^k E^{-1}.$$

The analysis of the limit process is thus closely related to the eigenstructure of M. A result from Davis [?] asserts that for any initial polygon $\mathbf{P}$, we obtain regular planar polygons in the limit. These will then be the tangent planes to our surface which thus is G^1.

20.5 Loop Subdivision

A triangle-based subdivision scheme was discussed by C. Loop [?]. Its input is a triangular mesh as discussed in Section **??**. It then successively refines this mesh, resulting in a smooth limit surface.

Loop subdivision proceeds as follows.

1) Form edge points $\mathbf{e}_j^{i+1}$: assuming that $\mathbf{v}_1^i$ and $\mathbf{v}_2^i$ are the endpoints of an edge in the mesh and that that $\mathbf{v}_3^i$ and $\mathbf{v}_4^i$ are the remaining vertices of the two triangles sharing the edge, set

$$\mathbf{e}_j^{i+1} = \frac{3}{8}(\mathbf{v}_1^i + \mathbf{v}_2^i) + \frac{1}{8}(\mathbf{v}_3^i + \mathbf{v}_4^i) \tag{20.11}$$

This process is easily visualized using a *mask*, shown in Figure 20.7. The shown coefficients have to be multiplied by a factor of 1/8 in order to work with barycentric combinations.

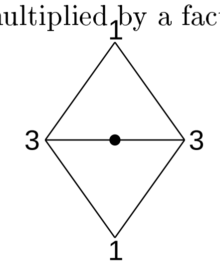

Figure 20.7: Loop subdivision: the edge point mask.

2) For each vertex $\mathbf{v}^i$ in the mesh, form a new vertex point $\mathbf{v}^{i+1}$. Assuming $\mathbf{v}^i$ has n neighbors $vv_1^i, \ldots, \mathbf{v}_n^i$ it is computed as follows:

$$\mathbf{v}^{i+1} = (1 - n\alpha)\mathbf{v}^i + \alpha \sum_{j=1}^{n} vv_j^i \tag{20.12}$$

where

$$\alpha = \frac{1}{n}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n})^2]$$

for $n > 3$ and $\alpha = \frac{3}{16}$ if $n = 3$.

3) Form new triangles.

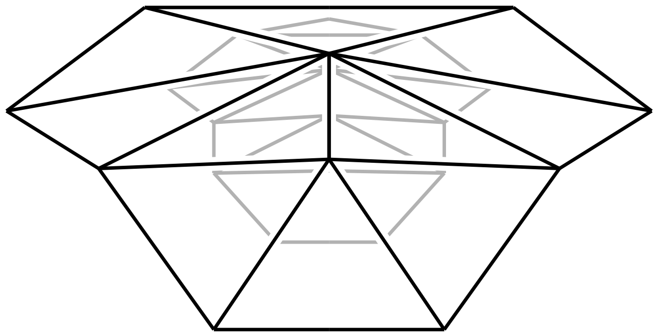

Figure 20.8: Loop subdivision: the original control net (black) and one level of subdivision (gray).

Figure 20.8 illustrates.

Now consider the mesh that is obtained after some number k of iterations. Select any vertex $\mathbf{v}$ of it. This vertex will converge to a point $\mathbf{v}^\infty$ on the limit surface. Using an eigenvalue analysis as mentioned above, one can show (see [**?**]) that this limit point is given by

$$\mathbf{v}^\infty = \frac{3 + 8\alpha(n-1)}{3 + 8n\alpha} + \frac{8\alpha}{3 + 8n\alpha} \sum_{j=1}^{n} \mathbf{v}_j \tag{20.13}$$

where the $\mathbf{v}_j$ are the neighbors of $\mathbf{v}$ in the mesh obtained after k iterations. For the special case $k = 0$, (20.13) gives the limit points corresponding the the original mesh vertices.

If all vertices in the mesh have valency six, then the resulting limit surface will be a collection of quartic Bézier triangles which form a C^2 surface over an equilateral triangulation of a simply connected region of a (domain) plane. For vertices with different valencies, the limit surface will be G^1. We note that closed surfaces cannot be formed using only points with valency six.

20.6 $\sqrt{3}$ Subdivision

This scheme was developed by Kobbelt [**?**] and was also considered by Sabin [**?**]. We start with a triangular mesh and then subdivide each triangle into three triangles by splitting it at its centroid. Next, all edges of the initial mesh are flipped – instead of joining the initial vertices, they now join adjacent centroids. Finally, each initial vertex $\mathbf{p}$ (having valency n) is replaced by a barycentric combination of its neighbors:

$$pp^{(1)} = (1-\alpha_n) + \alpha_n \mathbf{c}$$

where $\mathbf{c}$ is the average of $\mathbf{p}'$s neighbors. The value for α_n is set at

$$\alpha_n = \frac{1}{9}[4 - 2\cos(\frac{2\pi}{n})].$$

Each subdivision step rotates the directional structure of the triangular mesh; after two applications, the initial orientation is reestablished. Each original triangle has then given rise to nine new triangles. Following a rigorous study of the eigenstructure of the $\sqrt{3}$ operator, one can show that the resulting limit surface is G^2 except at extraordinary points, where it still is G^1.

20.7 Interpolating Subdivision Surfaces

When dealing with B-splines, we could start from a control polygon and design a curve, or we could start with data points and find an interpolating curve. We can also use recursive subdivision surfaces for interpolation. The idea goes back to Nasri[**?**] and to Lounsbery, Mann, DeRose [**?**]. The latter reference constructs interpolating Catmull-Clark surfaces, while Nasri constructs interpolating Doo-Sabin surfaces – we will focus on them.

So given is a polyhedron with vertices $\mathbf{p}_i$, and we wish to find another polyhedron with vertices $\mathbf{v}_i$ such that the resulting Doo-Sabin surfaces passes through the $\mathbf{p}_i$. Each of the (unknown) $\mathbf{v}_i$ will generate a V-face with vertices $\mathbf{q}_{i,k}; k = 1, \ldots, n_i$, where n_i is the valency of $\mathbf{v}_i$. We know that the centroids of these V-faces are on the surface, and we simply require them to be the given data poins:

$$\mathbf{p}_i = \frac{1}{n_i}(\mathbf{q}_{i,1} + \ldots + \mathbf{q}_{i,n_i}).$$

Note that the given points are not on the faces of the desired polyhedron, but rather on the V-faces obtained from it after one level of subdivision; see Fig. 20.9 for an illustration.

Since the relationship between the $\mathbf{q}_{i,k}$ and the unknowns $\mathbf{v}_i$ is known, we have a set of linear equations relating the given $\mathbf{p}_i$ to the unknown $\mathbf{v}_i$. For closed polyhedra, the number of equations equals the number of unknowns,

Figure 20.9: Interpolating Doo-Sabin surfaces: the intermediate points $\mathbf{q}_{i,k}$ are used to set up a linear system.

leading to a sparse linear system. For open polyhedra, the situation is more complicated; it is dealt with by Nasri [**?**].

Catmull-Clark subdivision surfaces may also be employed to generate surfaces passing through a prescribed set of data points. We can use (20.9) to generate a control mesh which interpolates to a set of known data points $\mathbf{v}^\infty$. All equations (20.9) form a linear system for the unknowns $\mathbf{v}^1$. This is a sparse system and thus quickly solved even if there are many (¿1000) data points.

It is mentioned in [**?**] that simply solving the least squares equations for the control points will result in "wiggly" surfaces. Adding shape equations as in Section **??** will overcome this problem.

In an similar fashion, equations (20.13) may be used to find a set of vertices $\mathbf{v}_i$ which will generate a Loop surface through the data points $\mathbf{p}_i$. All these equations form a sparse linear system for the $\mathbf{v}_i$. It may be solved using a sparse solver or by an iterative method. xxxx lit xxxxxx

A more direct way to interpolation, i.e., without the need to "invert" existing methods, is given by the *butterfly scheme*, due to Dyn, Gregory, and Levin [**?**]. Its structure is identical to Loop subdivision, but different weights are applied. At a given level in the refinement process, the vertex points are kept, and only new edge points are generated. Each new edge point is generated from the surrounding vertex points by a mask as shown in Figure 20.10. The (solid) edge point is obtained as a barycentric combination of nearby points; the coefficients in that combination are indicated in the Figure. They have to be multiplied by a factor of 1/16.

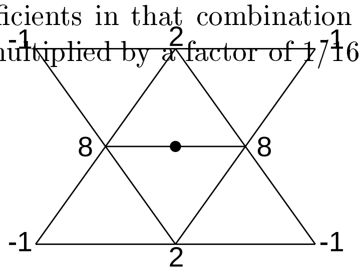

Figure 20.10: Butterfly subdivision: the edge point mask.

Just as in the four-point curve interpolation scheme of Section **??**, interpolation is assured since vertex points are kept.

20.8 Surface Splines

As we iterate through the Doo-Sabin algorithm, more and more of the surface is covered by biquadratic patches, just leaving the extraordinary regions. After two iterations, these are already nicely separated – they correspond to $s-$ sided regions. J. Peters had the idea of deviating from the Doo-Sabin procedure after two steps and to fill in these $s-$sided regions with a collection of bicubics, such that the overall surface is G^1, see [**?**] and also [**?**] or [**?**]. It is not equivalent to the Doo-Sabin surface, but it has the advantage of being a collection of standard patches without singularities.

The situation after two (or more) steps of the Doo-Sabin algorithm is shown in Figure 20.11. We have so far created the points marked by squares. The solid squares mark control points surrounding an $s-$sided region, while the open ones are control points of the network of biquadratic patches. We are going to cover the $s-$sided region by a collection of s bicubic patches, all having a center point $\mathbf{c}$ in common. This center point is simply the average of all solid control points surrounding the $s-$sided region, only partially shown in Figure 20.11.

Next, we have to degree elevate each quadratic boundary curve of the $s-$sided region and to subdivide it at its (parametric) midpoint. This gives two boundary curves of each bicubic patch.

Figure 20.11: Surface splines: the control points determining the Bézier points $\mathbf{b}_{ij}$ of one bicubic patch.

The "outer" two layers of each bicubic patch lie on bilinear patches as shown in Figure 20.11. Their computation[4] is illustrated for the four top left points in that figure:

$$\begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & \frac{5}{6} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} & \mathbf{b} \\ \mathbf{c} & \mathbf{d} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & \frac{5}{6} \end{bmatrix}$$

[4]We give a slightly simplified version of Peters' original development [**?**] here.

The remaining eight Bézier points along the outer patch boundary are found analogously.

The three remaining Bézier points $\mathbf{b}_{22}, \mathbf{b}_{23}, \mathbf{b}_{32}$ are determined as follows:

$$\mathbf{b}_{32}^{(i)} = \mathbf{b}_{23}^{(i+1)} = \frac{4}{3s} \sum_{j=1}^{s} \cos\left(j \frac{2\pi}{s}\right) \frac{\mathbf{d}^{(i+j)} + \mathbf{d}^{(i+j+1)}}{2},$$

where the superscript (i) refers to the $i-$th bicubic patch of the patch collection covering the $s-$sided region. Now all angles $\angle(\mathbf{b}_{33}, \mathbf{b}_{32}, \mathbf{b}_{23})$ are equal. The points $\mathbf{b}_{22}^{(i)}$ must be determined such that G^1 continuity is ensured around $\mathbf{b}_{33}$. Setting $c = \cos(2\pi/s)$ and $\mathbf{e}_i = (1-c)\mathbf{b}_{32}^{(i)} + c\mathbf{b}_{31}^{(i)}$, they are

$$\mathbf{b}_{22}^{(i)} = \begin{cases} -\sum_{j=1}^{s}(-1)^j \mathbf{e}_{i+j} & \text{if } n \text{ is odd,} \\ \frac{-2}{n}\sum_{j=1}^{s}(s-j)(-1)^j \mathbf{e}_{i+j} & \text{if } n \text{ is even.} \end{cases}$$

A surface spline is shown in Figure 20.12.

Figure 20.12: Surface splines: an example.

Surface splines may also be used to interpolate to a mesh of data points in the same way as Doo-Sabin surfaces did: after two steps of the Doo-Sabin algorithm, move those control points that surround given data points such that their average equals that data point. Then proceed as above, and interpolation is ensured.

20.9 Triangular Meshes

We encountered triangular meshes in Section 3.7 – there, we were dealing with 2D meshes. Keeping the same data structure, we may generalize 2D triangulations to 3D triangle meshes: these are surfaces consisting of a collection of triangles. The vertices $\mathbf{p}_i$ of the triangles are now 3D points. Triangle meshes are piecewise linear and thus are not smooth – although this defect may be overcome by using very many triangles. One example is provided by the "Digital Michelangelo" project, carried out by M. Levoy. The aim of the project is to digitally record sculptures by renaissance artist

Michelangelo. Each sculpture was digitized using a laser digitizer; the resulting "point cloud" was then triangulated. For some sculptures, around 2,000,000,000 triangles were needed, see [**?**].

Boundary edges are edges in the mesh which belong to one triangle only; all other edges, being part of two triangles, are called *interior edges.* A vertex is called *boundary vertex* if it is on a boundary edge; otherwise, it is called an *interior vertex.* The solid vertex in Figure 20.13 is an interior vertex.

A significant difference between 2D and 3D triangulations is their *topology.* Every 2D triangulation must have boundary vertices, but 3D triangulations do not have to have any. Consider the example of a tetrahedron, the simplest 3D triangle mesh. It has four vertices and four triangles, all of them are interior vertices.

20.10 Decimation

In any case, many triangulations are far too dense for an efficient representation of an object. What is called for then is a reduction in size, also known as *decimation.*[5] The goal is to remove as many points as possible, while still staying close to the initial triangulation.

A decimation algorithm will thus perform a check on every point and determine if it can be removed. If so, the point (and sometimes its neighbors) are removed and the resulting gap will be retriangulated. The process continues until no more data can be removed.

An important ingredient in many mesh algorithms is a test for determining if a mesh is flat in the vicinity of a vertex $\mathbf{p}$. There are several possibilities for this test; we discuss an angle-based one here.

For the following, we will need the concept of the *star* $\mathbf{p}_i^\star$ of a vertex $\mathbf{p}_i$: this is the set of all triangles having $\mathbf{p}_i$ as a vertex; see Fig. 20.13. Thus the star of a point is itself a triangle mesh.

Each of the triangles in $\mathbf{p}^\star$ forms an angle at $\mathbf{p}$. Call the sum of all these angles $\Sigma_\mathbf{p}$. If $\Sigma_\mathbf{p} = 360$, then all triangles are in one plane. Thus the quantity $360 - |\Sigma_\mathbf{p}|$ measures the nonplanarity of $\mathbf{p}^\star$. In our experience, the condition

$$|360 - \Sigma_\mathbf{p}| \leq 10 \tag{20.14}$$

is a good flatness indicator.

[5]The origin of the word can be traced to Roman times. When a legion deserted during battle, it was punished by having every tenth soldier killed (decimated).

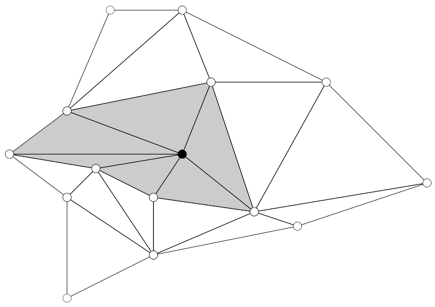

Figure 20.13: Star of a point: the shaded triangles form the star of the solid point.

The signed difference $360 - \Sigma_p p$ is also a good indicator for the shape of $\mathbf{p}^\star$. If it is positive, $\mathbf{p}^\star$ is tent-shaped; if it is negative, $\mathbf{p}^\star$ is saddle-shaped. These shapes are related to the *discrete Gaussian curvature* of $\mathbf{p}^\star$, see [**?**].

We now discuss a decimation method which is due to M. Lounsbery [**?**]. It checks if an edge in a triangulation can be safely removed. If so, the edge is replaced by a point on it, thereby replacing two points by one. Retriangulation is fairly trivial: Fig. 20.14 gives an example. A simple choice for the edge replacement point is the midpoint of the edge; more elaborate methods place it such that the shape of the resulting triangles is optimized.

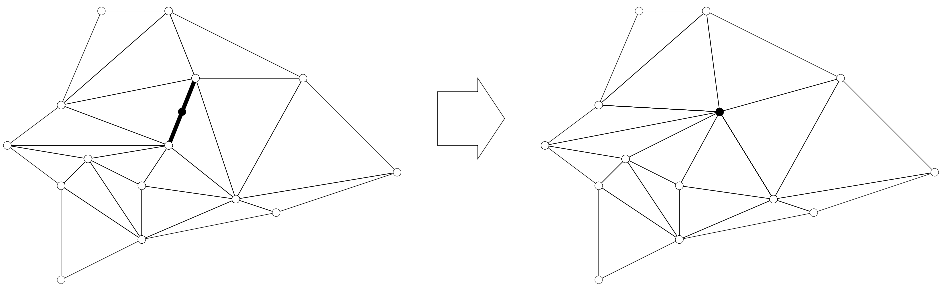

Figure 20.14: Edge collapse: an edge is marked for collapsing, left. It is then collapsed into a single point, right.

When can an edge be removed? We simply perform the above flatness test on the two edge vertices. If both meet the criterion (20.14), then the edge can be removed.

This decimation method lends itself well to a *multiresolution* analysis of a triangle mesh. If we keep track of each collapsed edge, we create a hierarchy of triangle meshes. We xxxxxxxxxxxxxxxxxxxx

20.11 Problems

*1. The Doo-Sabin recursion generates a sequence of F-faces for every face, in the limit converging to the centroid of the considered face. Show that the limiting F-faces are planar.

P1. Write a program to find an interpolating Doo-Sabin surface to the eight vertices of a cube.

P2. Modify S-patches as follows: the definition of "generalized barycentric coordinates" (18.27) may be replaced by $u_i = \frac{A_i}{\mathcal{A}}$, from (3.30).

P3. Given a tetrahedron, display the point set defined by (18.27).