February 1994

1–20,i–ix

# CURVATURE CONTINUOUS SPLINE SURFACES OVER IRREGULAR MESHES *

JORG PETERS †

**Abstract.** Concepts and techniques for the construction of smooth surfaces over irregular meshes are developed and made concrete by defining $C^2$-surface splines. These splines extend the B-spline paradigm for the construction of parametric piecewise polynomial surfaces to control-meshes with non quadrilateral cells and more or fewer than four cells meeting at a point. Mesh points serve as control points and are locally averaged to obtain a Bernstein-Bézier representation which in turn defines surface points as averages.

**Key words.** $C^2$ surface, corner cutting, box splines, blending, geometric continuity, spline mesh, free-form surface modeling, symbolic generation of constraints.

**AMS subject classifications.** 65D17,10,07 65Y25 68U07,05,

## 1. Introduction.

A single tensor-product B-spline complex can only model a small subclass of surfaces that arise in geometric modeling, because every interior point of the B-spline control mesh must be surrounded by exactly four quadrilateral cells. Thus the surface pieces must form a regular, checker board arrangement that allows only deformations of the plane and deformations of the torus to be modeled without singularity. Even for such topologically restricted objects, the rigid structure prevents a natural modeling of frequent features such as suitcase corners or house corners, where three or five quadrilaterals meet. In practice, patch trimming and certain singular parametrizations are used to overcome the restrictions. However, by destroying the consistent B-spline framework, these techniques create a number of special cases and difficult problems such as desingularization and the need for smoothly joining trimmed surfaces.

Tangent plane continuous splines over irregular meshes defined in [Peters '93,95] overcome the rigid tensor-product frame work, allowing freedom both in the number of patches meeting at a vertex and the number of edges to a mesh cell while preserving such desirable properties of B-splines as a low degree polynomial or rational representation of maximal smoothness and a geometrically intuitive variation of the surface in terms of the coefficients as evidenced by a local convex hull property. The local convex hull property, i.e. the property that every point on the surface is a convex combination of the mesh points nearby, also implies that linear features, say parts of the input mesh, can be reproduced while shape handles in the form of blend ratios give direct control over blend radii. A list of properties of surface splines is collected at the end of this section.

This paper describes a $C^2$-surface representation that meets the criteria for surface splines. Inspired by the $C^2$ box-spline on the four direction mesh, the construction uses three-sided patches in Bernstein-Bézier form (see [Boehm, Farin, Kahmann '84], [Farin '90] and [de Boor, Höllig, Riemenschneider '94] for details on the Bernstein-Bézier form and box splines.) The patches have quartic boundary curves and are of total degree six except for three octic monomial terms. Symmetries in the mesh reduce

* This work was supported by NSF NYI grant CCR-9457806.

†Department of Computer Science, Purdue University, W-Lafayette IN 47907-1398 (jorg@cs.purdue.edu).

the patch degree. The three-sided patches may be replaced by four-sided, biquartic, and bicubic tensor-product patches where the mesh is regular. The analogue of the tensor-product construction in [Peters '93,95] has degree bi-six. For most meshes this will result in a $C^2$ surface; otherwise the $C^2$ error is minimized. A bi-eight construction guarantees curvature continuity for all mesh connectivities. The tensor-product patch constructions is sketched by specifying the reparametrizations that define the spline space. As with $C^1$-surface splines [Peters '93,95], shape handles in the form of blend ratios are associated with each edge allowing for local control of the change of the normal and the curvature across that edge.

To make the surface construction concrete, explicit formulas of the Bernstein-Bézier coefficients in terms of the control mesh points have been generated (cf. Appendix 2). Given the reparametrizations and a choice of representation of the nullspace of the vertex-enclosure difference equations (see Section 2 for the details) the task of generating the formulas has been delegated to a symbolic routine written in Maple. This setup allows experimenting with different algorithms and decreases the likelihood of errors because the formulas are directly output by the routine and read by the C-language program that generates the surfaces. The formulas in Appendix 2 are an intermediate translation table when generating surface points from the high-level surface-spline control points.

**Related work.** Building on work by [Sabin '83] and [Goodman '91], [Höllig, Mögerle '89] pioneered the idea of geometrically continuous spline spaces. Explicit representations of such G-spline surfaces in terms of say the Bernstein-Bézier form require the solution of a large linear, irregularly sparse system of equations to match data. Newer work ([Piah '91], [Lee, Majid '91], [Peters '94], [Peters '93] and [Reif '93]), improves by exhibiting explicit formulas for the coefficients of the surface patches in terms of the input mesh. Still, the size of the support of these formulas makes it non trivial to predict the shape of the resulting surface. Even though a number of subdivision algorithms are known to generate tangent continuous surfaces ([Sabin '76], [Doo '78], [Catmull '78], [Loop '87], [Dyn, Levin, Liu '92], etc.) there are, with the exception of subdivision applied to box-spline control meshes, presently no generalized subdivision schemes that yield $C^2$ surfaces. The work on reparametrization and geometric smoothness by numerous researchers (see [Gregory '90] for a survey) is an essential tool for deriving free-form surface splines. Numerous algorithms are based on these techniques, but only the approach in [Hahn '89] yields $C^2$ surfaces; these surfaces are of degree bi-15. Moreover, as with G-splines, constraint systems have to be solved to enforce patch to patch smoothness making it difficult to predict the resulting shape.

Other approaches to building $C^2$ surfaces rely on the availability of consistent curvature information along a network of curves [Hagen, Pottmann '91], [Bajaj, Ihm, Warren '94]. A-patches [Bajaj, Chen, Xu '94] (see also [Guo '91],[Dahmen, Thamm-Schaar '93]), B-patches ([Seidel '91], [Dahmen, Micchelli, Seidel '92]) and S-patches [Loop, DeRose '90] provide alternative solutions to the smoothing problem at the cost of a presently non-standard patch representation.

**Properties of surface splines.** Following [Peters '93,95], we list a number of properties of a surface representation desirable for geometric modeling.

- *free-form modeling capability.* There are no restrictions on the number of cells meeting at a mesh point or the number of edges to a mesh cell. Mesh cells need not be planar.
- *built-in smoothness (unless explicitly reduced) and local smoothness preserving*

*editability.* For given connectivity and shape parameters, the surface spline form a vector space of geometrically smooth surface parametrizations. In order to manipulate the surface spline, it suffices to add, subtract or move the mesh points locally.

- *low degree parametrization.* The surface is parametrized by low degree polynomial patches. The representation can be extended to rational patches by using a fourth coordinate.
- *simple interpolation.* Mesh points and normals can be interpolated without solving a system of constraints.
- *evaluation by averaging.* The coefficients of the parametrization in Bernstein-Bézier form can be obtained by applying averaging masks to the input mesh. (The Bernstein-Bézier form in turn is evaluated by averaging.) Thus the algorithm is local and can be interpreted as a rule for cutting an input polytope such that the limit polytope is the spline surface.
- *convex hull property.* The surface lies locally and globally in the convex hull of the input mesh. In other words, every point on the surface can be computed as an average of the mesh points with coefficients that are positive and sum to one.
- *intuitive shape parameters.* The averaging process is geometrically intuitive. Parameters analogous to knot distances govern the depth of the cuts into the polytope outlined by the control mesh. (In concave regions the complement of the polytope is cut.) Smaller cuts result in a surface that follows the input mesh more closely and changes the normal direction more rapidly across the boundary. In the limit this allows adjusting the built-in smoothness, e.g. reducing it to continuity for zero cuts. Discontinuity can be achieved by a change of mesh connectivity.
- *taut interpolation of the control mesh for zero blend ratios.* Cuts of zero depth result in a singular parametrization at the mesh points analogous to singularities of a spline with repeated knots. The continuity of the surface is reduced, but in return the edges of the input mesh are interpolated and the surface is taut, e.g. planar when the mesh cell is planar.

In general any two of the following three properties can be achieved for any input mesh using $C^2$-surface splines. This is optimal in the sense that, as the reader may check, no surface construction (in fact not even a curve construction) can achieve all three properties simultaneously for all data sets.

convex hull property

curvature continuity interpolation

**Overview.** Section 2 develops concepts and techniques for the derivation and analysis of $C^k$-surface splines. Specific $C^2$-surface splines are defined in Section 3, in terms of Bernstein-Bézier patches. Section 4 establishes the continuity and vector space properties of these splines and Section 5 establishes shape properties of the resulting surfaces. Appendix 1 explains how coefficients of adjacent patches are labelled. Appendix 2 lists the formulas of individual Bernstein-Bézier coefficients in terms of intermediate control points. Appendix 3 features a Maple program that checks correctness of the $C^2$ construction given the formulas for two adjacent patches. Appendix 4 shows that the author has implemented the surface splines. Rather than a number of shapes, the variation of the curvature in terms of the blend ratios is shown.

## 2. Surface spline basics.

This section introduces concepts and techniques useful in deriving and analyzing surface splines. The general approach is made concrete by the $C^2$-surface splines defined in Section 3. The three topics are

1. Control meshes and mesh refinement.
2. Symmetry preserving $G^k$ joins between 2 patches and degree bounds.
3. Symmetry preserving $G^k$ joins among $n$ patches and the nullspace of the corresponding difference equations.

### 2.1. Control meshes and mesh refinement.

Surface splines are splines over irregular meshes. A mesh can be defined as a list of points with coordinates, usually in $\mathbb{R}^3$, and a list of connectivities, e.g. a list of cells where each cell is specified as an ordered list of points and any two consecutive points specify an edge of the mesh. The meshes we are interested in are *bivariate* in the sense that each edge is shared by exactly two cells. A mesh is called *irregular* to indicate that there are no further restrictions on its connectivity. In particular, a mesh point may have $n \neq 4$ neighbors and a cell can have $m \neq 4$ edges. Mesh cells need not be planar. When they are planar, they may be called facets and the mesh a polyhedron. Since surface splines average the mesh points to generate the surface, meshes are generally required to be *projectively convex*. A mesh is projectively convex, if for each cell, there exists a projection of the cell vertices into a plane such that none of the projected vertices lies in the convex hull of the other projected vertices. This property preserves the design intent, when a cell is a facet of a boundary representation with inner loops. Since inner loops define holes in the facet the cell should be broken up to prevent the surface from covering the intended hole.

A good strategy for dealing with an irregular mesh is to insert a midpoint on every edge and connect the midpoints of a cell to its centroid. This *midpoint refinement* has the advantage of decreasing the combinatorial complexity of the mesh: after the refinement every original vertex is surrounded by vertices with four neighbors and all cells are quadrilateral. The improved regularity can be used to trade, in the spirit of all spline constructions, a larger number of surface pieces for a lower degree of the polynomial surface For an example, see Figure 3.1which illustrates the first step of the construction in Section 3. It would be of interest to find a strategy for recursively refining the mesh such that the limit is a $C^k$ surface. At present no uniform strategy or subdivision algorithm is known to generate highly smooth surfaces from irregular meshes. Surface splines may be viewed as a two-stage averaging process; the first stage corresponds to the midpoint refinement and generation of the Bernstein-Bézier coefficients, the second applies a local corner cutting, namely de Casteljau's algorithm.

### 2.2. Symmetric $G^k$ joins between 2 patches and degree bounds.

Let $p$ be a $k$ times continuously differentiable map from a polygon $\Omega_p \subset \mathbb{R}^2$ to a surface piece in $\mathbb{R}^3$ and $\phi_p$ an invertible $C^k$ map from an open neighborhood of the line segment $E := \{(u,0) : u \in [0,1]\} \subset \mathbb{R}^2$ to an open neighborhood of the edge $e_1$ of $\Omega_p$. Let $q$ and $\phi_q$ be defined analogously; that is, both $\phi_p$ and $\phi_q$ are defined in a neighborhood of the edge $E$ (see Figure 2.1).

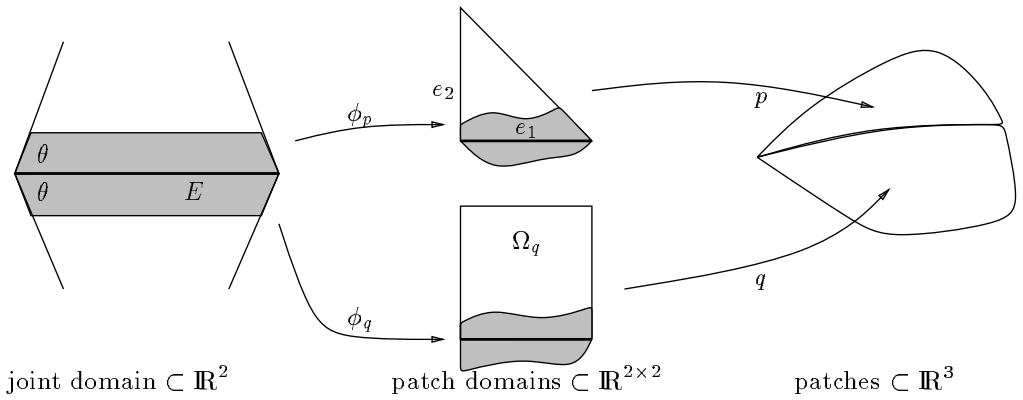



Fig. 2.1: Smooth join of two patches along a common boundary curve

**Definition.** *Two patches $p$ and $q$ join $G^k$ with respect to $\phi_p$ and $\phi_q$ if and only if along the edge $E$*

$$D_2^m[p \circ \phi_p - q \circ \phi_q](E) = 0 \qquad \text{for } m = 0..k, \tag{$G^k$}$$

*where $\circ$ denotes composition of maps, $D_2$ the derivative in the direction perpendicular to $E$ and $=$ equates functions restricted to $E$.*

To have affine invariance of the surface splines, the reparametrization must only depend on the connectivity of the input mesh, and not on the geometric position of the mesh points. A second requirement for effective geometric modeling is the ability to keep the construction local so that changes in the connectivity, say attaching a handle, do not necessarily propagate across the whole surface. Localizing the influence of connectivity to the immediate neighborhood determines the opening angles at an endpoint of $E$ to be $\theta := 2\pi/n$ if the endpoint has $n$ neighbors. Since the maps $\phi_p$ and $\phi_q$ are only defined locally, namely on a small strip that forms a neighborhood of $E$, and since that neighborhood is invariant under the reflection $r(u,v) := (u,-v)$, the maps $\phi_p$ and $\phi_q$ should be identical after reflection. That is $\phi_q = \phi_p \circ r$. This implies

$$D_2^m \phi_q = D_2^m(\phi_p \circ r) = (-1)^m D_2^m \phi_p.$$

We derive the continuity conditions for the special, symmetric choice of $\phi_q$ and $\phi_p$. Let $\phi^{[j]}$ denote the $j$th component of $\phi$, $j = 1$ or $j = 2$ let $E$ tace out the first unit vector $e_1$cw and, hence $D_1^m p = D_1^m q$.

*Example $m = 1$.* Define $\lambda := -2D_2\phi_p^{[1]}/D_2\phi_p^{[2]}$. The denominator is nonzero, because $\phi_p$ is by assumption regular and invertible and $\phi_p(e_1) = e_1$, hence $D_1\phi_p^{[2]} = 0$. Since $\phi_q(u,0) = (u,0)$, we may abbreviate $D_i q(\phi_q) = D_i q$ in the expansion

$$0 = D_2(p \circ \phi_p - q \circ \phi_q) = D_1 p D_2\phi_p^{[1]} + D_2 p D_2\phi_p^{[2]} + D_1 q D_2\phi_p^{[1]} + D_2 q D_2\phi_p^{[2]}$$

After division by $2D_2\phi_p^{[2]}$, we have the symmetric $G^1$ constraint

$$D_2 p + D_2 q = \lambda D_1 q. \tag{$G^1$}$$

*Example* $m = 2$. The $G^2$ constraints expand as follows.

$$\begin{aligned} D_2^2(q(\phi_q)) &= D^2 q\big(D_2\phi_q, D_2\phi_q\big) + Dq D_2^2\phi_q \\ &= D_1^2 q(D_2\phi_q^{[1]})^2 + 2D_1D_2q(D_2\phi_q^{[1]})(D_2\phi_q^{[2]}) + D_2^2q(D_2\phi_q^{[2]})^2 \\ &\quad + D_1 q D_2^2\phi_q^{[1]} + D_2 q D_2^2\phi_q^{[2]} \end{aligned}$$

Since $p$ and $q$ share the boundary curve, $D_1^m p = D_1^m q$. Subtracting the expansion for $D_2^2(p(\phi_p))$, dividing by $(D_2\phi_p^{[2]})^2$ and defining $\alpha := D_2^2\phi^{[2]}/(D_2\phi_p^{[2]})^2$, we get the symmetric $G^2$ constraint

$$D_2^2 p - D_2^2 q - \lambda D_1 D_2 p + \lambda D_1 D_2 q = (D_2 q - D_2 p)\alpha \qquad (\mathrm{G}^2)$$

(which can be rewritten as $[D_2^2 - \lambda D_1 D_2 + \alpha D_2]p = [D_2^2 - \lambda D_1 D_2 + \alpha D_2]q$.)

The key to surface splines of low degree is to find connecting-maps $\phi_i$ at the mesh points such that, at each edge, a low degree Hermite interpolant $\phi$ exists that joins the connecting-maps at the end points. Connecting-maps at the endpoints are discussed in detail in the next section. For now it suffices to know that their Taylor-expansion is not completely arbitrary. We make the effect of these restrictions precise in the context of curvature continuous splines. The midpoint refinement creates two types of vertices (cf. Figure 3.1): vertices of type P correspond to original mesh points and centroids and those of type M to edge-midpoints. In the case of regular, checker board arrangement of the patches, the Hermite interpolant to the expansions of the reparametrizations at the end points P, M can be simple: since each neighbor P has itself $n = 4$ neighbors we can choose $\phi_p$ to be the identity and the patches consist of tensor-product splines. If we split the cells by introducing points S as in Figure A.1, we get a four-direction box spline. If, however, $n \neq 4$ patches join at the vertex of type P, then theorems [Peters '94, Thm 4] and [Peters '94, Prop 5] rule out both a constant and a linear $\lambda$ and hence $\phi_p$. Hence the $C^2$-surface splines of Section 3 use the least degree polynomial Hermite interpolant by setting

$$\lambda(u) := (1 + \cos(\frac{2\pi}{n}))(1-u)^2 + 2(1-u)u + u^2.$$

For the given, minimal polynomial choice of $\lambda$, we now estimate the degree of a piecewise polynomial curvature continuous surface construction. Consider the two patches $p$ and $q$ with a common boundary curve $\gamma$ of degree $d$. Then the left side of the symmetric $G^1$ constraints,

$$\lambda D_1\gamma = D_2 p + D_2 q,$$

is of degree $2+d-1$ and hence $D_2p$ and $D_2q$ are formally of degree $d+1$. We can only say formally, since $D_2p$ and $D_2q$ may be polynomials in a degree-raised representation. The terms $\lambda D_1 D_2 p$ and $\lambda D_1 D_2 q$ in the symmetric $G^2$ constraints,

$$D_2^2 p - D_2^2 q - \lambda D_1 D_2 p + \lambda D_1 D_2 q = \alpha(D_2 q - D_2 p),$$

are therefore of degree $2 + d + 1 - 1$. Hence $D_2^2 p$ and $D_2^2 q$ are at least of degree $d + 2$ if there is no cancellation and if none of the polynomials is degree-raised, and $p$ and $q$ must be of degree $d + 4$.

*Example.* In Section 3.2, boundary curves $\gamma$ of polynomial degree 4 are constructed to match curvature data at the end points. Hence the polynomial degree of the overall construction is 8, while the tangent across the boundary is a polynomial of degree 5. The same type of argument yields a lower bound of bi-6 in the case of tensor-product patches. However, this bound can not be attained because of certain rank-dificiencies pointed out in [Peters '92].

### 2.3. $G^k$ joins among $n$ patches and the nullspace of the corresponding difference equations.

The smooth join of $n > 2$ patches at a vertex is qualitatively more difficult than the join between 2 patches, because when three or more patches join smoothly at a common point, the pairwise continuity constraints between the patches form a cyclic system [Peters '92]. Once the connecting maps at a vertex are chosen, the continuity constraints on the coefficients can be considered as difference equations. Their nullspace corresponds to the free parameters that determine the shape of the surface at the vertex. For the construction of Section 3, exactly six coefficients, corresponding to the dimension of a $C^2$ piecewise quadratic, are to be chosen in the 2-disk of coefficients labeled $P_{klm,i}$, $k + l + m = d$, $k = d - 2..d$ (cf. Figure 2.2). A possible approach is to randomly pick a patch $i_0$ and prescribe the six coefficients $P_{klm,i_0}$. But this gives unwarranted preference to patch $i_0$ over the other patches and, more importantly, does not readily provide rules for determining the remaining coefficients from the given control mesh by averaging. It is better to view the six degrees of freedom as representing the null space of the $G^2$ constraints. Usually, the number of neighbors is not a multiple of the number of data required to match the dimension of that null space. If it is, i.e. if $n = 3$ or $n = 6$, we can pick data symmetrically from the neighbors, and thus determine the solution of the difference equation and ultimately the surface uniquely. In the general case, a good strategy is to relate the available data to the nullspace by creating *intermediate control points* that represent the geometry of the data but belong to a subspace of $\mathbb{R}^{3n}$ that has the same dimension as the nullspace. Examples of such intermediate control points are $C_i$ in [Peters '93,95] and the quasi-control points in [Reif '93].

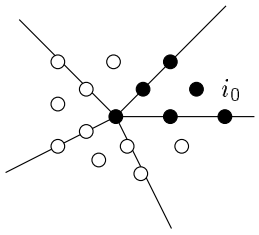



Fig. 2.2: Coefficients of a 2-disk, $n = 5$.

*Example.* Consider the the null space of the constraint system for second order continuity at a vertex of type P surrounded by $n$ patches that is generated by choosing connecting-map $\phi_p$ and $\phi_q$ such that

$$\lambda(u) = -2D_2\phi_p^{[1]}/D_2\phi_p^{[2]}(u) = (1+c)(1-u)^2 + 2(1-u)u + u^2$$
$$\alpha(u) = D_2^2\phi^{[2]}/(D_2\phi_p^{[2]})^2(u) = \kappa t.$$

The goal is to affinely and uniformly construct dependent control points, $P_{220,i}$, $P_{310,i}$ and $P_{400}$ that together represent the six freely choosable vector-valued quantities. We use one such degree of freedom by choosing $P_{400} = 0$. On eliminating the constraints across the splitting edges, we obtain the following $3n$ difference equations in the $3n$ variables $P_{310,i}$, $P_{611,i,1}$ and $P_{220,i}$ (the next section below gives the details of such an elimination).

$$0 = P_{310,i-1} - 2cP_{310,i} + P_{310,i+1} \tag{2.1}$$
$$\begin{aligned} 3(1+2c)P_{220,i-1} - 6cP_{220,i} - 3P_{220,i+1} + 56c(P_{611,i,1} - P_{611,i-1,1}) \\ = 2(3-15c-\kappa)P_{310,i-1} + 16cP_{310,i} + 2(-3+7c+\kappa)P_{310,i+1} \end{aligned} \tag{2.2}$$
$$\begin{aligned} 6(P_{220,i-1} + 6(1+2c)P_{220,i} - 56(P_{611,i,1} + P_{611,i-1,1}) \\ = 20P_{310,i-1} + 4(11-3c)P_{310,i} + 4P_{310,i+1} \end{aligned} \tag{2.3}$$

The first equation forces the tangent coefficients $P_{310,i}$ into a common plane, while (2.2) and (2.3) define the curvature of the surface at the vertex. The constraints represent difference equations with periodic boundary conditions and can be analyzed by discrete Fourier analysis. Let $C_i$ for $i = 1..n$ be $n$ points in space. The points need not be in any special configuration. In step 3.1 of the algorithm, the points are derived from the data surrounding a mesh point. If $n \neq 4$, the difference equations have a symmetric solution that averages the $C_i$ as follows.

$$B_{220} := \sum_j C_j/n \qquad \theta := 2\pi/n,$$
$$p_{220,i} := 2\sum_j \cos(\theta(j+i))C_j/n,$$
$$e_{220,i} := 2\sum_j \cos(\theta(j+2i))C_j/n$$
$$P_{220,i} := B_{220} + p_{220,i} + e_{220,i}$$
$$P_{310,i} := P_{040} + \frac{3}{6-2c}p_{220,i}$$

For $n = 4$, $c = 0$ and the constraints simplify to a set of equations that define the neighborhood of a vertex of a 4-direction $C^2$ box spline.

We now give a detailed derivation of (2.1) from ($G^1$). Let $p$ and $q$ be a polynomials of total degree 8 in Bernstein-Bézier form with coefficients $P_{ijk}$ and $Q_{ijk}$, $i+j+k=8$ (cf. Appendix 1), and let $P_{ij0} = Q_{ij0}$ be the coefficients of a joint boundary curve of degree 6. Let $E := \{(u,0), u \in [0,1]\}$ be the preimage of that boundary curve, $\lambda$ a quadratic polynomial with Bernstein-Bézier coefficients $\lambda_{20} = 1+c$, $\lambda_{11} = \lambda_{02} = 1$ and $D_i p(E)$, $i \in \{0,1\}$ be the univariate polynomial of the derivative of $p$ in the direction $e_i$ restricted to the boundary. Then

$$\lambda D_1 p = D_2 p + D_2 q \qquad (G^1)$$

equates two univariate vector-valued polynomials of degree 7. In Bernstein-Bézier representation

$$D_2 p(u) := 8 \sum_{i+j=7} (P_{i,j,1} - P_{i+1,j,0}) \binom{7}{j} (1-u)^i u^j$$

and

$$D_1 p(u) := 6 \sum_{i+j=5} (P_{i,j+1,0} - P_{i+1,j,0}) \binom{5}{j} (1-u)^i u^j .$$

The polynomial equation ($G^1$) holds if and only if all 8 coeffcients agree. For example, looking at the first coefficient,

$$(1+c)6(P_{510} - P_{600}) = 8(P_{7,0,1} - P_{8,0,0}) + 8(Q_{7,0,1} - Q_{8,0,0})$$

has to hold. Moving $P_{800}$ to the origin and noting that $(6/8)(P_{510} - P_{600}) = P_{710} - P_{800}$ if $P_{710}$ and $P_{800}$ are the coefficients of the boundary curve with the degree raised to eight, we have equivalently

$$(1+c)P_{710} = P_{701} + Q_{701}.$$

Similarly, if $\lambda$ is the constant map 2 and $p$ and $r$ abut along $P_{i0k} = R_{i0k}$, then

$$2P_{701} = P_{710} + R_{710}$$

has to hold.

To derive (2.1), we consider $2n$ patches $p_{i,j}$, $i = 1..n$, $j = 1,2$ meeting at $0 = P_{800,1,1} = \ldots = P_{800,n,2}$ with the boundaries identified according to the convention

$$P_{k0m,i} := P_{k0m,i,1} = P_{k0m,i,2} \quad \text{and} \quad P_{km0,i} := P_{km0,i,1} = P_{km0,i-1,2}.$$

If every second join is via the constant map 2 and every other join via the map $\lambda$, then the first coefficient yields the equations

$$\begin{aligned}(1+c)P_{710,i} &= P_{701,i-1} + P_{701,i} \\ 2P_{701,i} &= P_{710,i} + P_{710,i+1}.\end{aligned}$$

On eliminating $P_{701,i}$ for $i = 1..n$, we get the difference equation (2.1)

$$P_{710,i-1} - 2cP_{710,i} + P_{710,i+1} = 0.$$

**3. An algorithm for generating the Bernstein-Bézier control points of a $C^2$ surface from an irregular mesh of surface spline control points.**

Analogous to tensor product (B-)splines, a surface spline is defined by

- mesh points $\mathcal{M}$,
- mesh connectivity $\uparrow$,
- blend ratios (knot spacings) $a_i$,

and an evaluation algorithm. Since the evaluation of polynomials in Bernstein-Bézier form is well understood, it suffices to express the surface spline in terms of the Bernstein-Bézier form. In the following, this basis conversion is given explicitly by expressing the Bernstein-Bézier coefficients as combinations of intermediate control points derived from the mesh points by mesh refinement. The algorithm is broken into the six steps (see also Figure 3.3).

*Algorithm* $(\mathcal{M}, \uparrow, a_{ij}) \mapsto P_{ijk,l}$.

**1** Mesh refinement and intermediate control points.
**2** Quartic boundary curves along P–M.
**3** Position, Tangent and Curvature at P and M.
**4** Curvature continuity across P–M.
**5** Quintic splitting curves.
**6** Position, Tangent and Curvature at S.

The *input* is any mesh of points such that at most two cells abut along any edge. The mesh cells need not be planar, and there is no constraint on the number of cells meeting at a vertex. To achieve the design intent, each mesh cell should be convex in the sense that there exists a projection of the cell vertices into a plane such that none of the projected vertices lies in the convex hull of the other projected vertices. Thus a facet whose boundary representation has an inner loop should be broken up before using it as a cell, since the surface is generated by averaging and hence will generally smoothly cover the intended hole (see for example [Peters '93,95, Fig.5.4]). The mesh may model bivariate surfaces with or without boundary and of arbitrary topological genus; for a discussion of boundary conditions see [Peters '93], [Loop '94]. The input *parameters* are as follows. Associated with each pair cell and cell vertex are two scalar weights $0 < a_i < 1$, $i = 1, 2$, called *blend ratios*. Geometrically, smaller ratios result in a surface that follows the input mesh more closely and changes the normal direction more rapidly close to the mesh edges. The default is $a_i^* := 1/2$. The blend ratios are similar to relative knot spacings. In particular if all $a_i$ associated with an edge are zero, the mesh edge is interpolated and the smoothness is reduced to continuity. Discontinuity is modeled by mesh disconnectivity. The blend ratios of each cell may be modified independently of each other and of those in other cells. Other parameters, $h_{\mathrm{P}}$ (respectively $h_{\mathrm{M}}$ or $h_{\mathrm{S}}$) may be moved from their default value to pull the surface towards the mesh point P (respectively M or S) while $\mathrm{w}_n$ and $b_{\mathrm{P}}$ (respectively $b_{\mathrm{M}}$ or $b_{\mathrm{S}}$) scale the tangent and curvature vectors of the surface close to the mesh point.

The *output* is a surface that follows the outline of the input mesh and consists of no more than $8e$ quartic or octic, three-sided patches that form a $C^2$ surface, where $e$ is the number of edges of the input mesh. As in the case of $C^1$ free-form surface splines alternative representations consisting of bisextic, four-sided patches, or a combination of biquartic and bicubic, four-sided patches covering regular mesh regions and octic, three-sided patches for the remaining regions can be devised (see Remark 3.4).

### 3.1. Mesh Refinement and intermediate control points

The purpose of the mesh refinement is to simplify the combinatorial structure of the mesh and to generate intermediate control points $C_i$. The refinement looks like the first step of any line average or subdivision algorithm for regular meshes. It splits each original $n$-sided cell into $n$ four-sided *subcells* and then further into triangles creating three types of points labeled as follows.

P— input mesh points and cell centroids created in step 2 below.

M— edge midpoints created in step 1 below.

S— subcell centroids created in step 4 below.

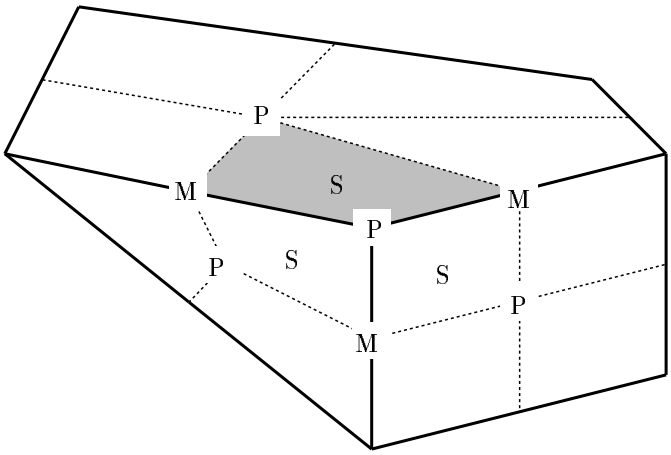



**Figure 3.1:** Midpoint subdivision.

The algorithm for the refinement is as follows.

1 For each edge, insert a midpoint.

2 For each cell, insert a centroid (the average of the vertices of the cell).

3 For each cell, connect the centroid to all midpoints creating subcells.
Now every type P vertex is surrounded by type M vertices of degree four and all subcells are quadrilateral.

4 For each subcell, insert a centroid.

5 For each subcell, connect the centroid to the four vertices of the subcell.

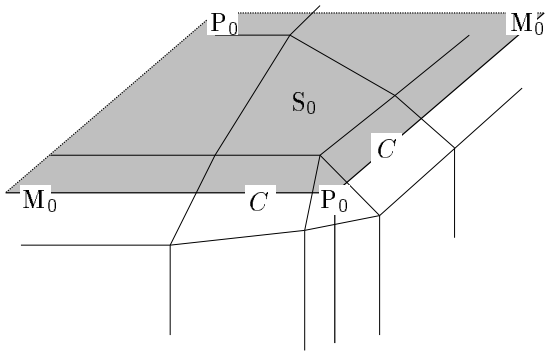



**Figure 3.2:** Intermediate control points from two subdivision steps.

The intermediate control points are an average of the original mesh points. Their position reflects and transmits the choice of the input blend ratios $a_i$ because the ratios weigh two Doo-Sabin refinement steps as in [Peters '93,95] (see also [Peters '93p 350]). The Doo-Sabin steps are applied to the mesh after step 3 above and serve only to generate intermediate control points. In particular, they do not generate more patches. We keep only the centroids of the resulting cells and label them $\mathrm{P}_0$, $\mathrm{M}_0$ or $\mathrm{S}_0$ if the cells correspond to a vertex of the refined mesh or $C$ if the cell corresponds to an edge.

Thus, at the end of Step 3.1 we have a refined mesh of intermediate control points of types $C$, $\mathrm{P}_0$, $\mathrm{M}_0$ and $\mathrm{S}_0$ spaced and situated subject to the blend ratios. In the following, various parameters may be chosen in an interval. The default value is indicated by the superscript $^*$. Coefficient labels follow the system explained in Appendix 1.

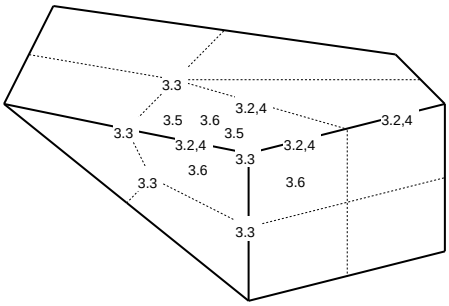



**Figure 3.3:** Construction steps associated with surface regions.

**3.2. Quartic boundary curves along** P–M. Let $C_1$, ..., $C_n$ be the subcell edge-coefficients surrounding P. Compute

$$\begin{array}{rll} B_{220} := & \frac{1}{n}\sum C_i, & \\ p_{220,i} := & \frac{2\mathrm{w}_n}{n}\sum_{j=1}^{n}\cos(\frac{2\pi}{n}j)(C_{i+j}-B_{220}), & \mathrm{w}_n\in[0..1],\quad \mathrm{w}_n^*:=1,\\ e_{220,i} := & \frac{b_\mathrm{P}}{n}\sum_{j=1}^{n}\cos(\frac{2\pi}{n}2(i-j))(C_{i+j}-B_{220}), & b_\mathrm{P}\in[0..1],\quad b_\mathrm{P}^*:=1. \end{array}$$

The coefficients of the quartic boundary curve corresponding to the edge PM are

$$\begin{array}{rll} P_{400} := & h_\mathrm{P}\mathrm{P}_0+(1-h_\mathrm{P})B_{220} & h_\mathrm{P}\in[0..a_\mathrm{P}],\quad h_\mathrm{P}^*:=1-a_\mathrm{P}\\ P_{310,i} := & P_{400}+a_\mathrm{P}p_{220,i}, & a_\mathrm{P}:=\frac{3}{2(3-c)},\\ P_{220,i} := & B_{220}+p_{220,i}+e_{220,i} & \\ P_{130,i} := & P_{040}+\frac{1}{4}(P_{220,i}-P_{220,i+2}). & \\ P_{040} := & h_\mathrm{M}\mathrm{M}_0+(1-h_\mathrm{M})\sum_{i=1}^{4}\frac{P_{220,i}}{4}, & h_\mathrm{M}\in[0..\frac{1}{2}],\qquad h_\mathrm{M}^*:=\frac{1}{2} \end{array}$$

**3.3. Position, Tangent and Curvature at** P **and** M. This step determines the 3-disk of Bernstein-Bézier coefficients $P_{klm,i}$, $k>4$ , $k+l+m=8$, $i=1..n$ surrounding the vertex P with $n\neq 4$ and the 3-disk of Bernstein-Bézier coefficients $P_{klm,j}$, $l>4$, $k+l+m=8$, $j=1..4$ surrounding vertices of type M. The explicit expressions are collected in Table 1 and Table 2 of Appendix 2. The coefficients at P for $n=4$ are obtained from Table 2 by swapping the first with the second subscript: the formula for $P_{klm}$ can be looked up under $P_{lkm}$.

**3.4. Curvature continuity across** P–M. The construction of the curvature caps at

P and M defines the quartic boundary curve with the coefficients $P_{400}, P_{310}, P_{220}, P_{130}, P_{040}$, and corridor of coefficients $P_{7-i,i,1}$ on either side of the edge that correspond to degree-raised sextic patches. Explicit formulas are listed in Table 3. The construction of the $C^2$ corridor connecting the curvature caps is local to each patch. That is, each of the coefficients $P_{242,j}$, $P_{332,j}$, and $P_{422,j}$, $j = 1, 2$ depends only on the coefficients of the $j$th of the two abutting patches plus a vector $K_{4-i,2+i,2}$, $i = 0, 1, 2$, that can be chosen freely without affecting the continuity. By default $K_{4-i,2+i,2} = 0$. Explicit formulas for coefficients $P_{4-i,2+i,2}$, $i = 0, 1, 2$ are listed in Table 3.

**3.5. Quintic splitting curves.** This step constructs the curves corresponding to the edges PS.

$$
\begin{aligned}
P_{500,1} &:= P_{800,1} \\
P_{401,1} &:= \frac{8}{5}P_{701,1} - \frac{3}{5}P_{800,1} \\
P_{302,1} &:= \frac{28}{10}P_{602,1} - \frac{24}{10}P_{701,1} + \frac{6}{10}P_{800,1} \\
P_{203,1} &:= \frac{56}{10}P_{503,1} - \frac{84}{10}P_{602,1} + \frac{48}{10}P_{701,1} - \frac{10}{10}P_{800,1} \\
P_{104,1} &:= P_{005} + \frac{1}{4}(P_{203,1} - P_{203,3}) \qquad\qquad h_s^* := \frac{1}{2}, \\
P_{005,1} &:= h_{\mathrm{S}}\mathrm{S}_0 + (1 - h_{\mathrm{S}})\frac{1}{4}(P_{302,1} + P_{032,2} + P_{302,3} + P_{032,4}), h_{\mathrm{S}} \in [0..\frac{1}{2}]
\end{aligned}
$$

The formulas for the constructions of curves MS differ only in that the first and the second index of the coefficients is switched. The boundary curves are degree-raised to yield the coefficients $P_{6-i,0,i}$ and $P_{0,6-i,i}$ for $i = 0..6$.

**3.6. Position, Tangent and Curvature at S.** If the patches were of degree 6, the remaining coefficients would be uniquely determined by the remaining (univariate) $C^2$ constraints across the splitting edges. By default, we therefore determine the coefficients $P_{klm}$, $m > 2$ of each patch by constructing a patch of degree 6 that satisfies the $C^2$ constraints across the splitting edges and agrees with the patch constructed so far except for $P_{4-i,d+i,2}$, $i = 0..2$. Concretely, we compute coefficients $P_{3-i,1+i,2}$ such that the difference between $P_{4-i,d+i,2}$, $i = 0..2$ and the coefficients of the sextic raised to degree 8 is minimal. The formulas for $P_{6-i-j,i,j}$, $j < 3$ satisfying this criterion are given in Table 4. With the coefficients $P_{6-i,0,i}$ and $P_{0,6-i,i}$ for $i = 0..6$ given by step 3.5, we compute the remaining coefficients of the sextic as

$$
\begin{aligned}
P_{213,i} &= \frac{1}{2}(P_{204,i} + P_{222,i}), \qquad P_{123,i+1} = \frac{1}{2}(P_{204,i} + P_{222,i+1}), \\
P_{204,i} &= \frac{1}{2}(P_{204,i} + \frac{1}{2}(P_{222,i} + P_{222,i+1})), \\
P_{114,i+1} &= \frac{1}{2}(P_{105,i} + \frac{P_{213,i+1} - P_{123,i}}{4} + P_{015,i} + \frac{P_{213,i} - P_{123,i-1}}{4}).
\end{aligned}
$$

The final step is to raise the degree of the sextic to obtain $P_{8-i-j,i,j}$ for $j > 3$ and

adjust

$$P_{314,i} = \frac{1}{2}(P_{305,i}P_{323,i}), \qquad P_{134,i+1} = \frac{1}{2}(P_{305,i} + P_{233,i+1}),$$
$$P_{305,i} = \frac{1}{2}(P_{305,i} + \frac{1}{2}(P_{233,i+1} + P_{323,i})),$$
$$P_{413,i} = \frac{1}{2}(P_{404,i} + P_{422,i}), \qquad P_{143,i+1} = \frac{1}{2}(P_{404,i} + P_{242,i+1}),$$
$$P_{404,i} = \frac{1}{2}(P_{404,i} + \frac{1}{2}(P_{422,i} + P_{242,i+1})).$$

Step 3.6 completes the construction. A number of variations on the construction are possible. For example, we can cover the neighborhood of a regular mesh point with biquartic patches, and the neighborhood of a regular point surrounded by regular mesh points using bicubic patches.

**3.7 Local interpolation of input mesh points.** A vertex P can be interpolated simply by choosing $h_{\mathrm{P}}$ in Step 3.2 so that $\mathrm{P} = h_{\mathrm{P}}\mathrm{P}_0 + (1-h_{\mathrm{P}})B_{220}$. Unless we choose the blend ratios in the neighborhood equal zero and loose tangent plane continuity, the surface will then generally not lie in the convex hull of the local mesh points.

**Remark 3.8** A user can interact with the surface splines via the control mesh, the blend ratios and the other parameters or some higher level editor and need not be aware of the details of the underlying representation in Bernstein-Bézier form.

**Remark 3.9** Except for the default choice of free parameters, the derivation of the formulas requires only the specification of a reparameterization map. For constructions based on four-sided patches, this map is stated after Theorem 4.1.

## 4. Continuity and vector space properties.

This section proves that splines based on the same mesh connectivity and ratios form a vector space of curvature continuous maps. The proof is unusual in that the checking of the constraints in terms of the Bernstein-Bézier coefficients is left to a symbolic routine listed in Appendix 3. After checking the correctness of the Maple code, the tedious comparisons of coefficients can be left to the computer. (The referees were provided with the code and the coefficients in electronic form.)

THEOREM 4.1. *The algorithm of Section 3 generates the Bernstein-Bézier representation of a $C^2$ surface.*

*Proof.* Applying the Maple routines of Table 5 to the coefficients of Tables 1–3 with a reparametrization $f$ such that

$$\lambda(t) := D_2 f^{[1]}(t) = (1-t)^2 + 2(1-t)t + (1-c)t^2, \quad \alpha(t) := D_2 D_2 f^{[1]}(t) = c(1-2c)t$$

shows that the symmetric $G^1$ and $G^2$ constraints hold along the edge MP:

$$D_2 p + D_2 q = \lambda D_1 q,$$
$$D_2^2 p - D_2^2 q - \lambda D_1 D_2 p + \lambda D_1 D_2 q = (D_2 q - D_2 p)\alpha.$$

Since $p$ and $q$ share the boundary curve corresponding to MP they join $G^2$.

Across the edges PS, respectively MS, the patches join parametrically $C^2$. That is, any six coefficients $P_{20}$, $P_{11}$, $P_{02}$, $Q_{20}$, $Q_{11}$, $Q_{02}$ on a line transversal to the common

edge of patches $p$ and $q$ respectively satisfy $P_{02} = Q_{20}$, $P_{11} := P_{02} - (Q_{02} - P_{20})/4$ and $Q_{11} := P_{02} + (Q_{02} - P_{20})/4$. The Maple routine also checks the first three $C^1$ and the first two $C^2$ constraints across the splitting edges PS and MS. The remaining constraints are straightforward to check. □

The corresponding reparametrization for a construcion using four-sided patches is

$$\lambda(t) := 2ct^2, \quad \alpha(t) := 2c(1-2c)t.$$

THEOREM 4.2. *$C^2$-surface splines with the same connectivity, and blend ratios form a vector space.*

*Proof.* Blend ratios and connectivity fix the reparametrizations. For fixed reparametrizations, linearity of differentiation implies the vector space property. □

## 5. Shape properties of $C^2$ surface splines.

This section establishes the convex hull property of free-form surface splines as well as the 'tautness property': the edges of the input mesh are interpolated and thus the outlines of the input polytope recaptured when the blend ratios are zero. The following theorem establishes the convex hull property under worst case estimates. Thus the bounds on the constants are more conservative than they have to be in generic use. For example, for extremely asymmetric data (cf. [Peters '93,95, Sec.4]), $\mathrm{w}_n \leq 1/2$ must hold, while for symmetric configurations $\mathrm{w}_n = 1$ yields a construction guaranteed to obey the convex hull property.

**Theorem 5.1.** *If $h_\mathrm{P}, h_\mathrm{M} < 1/5$, $w_n < 1/2$, $b_\mathrm{P} = 0$, then $K_{4-i,2+i,2}$, $i = 0, 1, 2$ can be chosen so that all coefficients of the BB-form are a convex combination of the input mesh points.*

*Proof.* The proof follows the steps of the algorithm. We write $X \subset H(Y)$ if every vertex of type $X$ is in the convex hull of the vertices of type $Y$ that enter the computation of $X$.

1. The mesh refinement followed by the two Doo-Sabin steps with weights $a_i \in [0, 1]$ are averaging steps and hence construct

$$\mathrm{P}_0, \mathrm{M}_0, \mathrm{S}_0, C \subset H(\mathrm{P}).$$

2. If $\mathrm{w}_n \leq \frac{1}{2}$ then $B_{220} + p_{220,i} \subset H(C)$. If $b_\mathrm{P} = 0$ and $h_\mathrm{P} \in [0..a_\mathrm{P}]$, the construction of the quartic boundary curve results in

$$\begin{array}{rcccl}
P_{400} & := & (1-h_\mathrm{P})\mathrm{P}_0 + h_\mathrm{P} B_{220} & \subset & H(\mathrm{P}_0, C) \\
P_{220,i} & := & p_{220,i} + e_{220,i} & \subset & H(C) \\
P_{310,i} & := & P_{400} + a_\mathrm{P}(P_{220,i} - \mathrm{P}_0) & \subset & H(P_{220,i}, \mathrm{P}_0), \\
P_{130,i} & := & P_{040} + \frac{1}{4}(P_{220,i} - P_{220,i+2}) & \subset & H(P_{040}, C) \\
P_{040} & := & h_\mathrm{M}\mathrm{M}_0 + (1-h_\mathrm{M})\sum_{i=1}^4 P_{220,i}/4 & \subset & H(\mathrm{M}_0, C)
\end{array}$$

The statement for $P_{130,i}$ follows from the fact that $P_{220,i}$, $\frac{P_{220,i}+P_{220,i+2}}{2}$, M and $P_{130,i}$, $P_{040}$, M form similar triangles of half the size. That is the construction is the same as the derivation of the control points of the Bernstein-Bézier form of a cubic spline from its B-spline control points.

3. Substituting $B_{220} := \sum P_{220,i}/n$, $p_{220,i} := P_{220,i} - e_{220,i}$, $\mathrm{P}_0 := (P_{800} - h_\mathrm{P} B_{220})/(1-h_\mathrm{P})$, we find that the dominant term in $c := \cos(2\pi/n)$ of each coefficient

of $P_{611,i}$, $P_{521,i}$ and $P_{512,i}$ is positive and hence

$$\begin{aligned}
P_{611,i} &\subset H(\mathrm{P}_0, B_{220}, p_{220,i}, p_{220,i+1}) \\
&\quad + O(e_{220,i-1}, e_{220,i}, e_{220,i+1}) \\
P_{521,i} &\subset H(\mathrm{P}_0, B_{220}, p_{220,i}, p_{220,i+1}, P_{130,i}) \\
&\quad + O(e_{220,i}) \\
P_{512,i} &\subset H(\mathrm{P}_0, B_{220}, B_{130}, p_{220,i}, p_{220,i+1}, P_{130,i}) \\
&\quad + O(e_{220,i}, e_{220,i+1}).
\end{aligned}$$

$P_{8-i,i,0}$, $i = 0..3$ lie in the convex hull by degree raising and $P_{8-i,0,i}$, $i = 0..3$ lie in the convex hull as averages of coefficients $P_{611,i}$, $P_{521,i}$, and $P_{512,i}$.

At M, choosing $h_{\mathrm{M}} < 2/5$

$$P_{161,i} \subset H(\mathrm{M}_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3})$$

and for $0 < h_{\mathrm{M}} < \frac{1}{432}(96 - 2(c_{i+1})^2 + 8c_i + c_{i+1} - c_{i+3} + 2(c_{i+3})^2) > 1/5$

$$\begin{aligned}
P_{152,i} &\subset H(\mathrm{M}_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3}) \\
P_{251,i} &\subset H(\mathrm{M}_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3})
\end{aligned}$$

4. Since we can choose $K_{4-j,2+j,2}$, $j = 0, 1, 2$, arbitrarily, we can force $P_{4-j,2+j,2}$ into $H(P_{220,i})$. This argument avoids stating the lenghty estimate for $K_{4-j,2+j,2} = 0$. The convex hull property for step 5 and 6 follows from the construction of similar triangles analogous to the argument for $P_{130,i}$. □

PROPOSITION 5.2. *An edge between two cells with zero transversal cut ratios is interpolated. Planar cells with zero cut ratios are covered by a planar surface.*

*Proof.* The proof follows the steps of the algorithm. Zero cut ratios coalesce all cell centers $C_i$ surrounding an original mesh point P. That is,

$$\mathrm{P} = C_i = P_{220,i} = P_{310,i} = P_{400} = \mathrm{P}_0.$$

Also, labeling the centroids with even indices,

$$\mathrm{M} = C_0 = C_2 = P_{220,0} = P_{220,2} = P_{130,0}P_{130,2} = P_{040}$$

while $P_{130} = (\mathrm{P} + \mathrm{M})/2$. Table 1 shows that $P_{klm} = \mathrm{P}$ for $k \geq 6$, $P_{521}$ and $P_{512}$ are on the edge P, M, the latter by the choice of P3bs as specified in Table 1 of Appendix 2. Consequently $P_{503,i}$ lies in the plane spanned by P and successive neighbors $\mathrm{M}_i$ and $\mathrm{M}_{i+1}$. Table 2 shows that $P_{klm}$ for $l \geq 5$ lie on the edge P, M, except for $P_{053,i}$ which lies in the plane spanned by M and successive neighbors $\mathrm{P}_i$ and $\mathrm{P}_{i+1}$. Since $P_{4-j,2+j,2}$ for $j = 0, 1, 2$ is determined entirely by the coefficient of the patch, they are determined by the edge P–M and the two adjacent edges of the subcell to which the patch belongs. The remaining construction steps average the given coefficients hence place them in the plane defined by the four edges of the subcell. □

**6. Conclusion.** The goal of surface splines is to extend the spline paradigm and techniques to irregular meshes and thereby overcome the limitations of the B-spline approach without sacrificing the natural smoothing property and the intuitive generation of the surface from the control mesh by a process of cutting with hyperplanes, known as corner cutting or subdivision. The step from $C^1$- to $C^2$-surface splines is both difficult and important since it requires a better understanding of the foundations of smooth surface constructions. While for $C^1$ surfaces intuition and hand calculation usually suffice to compute a representation in terms of Bernstein-Bézier coefficients, higher-order surface splines need to be tackled with a more abstract approach in terms of reparametrizations and solutions of classes of difference equations. Expressing the splines in terms of Bernstein-Bézier coefficients can then be left to a generic symbol manipulation routine. The specific formulas may be viewed as an assembler-level representation of the surface that will ultimately be of concern only to the specialist.

The surface splines presented here smooth a general, regular or irregular mesh of points into a $C^2$ surface parametrized by at most octic triangular patches. The formulas yield degree-raised sextic rather than quartic pieces when the mesh is locally regular, even though the reparametrization is the identity. One can explicitly recognize this case and place quartic box-spline patches instead. However, this would not be elegant. – Input meshes with the same connectivity and the same blend ratio for corresponding cells give rise to a vector space of surface splines. This and the convex hull property are useful for approximating and locally editing the spline surface. Detailed control over the curvature in directions corresponding to mesh edges can be exerted via blend ratios. The limit case, zero blend ratios, result in a $C^0$ surface that tightly interpolates the input mesh. It is also possible to interpolate the input without loosing smoothness and without solving a global sparse system of equations. The mechanism is analogous to interpolation by a quadratic spline at every second knot.

Due to the built-in smoothness, the representation reduces the number of unknowns for shape improvement of smooth surfaces and similar differential equations on surfaces. In particular, since many notions of shape are linked to the distribution of curvature it is sensible to work with a curvature continuous representation without having to enforce this condition explicitly. Moreover, as in the $C^1$ case, $C^2$-surface splines have explicit parameters, called blend ratios, that govern the depth and distribution of cuts for generating the surface from the control mesh. Such cuts are closely related to the overall distribution of curvature and therefore give hope that one might control curvature in some detail. Last but not least, surface splines are useful to smooth and blend the boundary representation of a solid model (c.f. http://www.cs.purdue.edu/people/jorg).

## REFERENCES


C. Bajaj, I. Ihm, and J. Warren, *Higher Order Interpolation and Least Squares Approximation Using Implicit Algebraic Surfaces*, ACM Transactions on Graphics 12, 4, 1993, 327 - 347.

C. Bajaj and J. Chen and G. Xu, *Interactive Modelling with A-Patches*, Computer Science Technical Report, CAPO-93-02, Purdue University, 1993.

W. Boehm, *Generating the Bézier points of triangular splines*, R.E. Barnhill, W. Boehm (eds.), North Holland, 1983, 77–91.

W. Boehm, G. Farin, J. Kahmann, *A survey of curve and surface methods in CAGD*, Computer Aided Geometric Design 1 (1984), 43–.

C. W. de Boor, K. Höllig, S. Riemenschneider, *Box splines*, Springer Verlag, NY, 1994.

E. Catmull, J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, CAD 10, No 6 (1978): 350–355.

W. Dahmen, C.A. Micchelli, H.P. Seidel, *Blossoming begets B-splines bases built better by B-patches*, Mathematics of Computation, Vol. 59, No. 199, July 1992: 97–115.

W. Dahmen, T.-M. Thamm-Schaar, *Cubicoids: modeling and visualization*, Computer Aided Geometric Design , Vol. 10, 1993, 93-108.

D. Doo, *A subdivision algorithm for smoothing down irregularly shaped polyhedrons*, Proceedings on interactive techniques in computer aided design, Bologna (1978): 157–165.

N. Dyn, D. Levin, D. Liu, *Interpolatory convexity preserving subdivision schemes for curves and surfaces*, preprint, 1992.

G. Farin, *Curves and surfaces for computer aided geometric design*, Academic Press, 1990.

T.N.T. Goodman, *Closed surfaces defined from biquadratic splines*, Constructive Approximation 7 1991, 149–160.

J.A. Gregory, *Smooth parametric surfaces and n-sided patches*, Computation of curves and Surfaces, W. Dahmen, M. Gasca and C.A. Micchelli, eds., Kluwer Academic Publishers, Dordrecht, 1990: 457–498.

B. Guo, *Modeling arbitrary smooth objects with algebraic surfaces*, PhD thesis, Computer Sciences, Cornell University, 1991.

H. Hagen, H. Pottmann, *Curvature continuous triangular interpolants*, in Mathematical Methods in CAGD (T. Lyche, L.L. Schumaker eds.), Boston, Academic Press, 373–384.

J. M. Hahn, *Filling polygonal holes with rectangular patches*, Theory and Practice of geometric modeling, W. Straßer and H.-P. Seidel eds., Springer 1989.

K. Höllig, H. Mögerle, *G-splines*, Computer Aided Geometric Design 7 (1989): 197–207.

Closed smooth piecewise bicubic surfaces, S.L. Lee, A. A. Majid, ACM Transactions on Graphics 10,4 (1991): 342–365.

C. Loop, *Smooth subdivision surfaces based on triangles*, Master's thesis, University of Utah, 1987.

C. Loop, T. DeRose, *Generalized B-spline surfaces of arbitrary topology*, Computer Graphics 24,4(1990): 347–356.

C. T. Loop, *Smooth spline surfaces over irregular meshes*, Proceedings of Siggraph, 1994.

G. M. Nielson, *The side-vertex method for interpolation on triangles*, J. Approx. Theory, 25, 318–336.

J. Peters, *Joining smooth patches at a vertex to form a $C^k$ surface*, Computer Aided Geometric Design, 9 (1992) 387-411.

J. Peters *Constructing $C^1$ surfaces of arbitrary topology using biquadratic and bicubic splines*, in Designing fair curves and surfaces, N. Sapidis (ed.), 1994.

J. Peters, *Smooth free-form surfaces over irregular meshes generalizing quadratic splines*, Computer Aided Geometric Design, 10 (1993) 347–361.

J. Peters, *A characterization of connecting maps as roots of the identity*, The Mathematics of Curves and Surfaces II, P.J. Laurent, L.L.Schumaker (eds.), 1994

J. Peters, *Smooth splines over irregular meshes built from few polynomial pieces of low degree*, CSD-TR-93-019, March 1993, to appear as *$C^1$ free-form surface splines*, SIAM J. of Numerical Analysis **32**-2,1995.

J. Peters, *Smoothing vertex-degree bounded polyhedra*, to appear in ACM Transactions on Graphics.

A.R.M. Piah, *Construction of smooth surfaces by piecewise tensor product polynomials*, CS Report 91/04 University of Dundee, UK, 1991.

U. REIF *Biquadratic G-spline surfaces*, Preprint 93-4, Math Inst A, Universität Stuttgart,1993, to appear in CAGD.

M. SABIN, *The use of piecewise forms for the numerical representation of shape*, PhD thesis, Hungarian Academy of Sciences, Budapest, Hungary, 1976.

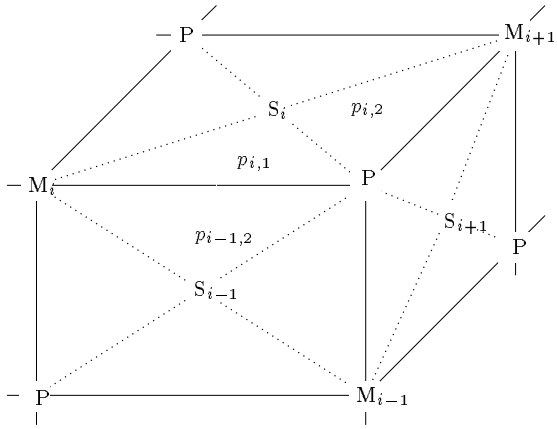
M. SABIN, *Non-rectangular surface patches suitable for inclusion in a B-spline surface*, in P. ten Hagen (ed.), Proceedings of Eurographics '83, North Holland, 57–69.

H.-P. SEIDEL, *Symmetric recursive algorithms for surfaces: B-patches and the de Boor algorithm for polynomials over triangles*, Constr. Approx. 7(1991): 257–279.

**Appendix 1. Labels for adjacent patches and their coefficients.**

For generation and implementation, it is useful to give two labels to the patches surrounding a vertex. Consider therefore the patches $p_{i,j}$ surrounding a point of type P. The index $i$ counts, in clockwise order, pairs of patches joined across an edge PS, while the index $j$ indicates the ordering within the pair. Throughout the paper indices are to be interpreted modulo $n$ and all capitalized coefficients $V$, $C$, $P$, etc. are points in space.

Fig. A.1:




Since each Bernstein-Bézier coefficient has naturally three indices, each patch of degree $d$ has a representation

$$p_{ij}(u,v) := \sum_{k+l+m=d} P_{klm,ij} \frac{d!}{k!l!m!} u^k v^l w^m, \quad u+v+w=1.$$

Associating $d00$ with mesh points of type P, $0d0$ with mesh points of type M, and $00d$ with mesh points of type S, allows expressing continuity between the patches by identifying

$$P_{k0m,i,1} = P_{k0m,i,2} \quad \text{and} \quad P_{km0,i,1} = P_{km0,i-1,2}.$$

Subscripts may be dropped when the coefficient is sufficiently identified by the context.

In Step 3.2 of the algorithm, boundary curves of degree 4 are generated. Their indices are schematically listed as

M → 040 130 220 310 400 ← P

The sextic patch generated in Step 3.6 generates coefficients with the following indices

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 006 | | | | | | | |
| | | | | | 015 | | 105 | | | | | | |
| | | | | 024 | | 114 | | 204 | | | | | |
| | | | 033 | | 123 | $_{i,1}$ | 213 | | 303 | | | | |
| | | 042 | | 132 | | 222 | | 312 | | 402 | | | |
| | 051 | | 141 | | 231 | | 321 | | 411 | | 501 | | |
| M 060 | −− | 150 | −− | 240 | −− | 330 | −− | 420 | −− | 510 | −− | 600 P | |
| | 051 | | 141 | | 231 | | 321 | | 411 | | 501 | | |
| | | 042 | | 132 | | 222 | | 312 | | 402 | | | |
| | | | 033 | | 123 | $_{i-1,2}$ | 213 | | 303 | | | | |
| | | | | 024 | | 114 | | 204 | | | | | |
| | | | | | 015 | | 105 | | | | | | |
| | | | | | | 006 | | | | | | | |

The octic patch has the following indices, some of which had to be typeset lower to fit the margin.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 008 | | | | | | | | |
| | | | | | 017 | | 107 | | | | | $p_{i,2}$ | | |
| | | | | 026 | | 116 | | 206 | | | | | | |
| | | | 035 | | 125 | | 215 | | 305 | | | | | |
| | | 044 | | 134 | | 224 | | 314 | | 404 | | | | |
| | 053 | | 143 | | 233 | $_{i,1}$ | 323 | | 413 | | 503 | | | |
| | | 152 | | 242 | | 332 | | 422 | | 512 | | 602 | | |
| | 161 | | 251 | | 341 | | 431 | | 521 | | 611 | | 701 | |
| | $--$ | 260 | $--$ | 350 | $--$ | 440 | $--$ | 530 | $--$ | 620 | $--$ | 710 | $--$ | P |
| | 161 | | 251 | | 341 | | 431 | | 521 | | 611 | | 701 | |
| | | 152 | | 242 | | 332 | | 422 | | 512 | | 602 | | |
| | 053 | | 143 | | 233 | $_{i-1,2}$ | 323 | | 413 | | 503 | | | |
| | | 044 | | 134 | | 224 | | 314 | | 404 | | | | |
| | | | 035 | | 125 | | 215 | | 305 | | | | | |
| | | | | 026 | | 116 | | 206 | | | | | | |
| | | | | | 017 | | 107 | | | | | $_{i-1,1}$ | | |
| | | | | | | 008 | | | | | | | | |

| | | | |
|---|---|---|---|
| | | 602 | |
| | 071 | | |
| M | $--$ | 170 | . |
| | 071 | | |
| | | 602 | |

**Appendix 2. BB coefficients in terms of intermediate control points.**

```
P710[j]          := 1 over    2
  P400[]           1
  P310[j]          1
  P310[j+1]        0
  P220[j-1]        0
  P220[j]          0
  P220[j+1]        0
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
P701[j]          := 1 over    4
  P400[]           2
  P310[j]          1
  P310[j+1]        1
  P220[j-1]        0
  P220[j]          0
  P220[j+1]        0
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
P620[j]          := 1 over    14
  P400[]           3
  P310[j]          8
  P310[j+1]        0
  P220[j-1]        0
  P220[j]          3
  P220[j+1]        0
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
P602[j]          := 1 over    112*c
  P400[]           -4*c-8*c^3+32*c^2+12
  P310[j]          56*c-36*c^2+8*c^3
  P310[j+1]        -8*c^2-12+48*c
  P220[j-1]        -3
  P220[j]          6*c+12*c^2
  P220[j+1]        3+6*c
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
P611[j,1]        := 1 over    112*c
  P400[]           -4*c-8*c^3+32*c^2+12
  P310[j]          72*c-36*c^2+8*c^3
  P310[j+1]        -8*c^2-12+32*c
  P220[j-1]        -3
  P220[j]          12*c+12*c^2
  P220[j+1]        3
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
P611[j-1,2]      := 1 over    112*c
  P400[]           84*c+8*c^3-48*c^2-12
  P310[j]          48*c+28*c^2-8*c^3
  P310[j+1]        8*c^2+12-32*c
  P220[j-1]        3
  P220[j]          12*c+12*c^2
  P220[j+1]        -3
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
@
```

```
P521[j,1]        := 1 over    168
  P400[]           -18+45*c-5*c^2
  P310[j]          -52*c+84+5*c^2
  P310[j+1]        42-5*c
  P220[j-1]        0
  P220[j]          6*c+54
  P220[j+1]        0
  P130[j]          6*c+6
  P130[j+1]        0
  P3bs[]           0
@
P521[j-1,2]      := 1 over    168
  P400[]           66-49*c+5*c^2
  P310[j]          84+32*c-5*c^2
  P310[j+1]        5*c-42
  P220[j-1]        0
  P220[j]          6*c+54
  P220[j+1]        0
  P130[j]          6*c+6
  P130[j+1]        0
  P3bs[]           0
@
P3bs[j]          := 1 over    336*c
  P400[]           228*c-175*c^2+105*c^3-16*c^4-36
  P310[j]          72+16*c^4-342*c-105*c^3+175*c^2
  P310[j+1]        0
  P220[j-1]        0
  P220[j]          -36-6*c^2+108*c
  P220[j+1]        0
  P130[j]          6*c+6*c^2
  P130[j+1]        0
  P3bs[]           0
@
P512[j,1]        := 1 over    336*c
  P400[]           -105*c^3+171*c^2-180*c+16*c^4+36
  P310[j]          -36+276*c+55*c^3-121*c^2-8*c^4
  P310[j+1]        -8*c^4+50*c^3+234*c-36-74*c^2
  P220[j-1]        0
  P220[j]          18+12*c^2+27*c
  P220[j+1]        18+6*c^2-27*c
  P130[j]          6*c+6*c^2
  P130[j+1]        0
  P3bs[]           336*c
@
P512[j-1,2]      := 1 over    336*c
  P400[]           -36+16*c^5+360*c-445*c^2+143*c^3-100*c^4
  P310[j]          347*c^2-16*c^5+92*c^4-93*c^3-36+204*c
  P310[j+1]        8*c^4-234*c+74*c^2+36-50*c^3
  P220[j-1]        18+6*c^2-27*c
  P220[j]          18+12*c^2+27*c
  P220[j+1]        0
  P130[j]          6*c+6*c^2
  P130[j+1]        0
  P3bs[]           336*c
@
P503[j]          := 1 over    672*c
  P400[]           72+342*c^2-210*c^3+32*c^4-360*c
  P310[j]          -72-195*c^2+510*c+105*c^3-16*c^4
  P310[j+1]        -72-195*c^2+510*c+105*c^3-16*c^4
  P220[j-1]        0
  P220[j]          36+18*c^2
  P220[j+1]        36+18*c^2
  P130[j]          6*c+6*c^2
  P130[j+1]        6*c+6*c^2
  P3bs[]           672*c
@
```

**Table 1** Coefficients of the curvature cap at P, $n \neq 4$.

The meaning of the table indices is apparent from the following example: $P_{701,j} := \frac{1}{2}P_{400} + \frac{1}{4}(P_{710,j} + P_{710,j+1})$. As usual, the indices are counted modulo $n$ and $c := \cos(\frac{2\pi}{n})$. The average of the coefficients $P_{5,jk}$, P3bs[ ], is free to choose; the default is computed as

$$B_{220} := \frac{1}{n}\sum P_{220,i},$$

$$B_{130} := \frac{1}{n}\sum P_{130,i},$$

$$\text{P3bs[ ]} := \frac{-1}{56c}(-55c + 4c^2 + 6)P_{400} + \frac{3}{56c}(c^2+2)B_{220} + \frac{1}{56}(c+1)B_{130}$$

```
P170[j]          := 1 over    8
  P220[j+0]        1
  P220[j+1]        0
  P220[j+2]       -1
  P220[j+3]        0
  P040[]           8
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P071[j]          := 1 over    16
  P220[j+0]        1
  P220[j+1]        1
  P220[j+2]       -1
  P220[j+3]       -1
  P040[]           16
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P260[j]          := 1 over    14
  P220[j+0]        5
  P220[j+1]        0
  P220[j+2]       -2
  P220[j+3]        0
  P040[]           11
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P062[j]          := 1 over    112
  P220[j+0]        17
  P220[j+1]        17
  P220[j+2]       -11
  P220[j+3]       -11
  P040[]           100
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P161[j-1,2]      := 1 over    112
  P220[j+0]        27
  P220[j+1]       -7
  P220[j+2]       -15
  P220[j+3]        7
  P040[]           100
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P161[j,1]        := 1 over    112
  P220[j+0]        27
  P220[j+1]        7
  P220[j+2]       -15
  P220[j+3]       -7
  P040[]           100
  P310[j+0]        0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
```

```
P350[j]          := 1 over    28
  P220[j]          15
  P220[j+1]        0
  P220[j+2]       -3
  P220[j+3]        0
  P040[]           14
  P310[j]          2
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P251[j-1,2]      := 1 over    8064
  P220[j+0]        3*c[1]-3*c[3]+6*c[3]^2-24*c[0]-6*c[1]^2+3600
  P220[j+1]        -17*c[0]-2*c[2]^2-4*c[1]+c[2]-4*c[3]-504+34*c[0]^2
  P220[j+2]        -3*c[1]+3*c[3]-6*c[3]^2+24*c[0]-1008+6*c[1]^2
  P220[j+3]        17*c[0]-c[2]+2*c[2]^2+4*c[1]+4*c[3]+504-34*c[0]^2
  P040[]           5184
  P310[j+0]        288
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P251[j,1]        := 1 over    8064
  P220[j+0]        -3*c[1]+3*c[3]-6*c[3]^2-24*c[0]+6*c[1]^2+3600
  P220[j+1]        17*c[0]-c[2]+2*c[2]^2+4*c[1]+4*c[3]+504-34*c[0]^2
  P220[j+2]        3*c[1]-3*c[3]+6*c[3]^2+24*c[0]-1008-6*c[1]^2
  P220[j+3]        -17*c[0]-2*c[2]^2-4*c[1]+c[2]-4*c[3]-504+34*c[0]^2
  P040[]           5184
  P310[j+0]        288
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P152[j-1,2]      := 1 over    8064
  P220[j+0]        -5*c[3]-c[2]+10*c[3]^2-7*c[0]-2*c[1]^2+2718+c[1]
  P220[j+1]        -846-c[1]-5*c[0]-7*c[3]+c[2]-2*c[2]^2+10*c[0]^2
  P220[j+2]        -1098+5*c[3]-10*c[3]^2-c[1]+7*c[0]+c[2]+2*c[1]^2
  P220[j+3]        c[1]+5*c[0]-c[2]+1170+7*c[3]-10*c[0]^2+2*c[2]^2
  P040[]           5976
  P310[j+0]        144
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P152[j,1]        := 1 over    8064
  P220[j+0]        -5*c[1]+c[3]-2*c[3]^2-7*c[0]-c[2]+10*c[1]^2+2718
  P220[j+1]        5*c[0]+2*c[2]^2+7*c[1]-c[2]+c[3]+1170-10*c[0]^2
  P220[j+2]        5*c[1]-c[3]+2*c[3]^2+7*c[0]+c[2]-1098-10*c[1]^2
  P220[j+3]        -5*c[0]+c[2]-2*c[2]^2-7*c[1]-c[3]-846+10*c[0]^2
  P040[]           5976
  P310[j+0]        144
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
@
P053[j]          := 1 over    224
  P220[j+0]        54
  P220[j+1]        54
  P220[j+2]        -27
  P220[j+3]        -27
  P040[]           166
  P310[j+0]        2
  P310[j+1]        2
  P310[j+2]        0
  P310[j+3]        0
@
```

**Table 2** Coefficients of the curvature cap at M.

```
P530[]          := 1 over    14
     P400[]       1
     P310[]       6
     P220[]       6
     P130[]       1
     P040[]       0
@
P440[]          := 1 over    70
     P400[]       1
     P310[]      16
     P220[]      36
     P130[]      16
     P040[]       1
@
P350[]          := 1 over    14
     P400[]       0
     P310[]       1
     P220[]       6
     P130[]       6
     P040[]       1
@
P341[j]         := 1 over    280
     P800[j]     -15
     P710[j]      64
     P620[j]     -84
     P440[j]      70
     P260[j]     -84
     P170[j]      64
     P080[j]     -15
     P701[j]      24
     P611[j]     -112
     P521[j]      168
     P251[j]      336
     P161[j]     -168
     P071[j]      32
@
P431[j]         := 1 over    280
     P800[j]     -15
     P710[j]      64
     P620[j]     -84
     P440[j]      70
     P260[j]     -84
     P170[j]      64
     P080[j]     -15
     P701[j]      32
     P611[j]     -168
     P521[j]      336
     P251[j]      168
     P161[j]     -112
     P071[j]      24
@
```

```
P422[j]         := 1 over    1680
     P800[j]      219*c-96*c^2-165
     P710[j]      864-1040*c+448*c^2
     P620[j]      -1764-672*c^2+1596*c
     P440[j]      -630-630*c
     P260[j]      -84-84*c
     P170[j]      144*c+144
     P080[j]      -45*c-45
     P701[j]      192-192*c+96*c^2
     P611[j]      896*c-448*c^2-1008
     P521[j]      3696+672*c^2-1344*c
     P251[j]      1008+1008*c
     P161[j]      -672-672*c
     P071[j]      144*c+144
     Remainder            K422
@
P332[j]         := 1 over    2240
     P800[j]      -27*c-240+18*c^2
     P710[j]      144*c-96*c^2+1024
     P620[j]      -1344-252*c+168*c^2
     P440[j]      -1120-420*c^2+630*c
     P260[j]      -1344-364*c-56*c^2
     P170[j]      -16*c+1024+96*c^2
     P080[j]      -30*c^2-240+45*c
     P701[j]      448
     P611[j]      -2240
     P521[j]      4032
     P251[j]      672*c^2+4032-1008*c
     P161[j]      -2240+1120*c-448*c^2
     P071[j]      -272*c+96*c^2+448
     Remainder            K332
@
P242[j]         := 1 over    1680
     P800[j]      -45
     P710[j]      144
     P620[j]      -84
     P440[j]      -630
     P260[j]      -224*c^2-1764+224*c
     P170[j]      -144*c+64*c^2+864
     P080[j]      -165
     P701[j]      144
     P611[j]      -672
     P521[j]      1008
     P251[j]      3696
     P161[j]      -1008+224*c^2-224*c
     P071[j]      144*c+192-64*c^2
     Remainder            K242
@
```

**Table 3** Coefficients of the curvature corridor PM.
The remainder terms are freely choosable vectors with default zero.

```
P132[j]          := 1 over    8640
     P800[j]       1086*c^2-2379*c
     P710[j]       11408*c-5152*c^2-1792
     P620[j]       7896*c^2-17724*c+8736
     P440[j]       -2940*c^2+10710*c
     P260[j]       -2604*c+672+504*c^2
     P170[j]       3680+416*c^2-976*c
     P080[j]       -1728+765*c-210*c^2
     P701[j]       1792+1920*c-960*c^2
     P611[j]       -6944+4480*c^2-8960*c
     P521[j]       -8736+13440*c-6720*c^2
     P251[j]       -672+4704*c^2-17136*c
     P161[j]       -17024-4032*c^2+15456*c
     P071[j]       928*c^2-3920*c+6688
     P602[j]       -1792
     P512[j]       8736
     P152[j]       24864
     P062[j]       -7840
     Remainder             -35/18*K422-7/9*K242+49/27*K332
@
P222[j]          := 1 over    2160
     P800[j]       411*c-174*c^2-270
     P710[j]       800*c^2-1936*c+1760
     P620[j]       -3864-1176*c^2+2940*c
     P440[j]       -3780-420*c^2-630*c
     P260[j]       -3864-84*c-504*c^2
     P170[j]       -16*c+1760+224*c^2
     P080[j]       -270-45*c-30*c^2
     P701[j]       -384*c-32+192*c^2
     P611[j]       -896*c^2-1568+1792*c
     P521[j]       -2688*c+9408+1344*c^2
     P251[j]       9408+672*c^2+1008*c
     P161[j]       -1568-672*c
     P071[j]       304*c-32-32*c^2
     P602[j]       896
     P512[j]       -3360
     P152[j]       -3360
     P062[j]       896
     Remainder             14/9*K422+14/9*K242+28/27*K332
@
P312[j]          := 1 over    8640
     P800[j]       510*c^2-1065*c-1728
     P710[j]       5168*c-2464*c^2+3680
     P620[j]       3864*c^2-8148*c+672
     P440[j]       -2940*c^2+6930*c
     P260[j]       -4452*c+8736+1848*c^2
     P170[j]       -1792+32*c^2+752*c
     P080[j]       495*c-210*c^2
     P701[j]       6688+768*c-384*c^2
     P611[j]       -17024+1792*c^2-3584*c
     P521[j]       -672+5376*c-2688*c^2
     P251[j]       -8736+4704*c^2-11088*c
     P161[j]       -6944-5376*c^2+12768*c
     P071[j]       1312*c^2-3920*c+1792
     P602[j]       -7840
     P512[j]       24864
     P152[j]       8736
     P062[j]       -1792
     Remainder             -7/9*K422-35/18*K242+49/27*K332
@
P051[j]          := 1 over    3
     P800[j]       0
     P710[j]       0
     P620[j]       0
     P440[j]       0
     P260[j]       0
     P170[j]       0
     P080[j]       -1
     P701[j]       0
     P611[j]       0
     P521[j]       0
     P251[j]       0
     P161[j]       0
     P071[j]       4
@
P141[j]          := 1 over    3
     P800[j]       0
     P710[j]       0
     P620[j]       0
     P440[j]       0
     P260[j]       0
     P170[j]       0
     P080[j]       -1
     P701[j]       0
     P611[j]       0
     P521[j]       0
     P251[j]       0
     P161[j]       0
     P071[j]       4
@
P231[j]          := 1 over    15
     P800[j]       0
     P710[j]       0
     P620[j]       0
     P440[j]       0
     P260[j]       -14
     P170[j]       12
     P080[j]       -3
     P701[j]       0
     P611[j]       0
     P521[j]       0
     P251[j]       42
     P161[j]       -28
     P071[j]       6
@
P321[j]          := 1 over    15
     P800[j]       -3
     P710[j]       12
     P620[j]       -14
     P440[j]       0
     P260[j]       0
     P170[j]       0
     P080[j]       0
     P701[j]       6
     P611[j]       -28
     P521[j]       42
     P251[j]       0
     P161[j]       0
     P071[j]       0
@
P411[j]          := 1 over    15
     P800[j]       3
     P710[j]       -8
     P620[j]       0
     P440[j]       0
     P260[j]       0
     P170[j]       0
     P080[j]       0
     P701[j]       -8
     P611[j]       28
     P521[j]       0
     P251[j]       0
     P161[j]       0
     P071[j]       0
@
P501[j]          := 1 over    3
     P800[j]       -1
     P710[j]       0
     P620[j]       0
     P440[j]       0
     P260[j]       0
     P170[j]       0
     P080[j]       0
     P701[j]       4
     P611[j]       0
     P521[j]       0
     P251[j]       0
     P161[j]       0
     P071[j]       0
@
```

**Table 4** Coefficients of a sextic patch $C^2$ across boundaries connecting to S.

## Appendix 3. Routines for checking the $C^2$ continuity of the surface across edges P, M in terms of its Bernstein-Bézier coefficients.

```
#
# MAPLE file:   bb1mult
# runs under:   Maple V Release 2
# author:       J"org Peters, jorg@cs.purdue.edu
# date:         Jan 1994
#
# multiply two univariate polynomials in BB-form
#
#-------------------------------------------------------------------

bb1mult := proc(b1, d1, b2, d2)

# bi: array(0..di)
# di : degree of bi
#-------------------------------------------------------------------

local ii,i1,i2, dgout,out:
if nargs <> 4 then
  ERROR(‘wrong number of arguments‘): fi:

dgout := d1 + d2:
out := array(0..dgout):

for ii from 0 to dgout do
    out[ii] := 0;
od:
for i1 from 0 to d1 do
  for i2 from 0 to d2 do
    mult := binomial(d1,i1)*binomial(d2,i2)/binomial(d1+d2,i1+i2);
    out[i1+i2] := out[i1+i2] + mult* b1[i1] * b2[i2]:
od:od:
out;
end:
#-------------------------------------------------------------------
```

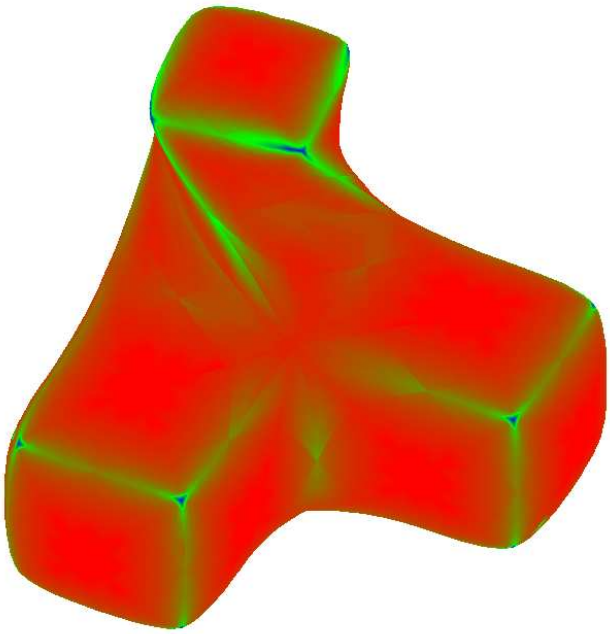
```
#
# MAPLE file:   chk
# runs under:   Maple V Release 2
# author:       J"org Peters, jorg@cs.purdue.edu
# date:         Jan 1994
#-------------------------------------------------------------
#
# check C1 and C2 conditions between the patches
#  p  and  q  and  across the splitting edges in the
# neighborhood of M and P
# (between p and r, respectively p and s)
#
#
#      s  /            \  r
#        /       p       \
#       M-----------------P
#        \       q       /
#         \             /
#
    d := 8:     # degree of the patches

# read the coefficients relevant for the C1 and C2 conditions
# the coefficients are expressed in terms of 14 variables:
#                                 coefficients determining the
#       p[4,0,0],               --- position of P
#       p[3,1,0], p710p,        --- tangent plane at P
#       p220m, p[2,2,0], p220p, --- curvature at P
#       p[0,4,0],               --- position of M
#       P220[1],P220[2], P220[3] together with p[2,2,0]
#                               --- curvature and tangent at M
#       base3,                  --- nullspace of 3rd layer at P
#       K242,K332,K422          --- nullspace of the 3 C2
#                                   constraints not associated
#                                   with the 3 disks at P and M
#
    read PMlayer0:
    p[1,3,0] := p[0,4,0] + (p[2,2,0]-P220[2])/4:
    read atP:
    read atM:
    read PMlayer12:
    read bb1mult:
```

```
# Dip   contains the coefficients of (1/d)D_i p
    D1p := array(0..d-1):
    D2p := array(0..d-1):
    D2q := array(0..d-1):
    for i from 0 to d-1 do
        D1p[i] := simplify(p[i+1,d-i-1,0]-p[i,d-i,0]);
        D2p[i] := simplify(p[i,d-i-1,1]-p[i,d-i,0]);
        D2q[i] := simplify(q[i,d-i-1,1]-q[i,d-i,0]);
        D2[i] := simplify(D2p[i]-D2q[i]);
    od:

# Dijp  contains the coefficients of  (1/(d*(d-1))) D_iD_j p
    D12  := array(0..d-2):
    D22  := array(0..d-2):
    for i from 0 to d-2 do
        D12p := p[i+1,d-i-2,1]-p[i+1,d-i-1,0]
               -p[i,d-i-1,1]+p[i,d-i,0];
        D22p := p[i,d-i-2,2]-2*p[i,d-i-1,1]+p[i,d-i,0];
        D12q := q[i+1,d-i-2,1]-q[i+1,d-i-1,0]
               -q[i,d-i-1,1]+q[i,d-i,0];
        D22q := q[i,d-i-2,2]-2*q[i,d-i-1,1]+q[i,d-i,0];
        D22[i] := simplify(D22p-D22q);
        D12[i] := simplify(D12p-D12q);
    od:

# derivatives of the connecting map f and the degree raising polynomial
    D2f1 := array(0..2):
    D2f1[0] := 1: D2f1[1] := 1: D2f1[2] := 1-c:
    D22f1 := array(0..1):
    D22f1[0] := 0: D22f1[1] := c*(1-2*c):
    rais := array(0..2):
    rais[0] := 1: rais[1] := 1: rais[2] := 1:

# D2p=D2q across the splitting edges
    ‘error in C1 conditions across splitting edges in the 3 disk at P and M‘;
    c1PS := array(0..2):
    c1MS := array(0..2):
    for i from 0 to 2 do
        ii := d-1-i:
        c1PS[i] := simplify(p[ii,0,1+i]-p[ii,1,i]
                           -(r[ii,1,i]-r[ii,0,1+i])):
        c1MS[i] := simplify(p[0,ii,1+i]-p[1,ii,i]
                           -(s[1,ii,i]-s[0,ii,1+i])):
    od:
    print(c1PS,c1MS);

# D22p=D22q across the splitting edges
    ‘error in C2 conditions across splitting edges in the 3 disk at P and M‘;
    c2PS := array(0..1):
    c2MS := array(0..1):
    for i from 0 to 1 do
        ii := d-2-i:
        c2PS[i] := simplify(p[ii,0,2+i]-2*p[ii,1,1+i]+p[ii,2,i]
                    -(r[ii,0,2+i]-2*r[ii,1,1+i]+r[ii,2,i])):
        c2MS[i] := simplify(p[0,ii,2+i]-2*p[1,ii,1+i]+p[2,ii,i]
                    -(s[0,ii,2+i]-2*s[1,ii,1+i]+s[2,ii,i])):
    od:
    print(c2PS,c2MS);

# D2p+D2q= D2f1*D1p across the edge PM
    ‘error in C1 conditions across the edge P M‘;
    u := bb1mult(D1p,d-1,D2f1,2):
    v := bb1mult(D2p,d-1,rais,2):
    w := bb1mult(D2q,d-1,rais,2):
    c1err := array(0..d+1):
    for j from 0 to d+1 do
        c1err[j] := simplify(u[j]-(v[j]+w[j])):
    od:
    print(c1err);

# D22p-D2f1*D12p+D22f1*D2p = D22q-D2f1*D12q+D22f1*D2q across the edge PM
    ‘error in C2 conditions across the edge P M‘;
    uv := bb1mult(D12,d-2,D2f1,2):
    vv := bb1mult(D22,d-2,rais,2):
    vw := bb1mult(D2,d-1,D22f1,1):
    c2err := array(0..d):
    for j from 0 to d do
        c2err[j] := simplify(vv[j]-uv[j]+vw[j]/(d-1)):
    od:
    print(c2err);
```

**Table 5** Maple routines for checking curvature continuity.

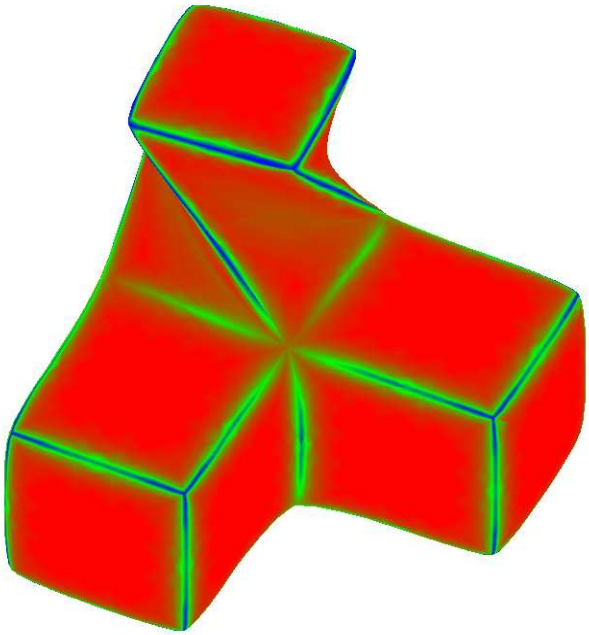
**Appendix 4. Maximal absolute curvature and blend ratios.** The following figures show maximal absolute curvature measured at the vertices of the three-sided patches after 3-fold subdivision and displayed with the same color scale.

**Fig. A4.1**



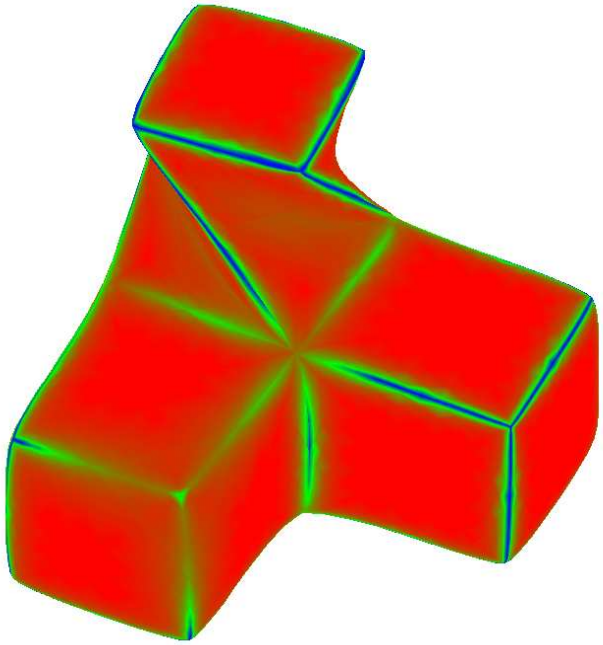
$\alpha_{ij} = 0.4$ globally

**Fig. A4.2**



$\alpha_{ij} = 0.25$ globally

**Fig. A4.3**

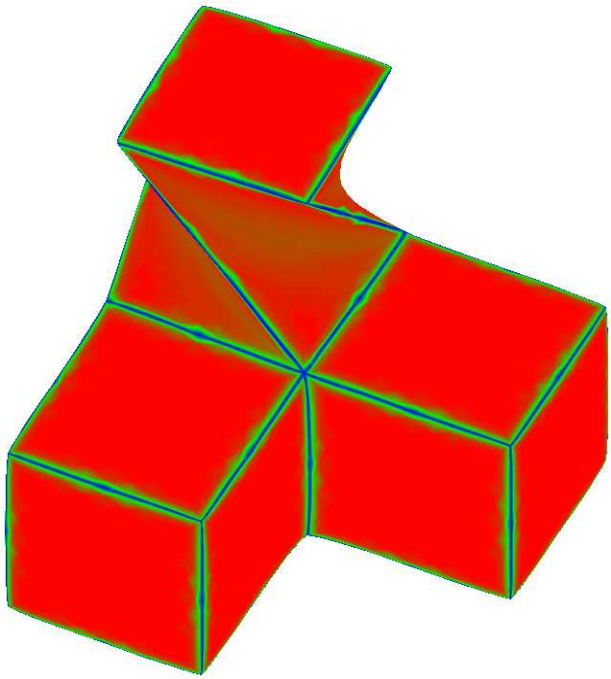


$\alpha_{ij} = 0.25$ globally except for the front corner of the left and the right cube.

The left cube's corner is smoothed to 0.5.

The right cube's corner is sharpened to 0.1

**Fig. A4.4**



$\alpha_{ij} = 0.10$ globally