1994

Curvature Continuous Spline Surfaces Over Irregular Meshes

Jörg Peters

Report Number:
94-040

Curvature Continuous Spline Surfaces Over Irregular Meshes

Jörg Peters
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-94-040
June, 1994

February 1994

1–20,i–ix

CURVATURE CONTINUOUS SPLINE SURFACES OVER IRREGULAR MESHES *

JORG PETERS †

Abstract. Concepts and techniques for the construction of spline surfaces over irregular meshes are developed and made concrete by exhibiting a family of 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. In particular, the mesh points serve as control points and are locally averaged to obtain the coefficients of a Bernstein-Bézier representation. The construction of C^2 surface splines in terms of three-sided Bernstein-Bézier patches is worked out in detail; the construction using four-sided patches or a mix of three-sided and four-sided patches is sketched.

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.

Splines assembled from B-splines, though widely used to represent functions and surfaces, can only model a small subclass of surfaces that arise in geometric modeling, because every interior point of the B-spline control mesh outlining the spline surface must be surrounded by exactly four quadrilateral cells. Hence the surface pieces must form a regular, checker board arrangement that allows only deformations of the plane and the torus to be modeled without singularity. Even then, the rigid structure prevents a natural modeling of frequent features such as suitcase corners or house corners, where three or five quadrilaterals meet. Special techniques like patch trimming and singular parametrization only *seem* to overcome the checker board restrictions; by destroying the consistent B-spline framework, in particular the built-in smoothness, these techniques only move the problem down the line, where it is even harder to address and leads to a maze of special cases.

Yet B-spline based splines have many desirable properties, such as a low degree polynomial or rational representation of maximal smoothness and a geometrically intuitive variation of the surface in terms of the coefficients. It is therefore desirable to devise a representation that removes the regularity restrictions from the input mesh, yielding a more unified approach to surface modeling, but reducing to the B-spline paradigm wherever the mesh is regular. A list of desirable properties of such a surface spline representation has been 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 the patch degree. For C^2 surfaces, such a construction is likely to be of the least degree [Peters '94, Sect.6]. The three-sided patches may be replaced by four-sided, biquartic,

* This work was supported by NSF grant CCR-9396164.

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

and bicubic tensor-product patches where the mesh is regular. The analoguous tensor-product construction has degree bi-six. The latter two constructions are sketched in this paper by giving the connecting-maps that define the spline space. As with C^1 surface splines [Peters '9xa], shape handles in the form of blend ratios are associated with each edge allowing for a local change of the normal and the curvature across the edge.

The derivation of the explicit formulas for Bernstein-Bézier coefficients in terms of the control mesh in this paper is somewhat unusual since this tedious task has been delegated to a symbolic routine whose input consists of a connecting map and a choice of representation of the nullspace of the vertex-enclosure difference equations (see Section 2 for the details). This top down approach allows experimenting with different algorithms and decreases the likelihood of typeos: the formulas in Appendix 2 are the direct output of the routine and are read by the C-language program that generates the surfaces and checks numerically that the resulting surface is C^2.

Related work. Building on work by [Sabin '83] and [Goodman '91], [Höllig, Mögerle '89] pioneered the idea of geometrically continuous spline spaces (see also [Piah '91], [Reif '93]). However, explicit representations of these G-splines in terms of say the Bernstein-Bézier form require the solution of large linear, irregularly sparse systems of equations to match data. This makes it difficult to predict the shape of the resulting surface. The Bernstein-Bézier coefficients of surface splines, in contrast, are given explicitly as local averages. 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 (cf. [de Boor, Höllig, Riemenschneider '94]), 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. Of the numerous algorithms based on these techniques, only the approach in [Hahn '89] constructs C^2 smooth surfaces, however at the cost of high 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 elegant new solutions to the smoothing problem at the cost of a presently non-standard patch representation.

Properties of free-form surface splines. Following [Peters '9xa], 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 polyno-

mial patches. The representation can be extended to rational patches by using a fourth coordinate.

- *simple interpolation.* Interpolation of input mesh points and normals can be done 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 analoguous 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.

Overview. Section 2 of this paper explains the concepts and techniques needed for the derivation and analysis of C^k surface splines. Specific C^2 surface splines are defined in Section 3, in terms of a Bernstein-Bézier representation. 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 the labelling scheme used for the coefficients of adjacent patches. 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. This program and the list of formulas in electronic form may be obtained from the author. Appendix 4 shows that the author has implemented the surface splines. Rather than showing a number of shapes, the variation of the curvature in terms of the blend ratios is illustrated.

2. Surface spline basics.

This section introduces the concepts and techniques used in the derivation and analysis of C^k surface splines. The general approach is made concrete in terms of the C^2 surface splines defined in Section 3. This section contains the motivation and derivation of the construction arranged under the following subheadings.

1. Control meshes and mesh refinement.
2. Symmetric G^k joins between 2 patches along an edge.
3. G^k joins among n patches at a vertex.
4. Degree bounds.
5. The nullspace of vertex-enclosure 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 polyhedral. 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 cells 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, the larger number of surface pieces for a lower degree of the polynomial surface For the case of C^1 surfaces, [Peters '9xb] defines a class of meshes that have a particularly simple smoothing algorithm and can be obtained from general meshes by refinement and projection.

Example The first step of the construction in Section 3 creates centroids, referred to as *vertices of type* P and edge midpoints referred to as *vertices of type* M. Note that the mesh is refined only once. 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.

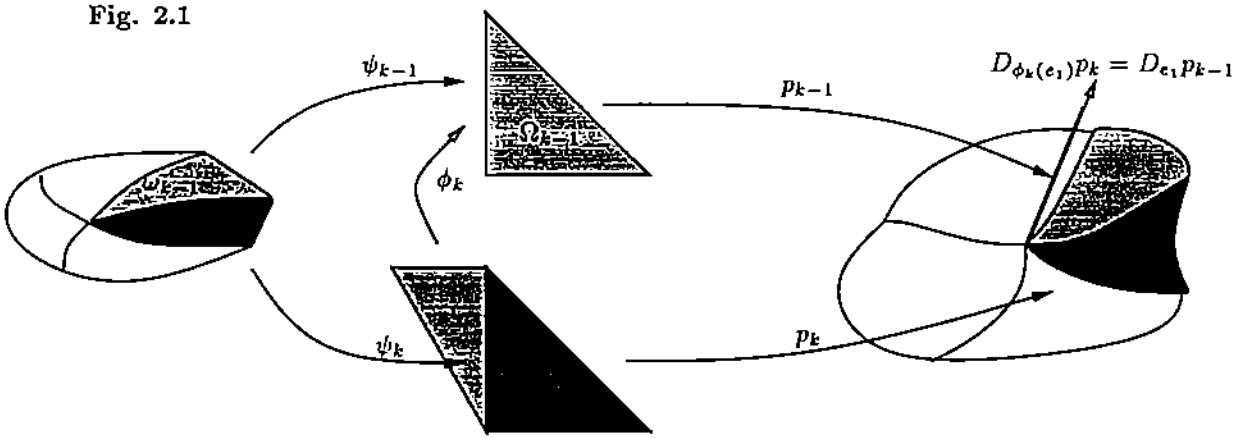
2.2. Symmetric G^k joins between 2 patches along an edge.

Following the definition in [Gregory '90] let p and q be C^kmaps from $\mathbb{R}^2$ to $\mathbb{R}^n$ and ϕ an invertible C^k map from $\mathbb{R}^2$ to $\mathbb{R}^2$ that connects the domains Ω_p and Ω_q of p and q by mapping a neighborhood of the domain edges $E_p \subset \Omega_p$ to a neighborhood of the domain edge $\phi(E_p) \subset \Omega_q$ while ϕ^{-1} maps points interior to Ω_q to exterior points of Ω_p; ϕ is called a connecting-map. As illustrated below, the connecting map ϕ_k can be thought of as the composition of two invertible maps ψ_k and ψ_{k-1}. Denote differentiation in the direction along E_p by D_1 and perpendicular to E_p by D_2. Then p and q join with G^k continuity if and only if along the edge E_p

$$D_2^m[p - q \circ \phi] \equiv 0 \qquad \text{for } m = 0..k, \tag{G^k}$$

where $\circ$ denotes composition of maps.

Fig. 2.1



$\mathbb{R}^2$ joint domain | $\mathbb{R}^{2\times n}$ patch domains | $\mathbb{R}^3$ surface

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. If the reparametrization depends only on the connectivity, the continuity constraints should reflect the symmetric role of p and q. By symmetry and reflection,

$$0 = D_2^m(p - q \circ \phi)(u, v)$$
$$0 = D_2^m(q - p \circ \phi)(u, -v)$$

yielding the *symmetrized G^k constraints*

$$0 = \frac{1}{2}D_2^m(p - q \circ \phi)(u, v) + (-1)^m \frac{1}{2}D_2^m(q - p \circ \phi)(u, -v).$$

When expanding the expression by applying the chain rule, some of the subexpression drop out, because $\phi^{[2]}(0, v) = -v$ and $D_1^m p = D_1^m q$ for $m = 0..k$, where $\phi^{[j]}$ is the jth component of ϕ.

Example $m = 1$. By symmetry, $D_2\phi^{[2]} \equiv -1$. Define $\lambda := D_2\phi^{[1]}$. Since $\phi(u,0) = (u,0)$, we may abreviate $Dq(\phi) = Dq$ in the expansion

$$\begin{aligned} 0 = D_2(p - q \circ \phi) &= D_2p - DqD_2\phi \\ &= D_2p - D_1qD_2\phi^{[1]} - D_2qD_2\phi^{[2]} \\ &= D_2p + D_2q - D_1q\lambda. \end{aligned}$$

From this, we read off the symmetrized G^1 constraint

$$D_2p + D_2q = D_1q\lambda. \tag{G1}$$

□

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

$$\begin{aligned} D_2^2p &= D_2^2(q(\phi)) \\ &= D^2q(D_2\phi, D_2\phi) + DqD_2^2\phi \\ &= D_1^2q(D_2\phi^{[1]})^2 + 2D_1D_2q(D_2\phi^{[1]})(D_2\phi^{[2]}) + D_2^2q(D_2\phi^{[2]})^2 \\ &\quad + D_1qD_2^2\phi^{[1]} + D_2qD_2^2\phi^{[2]} \\ &= D_1^2q\lambda^2 - 2D_1D_2q\lambda + D_2^2q + D_1qD_2^2\phi^{[1]} + D_2qD_2^2\phi^{[2]}. \end{aligned}$$

Subtracting the symmetric expansion for $D_2^2q = D_2^2(p(\phi))$, and observing that $D_1^mp = D_1^mq$, we get the symmetrized G^2 constraints

$$D_2^2p - D_2^2q - \lambda D_1D_2p + \lambda D_1D_2q = \frac{D_2q - D_2p}{2}D_2^2\phi^{[2]}.$$

With $\alpha := D_2^2\phi^{[2]}/2$, the constraints can be rewritten as

$$[D_2^2 - \lambda D_1D_2 + \alpha D_2]p = [D_2^2 - \lambda D_1D_2 + \alpha D_2]q. \tag{G2}$$

□

2.3. G^k joins among n patches at a vertex.

The smooth join of $n > 2$ patches at a vertex is qualitatively more difficult than the join between 2 patches. When three or more patches join smoothly at a common point, the pairwise continuity constraints between the patches form a cyclic system. Correspondingly, the composition of all n connecting-maps must map the initial domain to itself and must agree with the identity map, id, at the preimage of the common point of the patches up to the given order of continuity if at least one of the patches is non-singular at the point.

To make this requirement precise, let ω_k, $k = 1..n$ be a partition of a neighborhood of $o := (0,0)$ into not necessarily linear cones (cf. Figure 2.1). Let $\tilde{\psi}_k$ be an invertible C^k map that maps ω_k to $\Omega_k \subset \mathbb{R}^2$, the domain of the kth patch p_k. Then $\hat{\phi}_k := \tilde{\psi}_{k-1} \circ \tilde{\psi}_k^{-1}$ is a smooth invertible function that maps a region adjacent to the domain Ω_k to Ω_{k+1} and, in particular, $e_2 := (0,1)$ to $e_1 := (1,0)$. Clearly $\circ_{l=1}^n \hat{\phi}_l := \hat{\phi}_1 \circ \hat{\phi}_2 \circ \ldots \circ \hat{\phi}_n = \mathrm{id}$.

For the constructions to follow, we are interested in approximations ϕ_k to $\tilde{\phi}_k$. Let D_v denote differentiation in the direction of the vector v and, in particular, $D_i := D_{e_i}$. Since we will be exclusively concerned with the values of functions at o, we may whenever this is unambiguous, simply write the name of a function when we really mean that function's value at o. Thus, the Taylor jet of a function f, $J_r f := (D_1^m D_2^n f)_{m+n \le r}$ is an ordered collection of Taylor coefficients of a bivariate map f expanded at o up to the rth Taylor term. The approximation that we are intersted in satisfies $J_r \phi = J_r \tilde{\phi}$ and hence

$$J_r \text{ id} \simeq J_r(\circ_{l=1}^n \phi_l). \tag{C}$$

We call the latter condition the *circularity constraint* on the connecting-maps. [Peters '94] shows that after proper normalization the circularity constraints for a C^2 surface are as follows.

$$o \simeq \circ_{l=1}^n \phi_l, \tag{2.0}$$

$$e_i = D_i(\circ_{l=1}^n \phi_l) = (\prod_{l=1}^{n} D\phi_l)e_i, \tag{2.1}$$

$$o = D_j D_i(\circ_{l=1}^n \phi_l) = \sum_{k=1}^{n}(\prod_{l<k} D\phi_l)D^2\phi_k\Big((\prod_{l>k} D\phi_l)e_i, (\prod_{l>k} D\phi_l)e_j\Big), \tag{2.2}$$

where $D\phi$ is the Jacobian matrix of ϕ and each of the two components of the second derivative $D^2\phi_k(,)$ is a bilinear form with two vector-valued arguments.

Since Ω_k shares an edge with $\phi_k(\Omega_{k-1})$, we may assume that Ω_k and $\phi_k(\Omega_{k-1})$ share a coordinate direction corresponding to the common edge. We also assume that the e_2 level-lines are parallel to one another. This implies

$$D_j^m D_2^2 \phi^{[1]} = 0 \tag{A1}$$

$$D_j^m D_2 \phi^{[2]} = 0 \tag{A2}$$

for $m \ge 0$ and $j \in \{1, 2\}$ and yields the following characterization of connecting-maps at points of type P and M.

Theorem [Peters '94, Thm 4]. *If $\phi_l = \phi$, $l = 1..n$, and* A1 *and* A2 *hold, then (2.0), (2.1) and (2.2) hold if and only if for scalar constants x_1, x_2, a, $c := \cos(2\pi/n)$,*

$$\begin{aligned}\phi(u,v) := &\begin{bmatrix} 2c & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} \\ &+ \frac{1}{2}\begin{bmatrix} u & v \end{bmatrix}\begin{bmatrix} x_1 & x_3 \\ x_3 & 0 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} e_1 \\ &+ \frac{1}{2}\begin{bmatrix} u & v \end{bmatrix}\begin{bmatrix} x_2 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} e_2 \\ &+ O(u^3, u^2v, uv^2, v^3)\end{aligned}$$

If $n > 3$, then x_1, x_2, x_3 can be chosen independently and arbitrarily.
If $n = 3$, then $x_1 = x_2 = -2x_3$ must hold.

Example: C^2 surface splines as defined in Section 3 satisfy the theorem with the

following choice of constants. The function λ has been defined in the previous example.

$$\begin{aligned} c &:= \cos(2\pi/n) \qquad \kappa := -c(1-2c) \\ x_1 &:= D_2^2\phi^{[1]}(0,0) = -2\kappa \\ x_2 &:= D_2^2\phi^{[2]}(0,0) = -2\kappa \\ x_3 &:= D_1 D_2\phi^{[1]}(0,0) = \lambda'(0) = -2(1-(1+c)) = -2c \end{aligned}$$

We check that for $n = 3$, $c = -1/2$ and hence $x_3 = \kappa = 1$. Among the many choices of constants that satisfy the requirements of the theorem, the above choice is special in that it satisfies additional geometric requirements. For example, choosing $\kappa := -2c$ results in C^2 surfaces that self-intersect at mesh points when $n = 3$. The choice of constants for a construction with four-sided patches is the same since then $\lambda'(0) = 0 - (2c) = -2c$. □

The next theorem constrains connecting-maps at points of type M, that have 4 neighbors.

Proposition [Peters '94, Prop 5]. *If $n = 4$ and $J_1\phi_l = J_1\phi$ for $l = 1..4$, then (2.0-2.2) hold if and only if $c := \cos(2\pi/n)$, $y_1 = y_3, y_2 = y_4, x_{i,j} = -x_{i,j+2}$, for $i,j \in \{1,2\}$. and*

$$\begin{aligned} \phi_k(u,v) := & \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\ & + \frac{1}{2}\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} x_{1,k} & y_k \\ y_k & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} e_1 \\ & + \frac{1}{2}\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} x_{2,k} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} e_2 \\ & + h.o.t. \end{aligned}$$

The construction of C^2 surface splines using three-sided patches splits the cells surrounding the vertices of type M into two and connects the resulting patches parametrically C^2 across these splitting edges. Such points are therefore surrounded by 8 neighbors. The following corollary extends the previous theorem to this setup.

Corollary. *If $n = 8$ and every second connecting map is the identity, then the proposition applies to the remaining connecting maps.*

Example. The C^2 construction of Section 3 chooses $y_k = x_{1,k} = x_{2,k} := 0$. □

2.4. Connecting-maps and degree bounds.

The key to surface splines of low degree is to find connecting-maps at the mesh points such that, at each edge, a low degree Hermite interpolant ϕ exists to the connecting-maps at the end points. We make this precise in the context of curvature continuous splines. Recall that midpoint refinement creates two types of vertices: vertices of type P correspond to original mesh points and centroids, respectively and vertices of type M are edge-midpoints. We may choose $y = x_i = y_k = x_{i,k} = 0$ at M if each neighbor P has itself $n = 4$ neighbors. Clearly the lowest degree Hermite interpolant to the connecting-map expansions is then the identity, i.e. $\phi = \mathrm{id}$ and the mesh is of the regular, checker board type that allows a tensor-product or box spline construction.

For general n, the theorems above rule out even a linear ϕ because $\lambda(v) := (D_2\phi^{[1]})(0,v)$ must be at least quadratic. The C^2 surface splines of Section 3 use the least degree Hermite interpolant by setting

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

□

Consider now the two patches $p(u,v)$ and $q(u,v)$ with a common boundary curve $\gamma(u)$ such that the connecting-maps at $\gamma(0)$ satisfy the assumptions of [Peters '94, Prop. 5] and of [Peters '94, Thm. 4] at $\gamma(1)$. If γ is of degree d then the left side of the symmetric G^1 constraints,

$$\lambda D_1\gamma = D_2p + D_2q,$$

is of degree $d-1+2$ 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 λD_1D_2p and λD_1D_2q in the symmetric G^2 constraints,

$$D_2^2p - D_2^2q - \lambda D_1D_2p + \lambda D_1D_2q = \frac{1}{2}(D_2q - D_2p)D_2^2\phi^{[2]},$$

are therefore of degree $d+1-1+2$. Unless we have cancellation, D_2^2p and D_2^2q must hence be of degree $d+2$ and, if none of the intermediate polynomials are degree-raised, p and q must be of degree $d+4$.

Example. In Section 3, boundary curves γ 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 computed from a polynomial of degree 6. □

2.5. The nullspace of vertex-enclosure difference equations.

Once admissible connecting maps at a vertex are fixed, 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. Exactly six coefficients, corresponding to the dimension of a C^2 piecewise quadratic, are to be chosen in the 2-disk of coefficients labelled $P_{klm,i}$, $k+l+m=d$, $k=d-2..d$ (cf. Appendix 1). It is possible to simply prescribe the six coefficients P_{klm,i_0} of the i_0th patch. However, 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. A better point of view, is to continue to view the six degrees of freedom as representing the null space of the G^2 constraints. Of course, unless the number of neighbors of P is $n=3$ or $n=6$, the number of neighbors is not a multiple of the number of data required to match the dimension of the null space. When it is, 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.

Example. Consider the the null space of the constraint system for second order continuity at a vertex of type P with n neighbors that is generated by the choice of connecting-maps for C^2 surface splines as detailed in the previous two subsections. The goal is to obtain control points, $P_{220,i}$ and $P_{310,i}$ and P_{400} that together reflect the six degrees of freedom. We may set $P_{400} = 0$ since any other choice corresponds to a translation. On eliminating the constraints across the splitting edges, we obtain the following $3n$ equations in the $3n$ variables $P_{310,i}$ $P_{611,i,1}$ and $P_{220,i}$.

$$0 = P_{310,i-1} - 2cP_{310,i} + P_{310,i+1} \tag{3.5.1}$$

$$\begin{aligned} 3(1+2c)P_{220,i-1} - 6cP_{220,i} - 3P_{220,i+1} + 56c(P_{611,i,1} - P_{611,i-1,1}) \\ = 2(3-15c-\kappa)P_{310,i-1} + 16cP_{310,i} + 2(-3+7c+\kappa)P_{310,i+1} \end{aligned} \tag{3.5.2}$$

$$\begin{aligned} 6(P_{220,i-1} + 6(1+2c)P_{220,i} - 56(P_{611,i,1} + P_{611,i-1,1}) \\ = 20P_{310,i-1} + 4(11-3c)P_{310,i} + 4P_{310,i+1} \end{aligned} \tag{3.5.3}$$

The first equation forces the tangent coefficients $P_{310,i}$ into a common plane, while 3.5.2 and 3.5.3 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. For $n \neq 4$, the difference equations have the symmetric solution

$$\begin{aligned} P_{220,i} &:= B_{220} + p_{220,i} + e_{220,i} \\ P_{310,i} &:= P_{040} + a_P p_{220,i} \end{aligned}$$

where $a_P := 3/(6-2c)$, $B_{220} := \sum_j C_j/n$, $\theta := 2\pi/n$, $p_{220,i} := 2\sum_j \cos(\theta(j+i))C_j/n$, $e_{220,i} := 2\sum_j \cos(\theta(j+2i))C_j/n$ and the C_i, $i = 1..n$ are arbitray intermediate control points derived from the data at the neighboring vertices. For $n = 4$, $c = 0$ and the constraints simplify to the familiar equations of the tensor-product construction. □

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 splines, a free-form 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 known, it suffices to express the free-form 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 auxiliary control points derived from the mesh points by mesh refinement. The algorithm is broken into the six steps.

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 facets whose boundary representation has inner loops should be broken up before using them as cells, since the surface is generated by averaging and hence will generally smoothly cover the intended holes. 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. The blend ratios of each cell may be modified independently of each other and of those in other cells. Other parameters, h_{P} (respectively h_{M} or h_{S}) may be moved from their default value to shift the corresponding point in the direction of the normal at that point. Finally a_{P} (respectively a_{M} or a_{S}) determine the width of the tangent plane at the corresponding 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 patches alone or a combination of bicubic, four-sided patches covering regular mesh regions and octic, three-sided patches for the remaining regions are easily devised and will appear elsewhere.

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 by mimicking the first step of any line average or subdivsion algorithm for regular meshes. The relative position of the intermediate control points C_i reflect the choice of input blend ratios.

The first stage of the refinement splits each original n-sided cell into n four-sided *subcells*.

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.

The second stage splits each four-sided cell to obtain three-sided cells arranged in the spirit of the four-direction mesh.

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

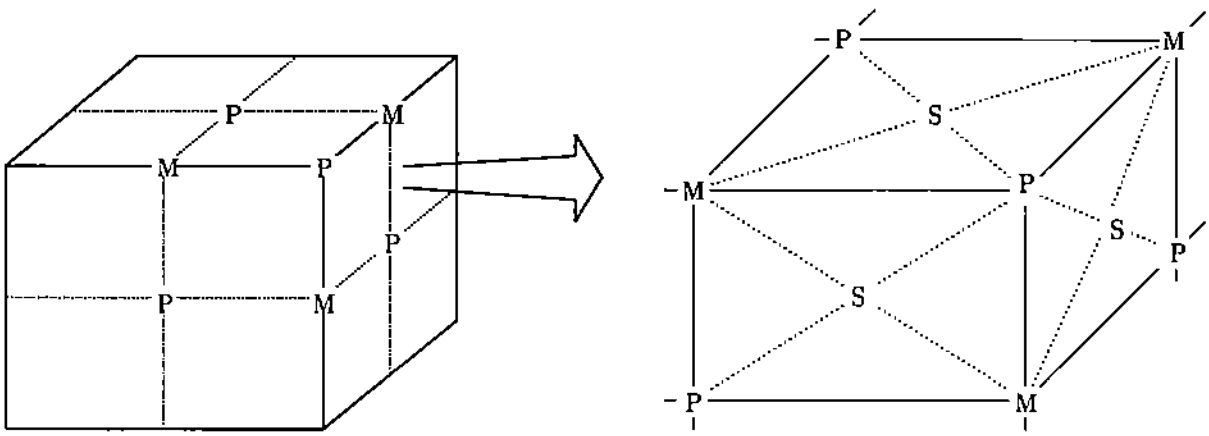
For later reference, the vertices of the refined mesh are grouped into three categories.

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

M — label for edge midpoints created in step 1.

S — label for subcell centroids created in step 4.

After Step 3 of the refinement every type P vertex is surrounded by type M vertices of degree four and all subcells are quadrilateral.



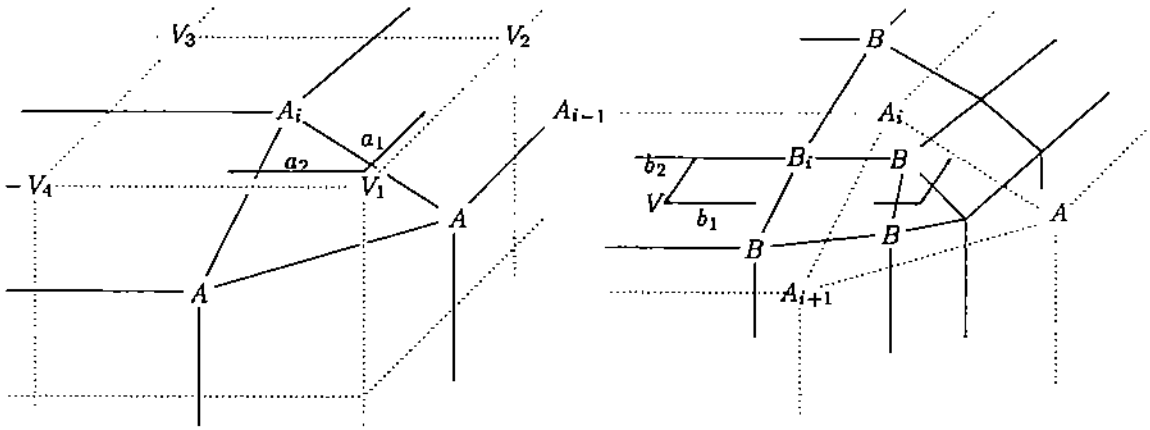
To generate the intermediate control points that transmit the choice of the blend ratios to the parametrization, we refine the mesh resulting from Step 1–3.

a. For each subcell, denote the position of the vertices in order V_1, V_2, V_3 and V_4, starting with the position of the input mesh point. Compute the level 1 intermediate control point

$$A_i := (1-a_{i1})(1-a_{i2})V_1 + (1-a_{i1})a_{i2}V_2 + a_{i1}(1-a_{i2})V_4 + a_{i1}a_{i2}V_3.$$

b. For each vertex of type P or M denote the intermediate control points of the surrounding subcells as $A_1, \ldots, A_n$ and compute the *default position*

$$V := \frac{1}{n}\sum_{j=1}^{n} A_j.$$



c. For each vertex of type P or M, let V be the position computed in Step b and A_1, ..., A_n the surrounding intermediate control points. For each pair V, A_i, compute the level 2 intermediate control point

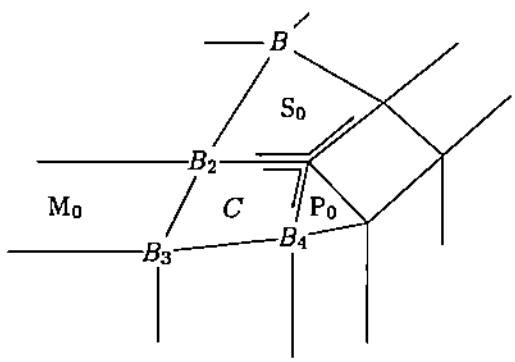
$$B_i := (1-b_{i1})(1-b_{i2})V + (1-b_{i1})b_{i2}\frac{A_{i-1}+A_i}{2} + b_{i1}(1-b_{i2})\frac{A_{i+1}+A_i}{2} + b_{i1}b_{i2}A_i,$$

where for $j \in \{0,1\}$,

$$b_{ij} := \begin{cases} a_{ij} & \text{if the vertex is an input mesh point,} \\ 1-a_{ij} & \text{if the vertex is a cell centroid,} \\ 1-a_{ij} & \text{if the vertex is type M and} \\ & \text{the } i+j\text{th neighbor vertex is a cell centroid,} \\ a_{ij} & \text{if the vertex is type M and} \\ & \text{the } i+j\text{th neighbor vertex is an input mesh point} \end{cases}$$

Remark: The ratios b_{ij} may be used as an independent set of blend ratios as long as $b_{ij} \leq a_{kj}$ for neighboring blend ratios a_{kj}.

d. For each subsubcell defined by level 2 intermediate control points B_j, $j = 1..n$, compute the centroid $\frac{1}{n}\sum B_j$. Each subsubcell is associated either with a vertex of type P, or of type M or of type S with an edge of the once refined mesh. Correspondingly the centroid is labelled P_0, M_0, S_0 or C_i. The latter is called subcell edge-coefficient. The default for a_{ij} is $a_{ij} := 0.5$.



At the end of Step 3.1 we have a refined mesh of intermediate control points C, P_0, M_0 and 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.

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

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

The coefficients of the quartic boundary curve are

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

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, respectively 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 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.

Analogous to C^1 surface spline the construction of the C^2 corridor connecting the curvature caps is by default 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 jth of the two abutting patches and a vector $K_{4-i,2+i,2}$, $i = 0, 1, 2$, that can be chosen freely without destroying the continuity. 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 connecting P with S, respectively M with S. It is possible to achieve the C^2 join with quartic curves, but such a construction probably does not satisfy the convex hull property for all blend

ratios. The curve P–S defined below is therefore of degree five.

$$P_{500,1} := P_{800,1}$$
$$P_{401,1} := \frac{8}{5}P_{701,1} - \frac{3}{5}P_{800,1}$$
$$P_{302,1} := \frac{28}{10}P_{701} - \frac{24}{10}P_{701} + \frac{6}{10}P_{800,1}$$
$$P_{104,1} := P_{005} + \frac{1}{4}(P_{203,1} - P_{203,3})$$
$$P_{005,1} := h_S S_0 + (1 - h_S)\frac{1}{4}(P_{302,1} + P_{032,2} + P_{302,3} + P_{032,4}), h_S \in [0..\frac{1}{2}]$$

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 coefficents 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 degree-raising 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 coeffcients 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

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

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.

Local interpolation of input mesh points. In Step 3.2, for each vertex P to be interpolated as follows. Move P_0 and $P_{220,i}$ along the axis of the default P_0 and B_{220} such that $P \simeq h_P P_0 + (1 - h_P) B_{220}$.

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 can be left to the computer. The referees are 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 shows that the two univariate polynomials

$$\big(D_2^m(q - p \circ \phi)\big)(u, 0), \quad m = 1..2,$$

vanish identically along the edge MP; the polynomial vanishes for $m = 0$ because p and q share the boundary curve MP. Hence p and q 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. □

It is now easy to show the vector space property.

THEOREM 4.2. *C^2 surfaces generated from input meshes with the same connectivity, choice of three-sided and four-sided patches, and blend ratio for each subcell 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 constraints on some of the constants are more conservative than they have to be in generic use. For example, for extremely unsymmetric data (cf. [Peters '9xa, Sec.4]), $\omega_n \leq 1/2$ must hold, while for symmetric configurations $\omega_n = 1$ yields a construction guaranteed to obey the convex hull property.

Theorem 5.1. *For a suitable choice of h_P, h_M, $\omega_n < 1/2$, $K_{4-i,2+i,2}$, $i = 0, 1, 2$, and $e_{220,i}$, the coefficients of the BB-form can be written as a convex combination of the points of the input mesh.*

Proof. The proof follows the flow 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 step enforces in order

$$\begin{array}{cccc} 1,2,4 & \{\mathrm{P},\mathrm{M},\mathrm{S}\} & \subset & H(\mathrm{P}), \\ a & A & \subset & H(\mathrm{P},\mathrm{M}), \\ b & \mathrm{P},\mathrm{M} & \subset & H(A), \\ c & B & \subset & H(A,\mathrm{P},\mathrm{M}), \\ d & C & \subset & H(B). \end{array}$$

2. If $\omega_n \le \frac{1}{2}$ then $B_{220} + p_{220,i} \subset H(C)$ and hence for a sufficiently small $e_{220,i}$ and $h_P \in [0..a_P]$, the construction of the quartic boundary curve results in

$$\begin{array}{rcll} P_{400} & := & (1-h_P)P_0 + h_P B_{220} & \subset H(P_0, C) \\ P_{220,i} & := & p_{220,i} + e_{220,i} & \subset H(C) \\ P_{310,i} & := & P_{400} + a_P(P_{220,i} - P_0) & \subset H(P_{220,i}, P_0), \\ P_{130,i} & := & P_{040} + \frac{1}{4}(P_{220,i} - P_{220,i+2}) & \subset H(P_{040}, C) \\ P_{040} & := & h_M M_0 + (1-h_M)\sum_{i=1}^{4} P_{220,i}/4 & \subset H(M_0, C) \end{array}$$

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 $B_{220} := \sum P_{220,i}/n$, $p_{220,i} := P_{220,i} - e_{220,i}$, $P_0 := (P_{800} - h_P B_{220})/(1-h_P)$, we find that the dominant term in c of each coefficient of $P_{611,i}$, $P_{521,i}$ and $P_{512,i}$ is positive and hence

$$\begin{aligned} P_{611,i} &\subset H(P_0, B_{220}, p_{220,i}, p_{220,i+1}) + O(e_{220,i-1}, e_{220,i}, e_{220,i+1}) \\ P_{521,i} &\subset H(P_0, B_{220}, p_{220,i}, p_{220,i+1}, P_{130,i}) + O(e_{220,i}) \\ P_{512,i} &\subset H(P_0, B_{220}, B_{130}, p_{220,i}, p_{220,i+1}, P_{130,i}) + O(e_{220,i}, e_{220,i+1}). \end{aligned}$$

$P_{8-i,i,0}$, $i = 0..3$ 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 $h_M < 2/5$

$$P_{161,i} \subset H(M_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3})$$

and for $0 < h_M < \frac{1}{432}(96 - 2(c_{i+1})^2 + 8c_i + c_{i+1} - c_{i+3} + 2(c_{i+3})^2)$

$$\begin{aligned} P_{152,i} &\subset H(M_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3}) \\ P_{251,i} &\subset H(M_0, P_{220,0}, P_{220,1}, P_{220,2}, P_{220,3}) \end{aligned}$$

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 the convex hull of $P_{220,i}$. 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 is in the order of the algorithm. Zero cut ratios coalesce all cell centers C_i surrounding an original mesh point P into P. That is,

$$P = C_i = P_{220,i} = P_{310,i} = P_{400} = P_0.$$

Also, labeling the centroids with even indices,

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

while $P_{130} = (P + M)/2$. Table 1 shows that $P_{klm} = P$ for $k \ge 6$, 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 M_i and M_{i+1}. Table 2 shows that P_{klm} for $l \ge 5$ lie on the edge P, M, except for $P_{053,i}$ which lies in the plane spanned by M and successive neighbors P_i and 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 long term goal of this line of research is to develop a general framework for free-form surface splines, of which the tensor-product B-spline is particular instance. 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 the C^1 case intuition and hand calculation in terms of the Bernstein-Bézier form can still lead to success, the definition of higher-order surface splines must free itself from the Bernstein-Bézier representation. For, while one may still choose to make the definition concrete by expressing the surface in terms of Bernstein-Bézier coefficients, the structure would be buried under the necessarily complex and long formulas. As illustrated with this paper, the translation process may ultimately be left to generic symbolic manipulation routines.

The splines over irregular meshes defined in this paper are a first step towards the solution of the general problem. They 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. A number of questions and problems remain however. Is there a setting of the default values such that the surface becomes the quartic C^2 box spline over regular meshes? Is there a setting that simplifies the proof of the convex hull property? How should one optimize over the free parameters to obtain a desirable curvature distribution?

The surface splines smoothen a general, regular or irregular mesh of points into a C^2 surface parametrized according to the user's choice by quartic and octic three-sided patches, or bicubic, biquartic and octic patches. Input meshes with the same connectivity and the same blend ratio for corresponding cells give rise to a vector space of free-form surface splines. This and the convex hull property are useful for approximating and locally editing the spline surface. The role of the knot spacing is played by geometrically intuitive blend ratios. Zero blend ratios result in a C^0 surface that tightly interpolates the input mesh. It is possible to interpolate the input mesh without solving a global sparse system of equations analoguous to interpolation by a quadratic spline at every second knot.

Due to the built-in smoothness, the representation reduces the number of unknowns for such problems as 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. Finally, free-form surface splines can be used directly to smooth and blend the boundary representation of a solid model.

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.

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, *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, *C^1 free-form surface splines*, to appear in SIAM J. of Numerical Analysis (accepted 1993).

J. Peters, *Smoothing vertex-degree bounded polyhedra*, submitted to ACM TOG

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.

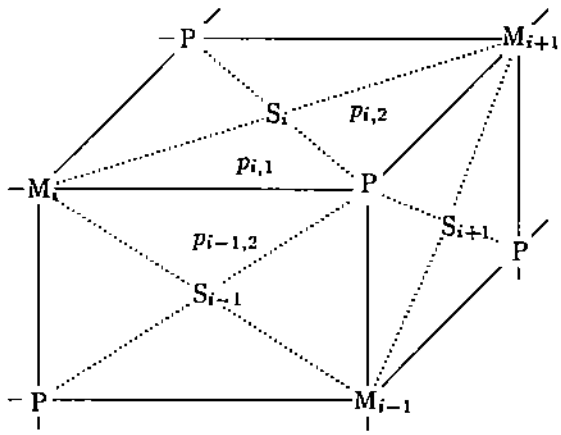
M. Sabin, *The use of piecewise forms for the numerical representation of shape*, PhD thesis, Hungarian Academy of Sciences, Budapest, Hungary, 1976.

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_i 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.



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
;
P610[j]          := 1 over    16
  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[]           -8*c^3+12*32*c^2-4*c
  P310[j]          -36*c^2+56*c+8*c^3
  P310[j+1]        -12+48*c-8*c^2
  P220[j-1]        -3
  P220[j]          12*c^2+6*c
  P220[j+1]        3+6*c
  P130[j]          0
  P130[j+1]        0
  P3bs[]           0
;
P611[j,1]        := 1 over    112*c
  P400[]           -8*c^3+12*32*c^2-4*c
  P310[j]          72*c-16*c^2+8*c^3
  P310[j+1]        -12-8*c^2+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^3-12-48*c^2
  P310[j]          48*c+28*c^2-8*c^3
  P310[j+1]        12+8*c^2-32*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[]           -5*c^2+45*c-18
  P310[j]          5*c^2-52*c+84
  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[]           5*c^2-49*c+66
  P310[j]          -5*c^2+32*c+84
  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
;
P3bs[j]          := 1 over    336*c
  P400[]           -16*c^4+105*c^3-36+228*c-175*c^2
  P310[j]          72+16*c^4-342*c+175*c^2-105*c^3
  P310[j+1]        0
  P220[j-1]        0
  P220[j]          -6*c^2+108*c-36
  P220[j+1]        0
  P130[j]          6*c^2+6*c
  P130[j+1]        0
  P3bs[]           0
;
P512[j,1]        := 1 over    336*c
  P400[]           -105*c^3+16*c^4-180*c+171*c^2+36
  P310[j]          55*c^3-8*c^4-121*c^2+276*c-36
  P310[j+1]        -8*c^4-36+234*c+50*c^3-74*c^2
  P220[j-1]        0
  P220[j]          18+27*c+12*c^2
  P220[j+1]        -27*c+6*c^3+18
  P130[j]          6*c^2+6*c
  P130[j+1]        0
  P3bs[]           336*c
;
P512[j-1,3]      := 1 over    336*c
  P400[]           -445*c^3+16*c^5-105*c^4+143*c^3-36+360*c
  P310[j]          347*c^2-93*c^3+204*c+92*c^4-16*c^5-36
  P310[j-1]        36+50*c^3+74*c^2-234*c+8*c^4
  P220[j-1]        -27*c+6*c^2+18
  P220[j]          18+27*c+12*c^2
  P220[j-1]        0
  P130[j]          6*c^2+6*c
  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^2-16*c^4+105*c^3+510*c
  P310[j+1]        -72-195*c^2-16*c^4+105*c^3+510*c
  P220[j-1]        0
  P220[j]          18*c^2+36
  P220[j+1]        18*c^2+36
  P130[j]          6*c^2+6*c
  P130[j+1]        6*c^2+6*c
  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]          1
  P220[j+1]        0
  P220[j+2]       -1
  P220[j+3]        0
  P040[]           8
  P310[j]          0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
;
P071[j]          := 1 over   16
  P220[j]          1
  P220[j+1]        1
  P220[j+2]       -1
  P220[j+3]       -1
  P040[]          16
  P310[j]          0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
;
P260[j]          := 1 over   14
  P220[j]          5
  P220[j+1]        0
  P220[j+2]       -2
  P220[j+3]        0
  P040[]          11
  P310[j]          0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
;
P062[j]          := 1 over  112
  P220[j]         17
  P220[j+1]       17
  P220[j+2]      -11
  P220[j+3]      -11
  P040[]         100
  P310[j]          0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
;
P161[j-1,2]      := 1 over  112
  P220[j]         27
  P220[j+1]       -7
  P220[j+2]      -15
  P220[j+3]        7
  P040[]         100
  P310[j]          0
  P310[j+1]        0
  P310[j+2]        0
  P310[j+3]        0
;
P161[j,1]        := 1 over  112
  P220[j]         27
  P220[j+1]        7
  P220[j+2]      -15
  P220[j+3]       -7
  P040[]         100
  P310[j]          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]          6*c[3]^2-3600-6*c[1]^2-3*c[3]+1*c[1]-24*c[0]
  P220[j+1]       -4*c[1]-2*c[2]^2+34*c[0]^2-17*c[0]-552*c[2]-4*c[3]
  P220[j+2]       24*c[0]+6*c[1]^2-1008-6*c[3]^2+3*c[3]-3*c[1]
  P220[j+3]       552-34*c[0]^2+17*c[0]-c[2]+4*c[1]+2*c[2]^2+4*c[3]
  P040[]         5184
  P310[j]         288
  P310[j+1]         0
  P310[j+2]         0
  P310[j+3]         0
;
P251[j,1]        := 1 over  8064
  P220[j]          6*c[1]^2-6*c[3]^2+3*c[3]+3600-24*c[0]-3*c[1]
  P220[j+1]       552-34*c[0]^2+17*c[0]-c[2]+4*c[1]+2*c[2]^2+4*c[3]
  P220[j+2]       -1008+6*c[3]^2-3*c[3]+24*c[0]+3*c[1]-6*c[1]^2
  P220[j+3]       -4*c[1]-2*c[2]^2+34*c[0]^2-17*c[0]-552+c[2]-4*c[3]
  P040[]         5184
  P310[j]         288
  P310[j+1]         0
  P310[j+2]         0
  P310[j+3]         0
;
P152[j-1,2]      := 1 over  8064
  P220[j]         10*c[3]^2-c[2]+c[1]-2*c[1]^2-7*c[0]+2706-5*c[3]
  P220[j+1]       -2*c[2]^2-5*c[0]-930-c[1]+10*c[0]^2-7*c[3]+c[2]
  P220[j+2]       -10*c[3]^2+2*c[1]^2+7*c[0]+c[2]+5*c[3]-1158-c[1]
  P220[j+3]       7*c[3]-c[2]-10*c[0]^2+2*c[2]^2+c[1]+1182+5*c[0]
  P040[]         6120
  P310[j]         144
  P310[j+1]         0
  P310[j+2]         0
  P310[j+3]         0
;
P152[j,1]        := 1 over  8064
  P220[j]         -2*c[3]^2-c[2]-5*c[1]+10*c[1]^2-7*c[0]+2706+c[3]
  P220[j+1]       2*c[2]^2+5*c[0]+1182+7*c[1]-10*c[0]^2+c[3]-c[2]
  P220[j+2]       2*c[3]^2-10*c[1]^2+7*c[0]+c[2]-c[3]-1158+5*c[1]
  P220[j+3]       -c[3]+c[2]+10*c[0]^2-2*c[2]^2-7*c[1]-930-5*c[0]
  P040[]         6120
  P310[j]         144
  P310[j+1]         0
  P310[j+2]         0
  P310[j+3]         0
;
P053[j]          := 1 over   224
  P220[j]         54
  P220[j+1]       54
  P220[j+2]      -29
  P220[j+3]      -29
  P040[]         170
  P310[j]          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
;
PJ50[]          := 1 over    14
     P400[]       0
     P310[]       1
     P220[]       6
     P130[]       6
     P040[]       1
;
PJ41[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            X422
;
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            X332
;
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+84*c^2+804
     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            X242
;
```

Table 3 Coefficients of the curvature corridor PM.

```
P132[j]          := 1 over    8640
     P800[j]       1086*c^2-3179*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]       1680+416*c^2-976*c
     P080[j]       -1728-765*c-210*c^2
     P701[j]       1792+1920*c-980*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*K333
0
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+2840*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]       -3808*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*K333
0
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*K333
0
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
0
```

```
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
0
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]       -1
     P701[j]       0
     P611[j]       0
     P521[j]       0
     P251[j]       42
     P161[j]       -28
     P071[j]       6
0
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
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
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
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:   bbimult
# runs under:   Maple V Release 2
# author:       J"org Peters, jorg@cs.purdue.edu
# date:         Jan 1994
#
# multiply two univariate polynomials in BB-form
#
#-------------------------------------------------------------------

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

# b1: array(0..d1)
# d1 : degree of b1
#-------------------------------------------------------------------

local i1,i2,i12, dgout,out:
if nargs <> 4 then
  ERROR('wrong number of arguments'): fi;

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

for i1 from 0 to dgout do
    out[i1] := 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:
#-------------------------------------------------------------------
```

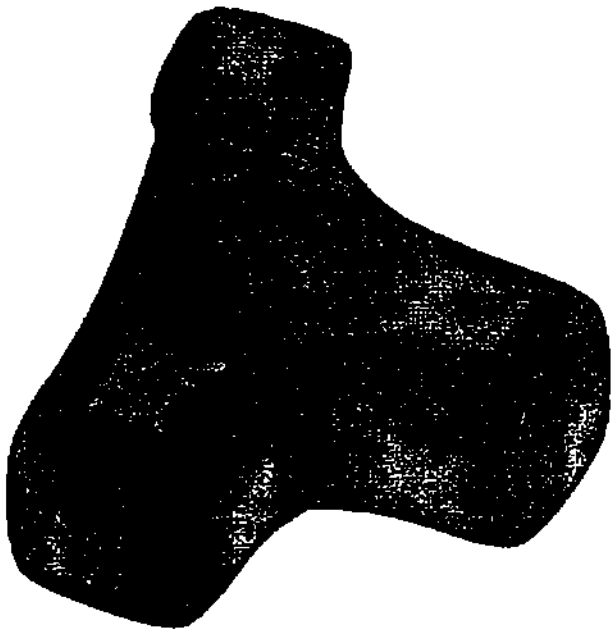
```
#
# MAPLE file:   chk
# runs under:   Maple V Release 2
# author:       J"org Peters, jorg@cs.purdue.edu
# date:         Jan 1994
#-------------------------------------------------------------------
#
# check C1 and C2 conditions between the patches
#  p  and  q  and  across the splitting edges in the
# neighborhood of M and P
# (between p and r, respectively p and s)
#
#
#      s  /             \  r
#        /       p       \
#       M-----------------P
#        \       q       /
#         \             /
#
    d := 5;     # 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
#       base1,                    --- nullspace of 3rd layer at P
#       K343,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 bbimult:
```

```
# Dip   contains the coefficients of (1/d)D_i p
    Dip := array(0..d-1):
    D2p := array(0..d-1):
    D2q := array(0..d-1):
    for i from 0 to d-1 do
        Dip[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
    Di2  := array(0..d-2):
    D22  := array(0..d-2):
    for i from 0 to d-2 do
        Di2p := 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];
        Di2q := 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);
        Di2[i] := simplify(Di2p-Di2q);
    od:

# derivatives of the connecting map f and the degree raising polynomial
    Df1 := array(0..2):
    Df1[0] := 1: Df1[1] := 1: Df1[2] := 1-c:
    Df2f1 := array(0..1):
    Df2f1[0] := 0: Df2f1[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= Df1*Dip across the edge PM
    'error in C1 conditions across the edge P M';
    u := bbimult(Dip,d-1,Df1,2):
    v := bbimult(D2p,d-1,rais,2):
    w := bbimult(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-Df1*Di2p+Df2f1*D2p = D22q-Df1*Di2q+Df2f1*D2q across the edge PM
    'error in C2 conditions across the edge P M';
    uv := bbimult(Di2,d-2,Df1,2):
    vv := bbimult(D22,d-2,rais,2):
    ww := bbimult(D2,d-1,Df2f1,1):
    c2err := array(0..d):
    for j from 0 to d do
        c2err[j] := simplify(vv[j]-uv[j]+ww[j]/(d-1)):
    od:
    print(c2err);
```

Table 5 Maple routines for checking curvature continuity.

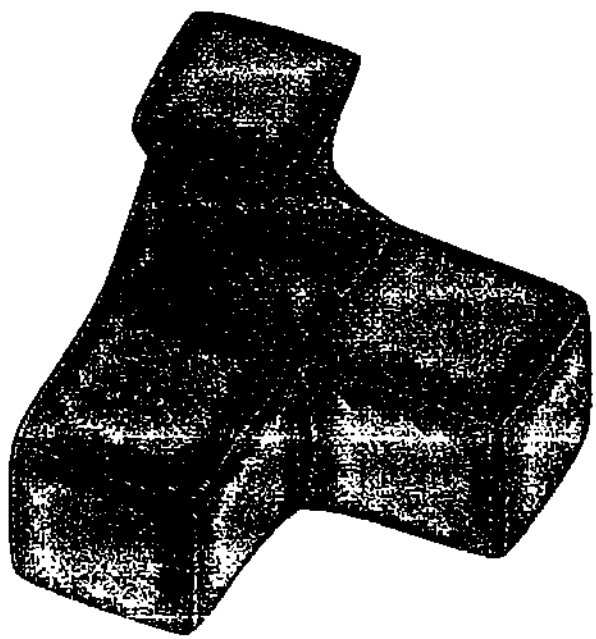
Appendix 4. Maximal absolute curvature and blend ratios. The following figures show maximal absolute curvature measured at the vertices of the three-sided patches after 3-fold subdivision and displayed with the same color scale.

Fig. A4.1



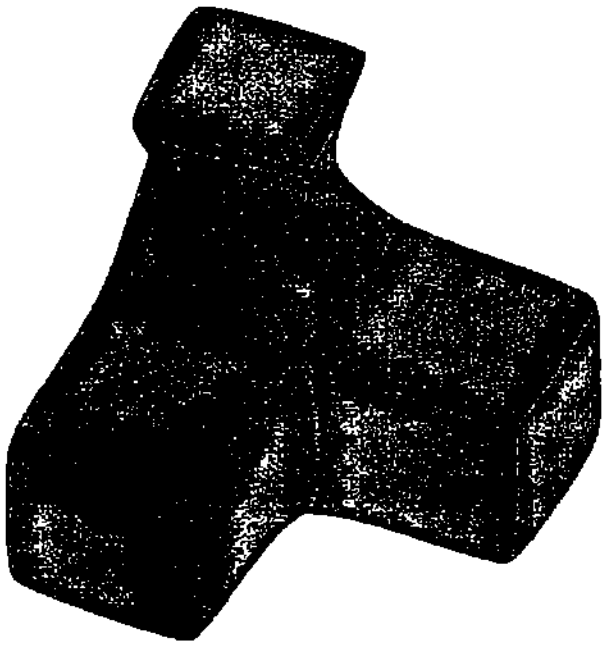
$\alpha_{ij} = 0.4$ globally

Fig. A4.2



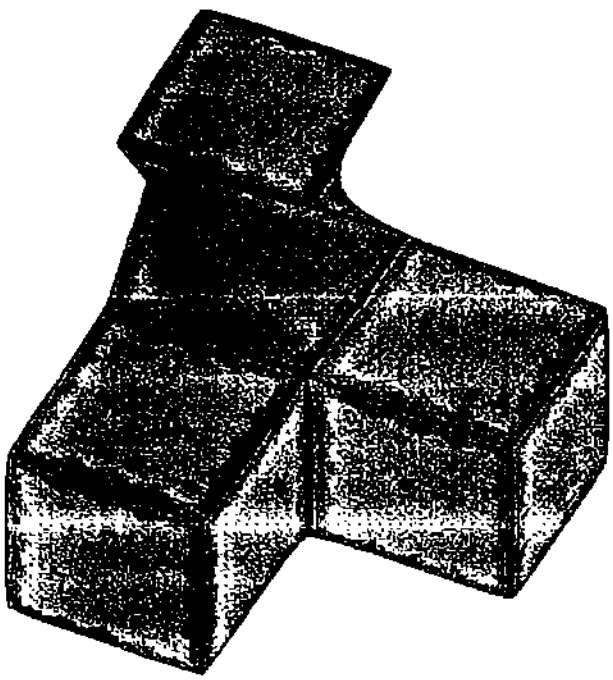
$\alpha_{ij} = 0.25$ globally

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