Polyhedral modeling

Georges-Pierre Bonneau, Stefanie Hahmann

Polyhedral modeling

Georges-Pierre Bonneau * Stefanie Hahmann

Laboratoire LMC - CNRS
University of Grenoble, France

Abstract

Polyhedral meshes are used for visualization, computer graphics or geometric modeling purposes and result from many applications like iso-surface extraction, surface reconstruction or CAD/CAM. The present paper introduces a method for constructing smooth surfaces from a triangulated polyhedral mesh of arbitrary topology. It presents a new algorithm which generalizes and improves the triangle 4-split method [7] in the crucial point of boundary curve network construction. This network is then filled-in by a visual smooth surface from which an explicit closed form parametrization is given. Furthermore, the method becomes now completely local and can interpolate normal vector input at the mesh vertices.

Keywords and phrases: triangular meshes, visual continuity, arbitrary topology, visualization.

1 INTRODUCTION

Polyhedral meshes consisting of a collection of vertices, edges and triangular faces which describe an oriented 2-manifold in $\mathbb{R}^3$ are dealt with in many applications. They result f.ex. from iso-surface extraction algorithms like marching cubes. Surface reconstruction methods used for virtual reality purposes of CAD/CAM applications produce a fine triangulated polyhedral mesh approximating an object from which a 3D scanner has sampled millions of unorganized data points. In medical imaging where CT or MRI scanner produce a set of slices with a grid of gray level values a 3D volume is reconstructed. But contour line approximation on each slide followed by contour reconstruction also produces a polyhedral mesh. They also occur in CAD/CAM applications where the classical tensor product surface models fail. Topological complex situations like inner corner blends need an extraneous treatment. Surfaces defined on polyhedral meshes can however represent surfaces of arbitrary topological type.

In particular in CAD/CAM applications or surface reconstruction but also for computer graphics or visualization purposes, it is often desired to get a visual smooth representation of the object which is roughly approximated by a polyhedral mesh. On one hand surface subdivision algorithms converge to a smooth surface starting from a polyhedral mesh, but they generally don't provide an explicit closed form expression of the resulting surface. Therefore, computations on such surfaces like intersections or evaluations of intrinsic quantities like curvature can become tricky. On the other hand, and this is the subject the present paper deals with, parametric polynomial surfaces fulfill all these requirements. The polyhedral mesh serves as a control mesh. Recall that, since every polyhedra can be triangulated, it is sufficient to consider as input a triangular mesh. Triangular surface patches which join with geometric continuity in order to produce an overall smooth surface while interpolating the mesh vertices can then be computed.

The main difficulty in developing a triangular interpolation scheme for polyhedral meshes of arbitrary topological type consists of ensuring the G^1 continuity (visual smoothness, tangent plane continuity). Three kinds of schemes exist for polynomial representations: the Clough-Tocher-like domain splitting methods [3], [12] the G^2 continuous constructions [11]; or the twist compatible boundary curve schemes [8], [7]. Further non-polynomial methods also exist [5], [6], [10], [1]. In addition to an explicit closed form parametrization, the scheme should be local. This means that a local modification of the polyhedral control mesh should only locally affect the surface. This allows an interactive modeling of smooth surfaces by interactive modeling of polyhedral meshes.

The present paper deals with such a polyhedral modeling method. It is based on a recently developed method [7], called 4-split method (chapter 2), and presents a new algorithm (chapter 3) which generalizes and improves the 4-split scheme in such a way that the number of degrees of freedom for the construction of the boundary curves can significantly be increased. A more flexible and more intuitive control of the surface shape is now possible. Further features of the scheme are localness and normal vector interpolation at the mesh vertices.

2 THE 4-SPLIT METHOD

The triangular 4-split method interpolates a polyhedral mesh of arbitrary topology by a visual smooth (G^1 continuous) piecewise quintic Bézier surface. It is local, affine invariant and provides an explicit closed form parametrization. Several degrees of freedom are available for shape control.

The following subsections briefly present first the method's basic idea and why this is benefit in comparism to other existing schemes. Then the algorithm itself is outlined without going into too much detail, but enough for reproducing this method.

Let the polyhedral control mesh $\mathcal{M}$ be a set of vertices, edges and triangular faces that describe an oriented 2-manifold in $\mathbb{R}^3$. The surface $\mathcal{S}$ which interpolates the vertices of $\mathcal{M}$ is composed of triangular surfaces M^i which are in one-to-one correspondence to the mesh facets and which meet with tangent plane (G^1) continuity along the common boundary curves. The M^i's are piecewise

*Laboratoire LMC-IMAG, BP.53, F-38041 Grenoble Cedex 9, France.
E-mail: [bonneau|hahmann]@imag.fr
URL: http://www-lmc.imag.fr/lmc-mga/Georges-Pierre.Bonneau
URL: http://www-lmc.imag.fr/lmc-mga/Stefanie.Hahmann

polynomial images of the unit domain triangle, we call them *macro-patches*.

2.1 Four split of the mesh triangles.

The basic idea of the present method is to subdivide each domain triangle into four subtriangles by joining together the edge midpoints. Each macro-patch therefore consists of four Bézier patches. A convenient parametrization of the macro-patches concentrates on the mesh vertices, see fig. 1.

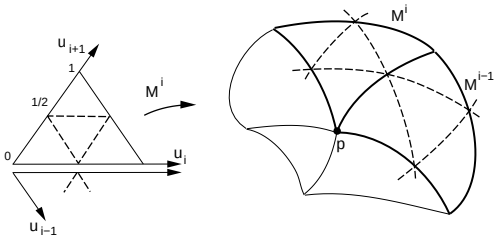

Figure 1: Parameterization of n patches around a common vertex.

This triangle 4-split has several advantages:

- it allows to introduce more degrees of freedom for each macro-patch,
- the boundary curves and cross-boundary tangents are piecewise polynomial functions (2 pieces),
- the surface scheme is local.

It will be seen later that the requirements on the boundary curves concern their first and second derivatives at the endpoints. In order to keep the surface scheme local the derivatives should be independent from one vertex to another. Instead of degree 5 curves, piecewise curves need only to be of degree 3. Similar considerations hold for the cross-boundary tangents and finally the macro-patches are composed of four quintic Bézier triangles, which is the lowest possible degree for this kind of methods.

The polyhedral 4-split mesh interpolation method consists of three main steps:

- generating first and second derivative informations at the mesh vertices,
- computing first order cross derivatives along the curves,
- computing inner Bézier points for each macro-patch.

2.2 Generating first and second derivative informations at the mesh vertices.

The first step of the algorithm consists of constructing a network of piecewise cubic C^1 boundary curves of the macro-patches interpolating the mesh vertices. It turns out that the G^1 conditions for joining a network of patches together imposes quite restrictive conditions on the set of boundary curves incident to a common mesh vertex.

Let $\boldsymbol{p}$ be a mesh vertex of order n, i.e. $\boldsymbol{p}$ is common to n mesh facets (or edges) and therefore has n neighbour points $\boldsymbol{p}_i$. The n first and second derivatives of the n boundary curves in $\boldsymbol{p}$ are not free. The boundary curve network is constructed from these derivative informations by computing for each vertex the curve pieces incident to that vertex. Since each piece of curve is a cubic Bézier parameterized over $[0, \frac{1}{2}]$, this means that the control points $\boldsymbol{b}_0$, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ must be computed. They are directly related to the first and second derivatives at the curve's endpoints by $\boldsymbol{x}(0) = \boldsymbol{b}_0 \quad \boldsymbol{x}'(0) = 6(\boldsymbol{b}_1 - \boldsymbol{b}_0) \quad \boldsymbol{x}''(0) = 24(\boldsymbol{b}_2 - 2\boldsymbol{b}_1 + \boldsymbol{b}_0)$. The complementary curve pieces are computed when the neighbour mesh vertices are treated. Finally, the curve network is closed by computing for each boundary curve the middle control point $\boldsymbol{b}_3$ which is common to both curve pieces and which should be calculated such that the two pieces join with C^1 continuity.

Let us consider a mesh vertex $\boldsymbol{p}$ and its n curve pieces as illustrated in fig. 2.

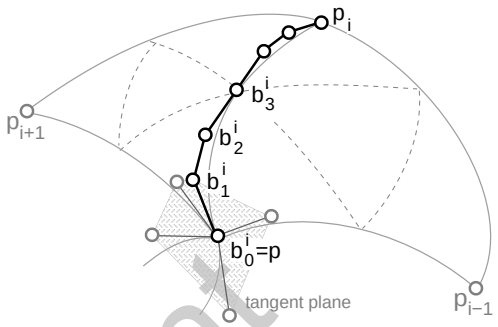

Figure 2: Boundary curves incident to a mesh vertex.

The control points $\boldsymbol{b}_0^i$ are fixed by the interpolation condition. Then the first and second derivative informations of the n boundary curves have to be calculated in order to determine $\boldsymbol{b}_1^i$ and $\boldsymbol{b}_2^i$. These derivatives have to satisfy a set of equations (see (4) and (7) of chapter 3) and are therefore not free. The following solution has been found [7], which consists of

$$\begin{aligned} \boldsymbol{b}_0^i &= \boldsymbol{p} \\ \boldsymbol{b}_1^i &= \boldsymbol{b}_0^i + \tfrac{\beta}{n} \textstyle\sum_{j=1}^{n} \cos\!\big(\tfrac{2\pi(j-i)}{n}\big)\boldsymbol{p}_j \\ \boldsymbol{b}_2^i &= [1 - \tfrac{2}{3}\gamma_2]\boldsymbol{b}_0^i + \tfrac{\gamma_1\beta}{n} \textstyle\sum_{j=1}^{n} \cos\!\big(\tfrac{2\pi(j-i)}{n}\big)\boldsymbol{p}_j \\ &\quad + \gamma_2 \textstyle\sum_{j=1}^{n} \boldsymbol{p}_j \begin{cases} 1/6 & \text{if } j = i-1, i+1 \\ 1/3 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{1}$$

where $i = 1, \ldots, n$ is taken modulo n. A few degrees of freedom $\{\beta, \gamma_1, \gamma_2\}$ are nevertheless available for each vertex. The control points $\boldsymbol{b}_1^i$ form an affine transformation of a regular n-gon. They all lie in the same plane, which is the tangent plane at $\boldsymbol{p}$. β controls the magnitude of the first derivatives. The second derivative control points $\boldsymbol{b}_2^i$ lie in a plane containing the 3 points $\boldsymbol{b}_0^i$, $\boldsymbol{b}_1^i$ and $\boldsymbol{d}_i = \frac{\gamma_2}{6}(2\boldsymbol{p} + \boldsymbol{p}_{i-1} + 2\boldsymbol{p}_i + \boldsymbol{p}_{i+1})$. More details can be found in [7].

2.3 Computing first order cross derivatives along the edges.

The second step of the algorithm is also related to the G^1 continuity conditions. It consists of computing the network of tangent planes along the boundary curves. The boundary curves determine already the tangent planes of the macro-patches at the mesh vertices. At each curve point $u_i \in [0, 1]$ the tangent plane is spanned by the curve derivative $\frac{\partial M^i}{\partial u_i}(u_i, 0)$ and one of the cross-boundary derivatives $\frac{\partial M^i}{\partial u_{i+1}}(u_i, 0)$.

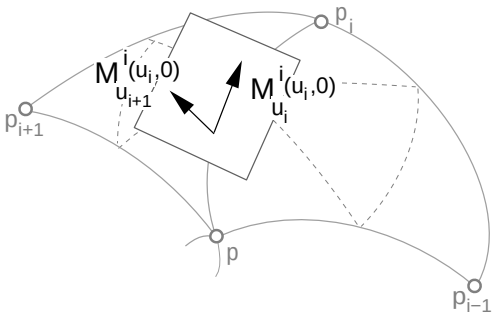

Figure 3: Cross-boundary tangents.

The cross-boundary derivatives are defined as

$$\frac{\partial M_i}{\partial u_{i+1}}(u_i,0) = \Phi_i(u_i)\frac{\partial M_i}{\partial u_i}(u_i,0) + \Psi_i(u_i)\boldsymbol{V}_i(u_i), \quad (2)$$

where Φ_i and Ψ_i are scalar functions and $\boldsymbol{V}_i$ is a vector valued function. These functions should

- be of minimal degree, and
- interpolate the first and second derivatives at the curve endpoints.

It turns out that $\boldsymbol{V}_i$ is a piecewise continuous degree two polynomial, which can be written in Bézier form. As in the case of the boundary curves, only the pieces incident to the mesh vertex $\boldsymbol{p}$ are computed:

$$\begin{aligned}
\boldsymbol{V}_i(u_i) &= \textstyle\sum_{j=0}^{2} \boldsymbol{v}_j^i B_j^2(u_i), \quad u_i \in [0,\tfrac{1}{2}] \quad \text{with} \\
\boldsymbol{v}_0^i &= \tfrac{6\beta}{n}\textstyle\sum_{j=1}^{n} \sin\tfrac{2\pi(j-i)}{n}\boldsymbol{p}_j \\
\boldsymbol{v}_1^i &= \tfrac{6\beta}{n}\textstyle\sum_{j=1}^{n} \tfrac{1}{\psi_i^0}\left[(\phi^1 - 8\phi^0 + 4\phi^0)\tan(\tfrac{\pi}{n}) - \psi_i^1\right] \\
&\quad \cdot \sin\left(\tfrac{2\pi(j-i)}{n}\right) + \tfrac{4}{\psi_i^0}\gamma_2\phi^0 \begin{cases} 1 & \text{if } j = i+1 \\ -1 & \text{if } j = i-1 \end{cases}, \\
\boldsymbol{v}_2^i &\quad \text{is free subject to } \boldsymbol{V}_i(\tfrac{1}{2}^+) = \boldsymbol{V}_i(\tfrac{1}{2}^-)
\end{aligned} \quad (3)$$

where $\Phi^0 = \cos\frac{2\pi}{n}$, $\Phi^1 = 1 + 2\sin\frac{2\pi}{n}$, $\Psi^0 = \sin\frac{2\pi}{n}$, $\Psi_i^1 = \sin\frac{2\pi}{n_i} - \sin\frac{2\pi}{n}$, and n_i is the order of $\boldsymbol{p}_i$. Ψ_i is a linear function, therefore the cross-boundary tangents are therefore piecewise cubic. However, the surface must be of degree 5 in order to ensure C^1 continuity between the 4 inner Bézier patches. The ribbons of cross-boundary tangents are now determined, and the first inner row of control points of the macro-patches can be computed from (1) and (3).

2.4 Computing inner Bézier points.

Each macro-patch is composed of four quintic triangular Bézier patches. The boundary curves of the macro-patch are the twice degree elevated curves of section 2.2. The cross-boundary tangents of sect. 2.3 determine the first inner row of control points after one degree elevation. The remaining 15 inner control points, which are highlighted in fig. 4-left, are now computed by joining the four inner patches with C^1 continuity. The necessary and sufficient C^1-continuity conditions between two internal Bézier patches inside one macro-patch are shown in fig. 4-right: all pairs of adjacent triangles must form a parallelogram, otherwise C^1 continuity would not be possible. In [7] it was shown, that the first and last pairs of adjacent triangles in fig. 4-right already form parallelograms (gray shaded).
The 15 control points are therefore subject to 9 linear parallelogram equations ensuring C^1 joints of the 4 patches. 6 control points are thus free for shape control of the macro-patch. Minimization of an energy functional produces a well shaped smooth surface.

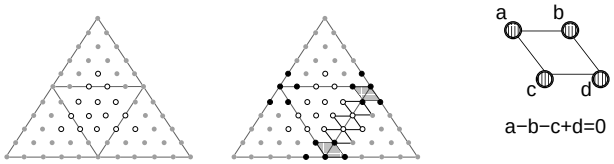

Figure 4: C^1-conditions between two adjacent quintic Bézier patches.

3 GENERATING DERIVATIVE INFORMATION AT THE MESH VERTICES

The polyhedral mesh interpolation method which we now want to describe in detail has the same skeleton as the 4-split method sketched out in chapter 2. It preserves all the attractive features but generalizes the method in an important point, that is the construction of the boundary curve network. A more flexible approach will be presented offering more degrees of freedom which are necessary in order to obtain a well shaped curve network. It is well known that visual smoothness, which stands for G^1 continuity, generally doesn't always guarantee a "nice shape" [9]. "Bad" shape features, like wiggles, bumps or self-intersections, should be avoided and the shape inherent to the polyhedral input mesh should be reproduced smoothly. It is therefore obvious that the construction of the boundary curve network is the crucial point of such an interpolation method; it is predominant for the final result.

The generalized method we propose for the construction of a G^1 compatible curve network interpolating a polyhedral mesh is due to a detailed study of the G^1 and the twist conditions which are imposed at the mesh vertices. This leads to a new and quite different approach as what has been done until now in related works [7], [8]. The benefit is threefold: more freedom and intuitive control for the boundary curve network is available; one gets a more local interpolant; normal vector interpolation becomes possible.

3.1 G^1 Continuity.

Let us start by remembering that two patches X and Y sharing a common boundary curve meet with *first order geometric continuity*, denoted G^1, if they share a common tangent plane at all points of their common boundary, i.e. if there exist three scalar functions Φ, μ, ν such that

$$\Phi(s)Y_s(s,0) = \mu(s)X_r(0,s) + \nu(s)Y_t(s,0)\,.$$

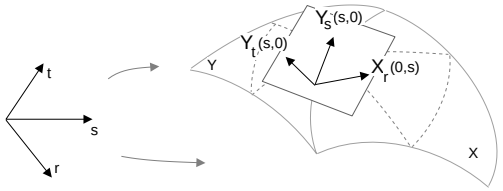

Figure 5: G^1 continuity between two adjacent patches.

When joining polynomial patches together to a network of patches with G^1 continuity special attention has to be paid to the patch corners. The idea is to construct the triangular surface patches in one-to-one correspondence to the polyhedral mesh faces. Let us consider the n patches parameterized as in fig. 1, each triangular patch

$M^i = M^i(u_i, u_{i+1})$ is the piecewise polynomial image of the unit triangle. $i = 1, \ldots, n$ is taken modulo n.

In order to obtain at the end an explicit expression for each patch some simplifying and symmetric assumptions on the G^1 conditions have to be made as follows:

$$\Phi_i(0) = \Phi_{i-1}(0) =: \Phi^0, \quad \Phi_i'(0) = \Phi_{i-1}'(0) =: \Phi^1,$$

$$\mu_i(u_i) \equiv \nu_i(u_i) \equiv \frac{1}{2} \quad i = 1, \ldots, n.$$

The G^1 condition between a pair of surface patches M^i and M^{i-1} now looks as follows:

$$\Phi_i(u_i)\frac{\partial M^i}{\partial u_i}(u_i, 0) = \frac{1}{2}\frac{\partial M^{i-1}}{\partial u_{i-1}}(0, u_i) + \frac{1}{2}\frac{\partial M^i}{\partial u_{i+1}}(u_i, 0). \quad (4)$$

At the patch corner $\boldsymbol{p}$, i.e. for $u_i = 0$, the condition (4) can be written in matrix form as follows:

$$P\boldsymbol{r}^1 = 0, \quad (5)$$

where

$$P = \begin{bmatrix} \Phi^0 & -\frac{1}{2} & 0 & \ldots & 0 & -\frac{1}{2} \\ -\frac{1}{2} & \Phi^0 & -\frac{1}{2} & & & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & & 0 \\ 0 & & & & & -\frac{1}{2} \\ -\frac{1}{2} & & & & -\frac{1}{2} & \Phi^0 \end{bmatrix}, \quad \boldsymbol{r}^1 = \begin{bmatrix} \frac{\partial M^1}{\partial u_1}(0,0) \\ \\ \vdots \\ \\ \frac{\partial M^n}{\partial u_n}(0,0) \end{bmatrix}.$$

Note, that $\boldsymbol{r}^1$ is a $(n \times 3)$ vector containing the first partial derivatives of the patch boundary curves at the vertex $u_i = 0$.
The function Φ_i can be determined from the fact that $\det P = 0$ should be satisfied and that the derivative of Φ_i at $u_i = 0$ should not depend on the value $\Phi(1)$. Φ_i should also be of lowest possible degree which leads to the following choice:

$$\Phi_i(u_i) = \begin{cases} \cos\frac{2\pi}{n}(1 - 2u_i) + u_i & \text{for } u_i \in [0, \frac{1}{2}] \\ (1 - u_i) + (1 - \cos\frac{2\pi}{n_i})(2u_i - 1) & \text{for } u_i \in [\frac{1}{2}, 1] \end{cases} \quad (6)$$

3.2 Twist Compatibility.

When joining n polynomial patches around a common vertex with G^1 continuity an additional condition has to be satisfied at the common vertex. It is called *twist compatibility condition* and given by

$$T\boldsymbol{t} = \Phi^1\boldsymbol{r}^1 + \Phi^0\boldsymbol{r}^2, \quad (7)$$

where

$$T = \begin{bmatrix} \frac{1}{2} & 0 & \cdots & & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \cdots & & & 0 \\ & & \ddots & & & \\ 0 & \cdots & & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \cdots & & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

$$\boldsymbol{t} = \begin{bmatrix} \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{t}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 M^1}{\partial u_1 \partial u_2}(0,0) \\ \\ \vdots \\ \\ \frac{\partial^2 M^n}{\partial u_n \partial u_1}(0,0) \end{bmatrix}, \quad \boldsymbol{r}^2 = \begin{bmatrix} \frac{\partial^2 M^1}{\partial u_1^2}(0,0) \\ \\ \vdots \\ \\ \frac{\partial^2 M^n}{\partial u_n^2}(0,0) \end{bmatrix}.$$

It is obtained by differentiating (C) with respect to u_i and evaluation at $u_i = 0$. The condition (7) must hold because polynomial patches lie in the continuity class C^2, implying that the *twist terms* (mixed partial derivatives) are identical.
Different approaches exist in order to solve the twist compatibility problem [12], [10], [11], [8], [7]. We solve the system (7) which is singular if n is even, by constructing a *twist compatible* boundary curve network i.e. boundary curves are determined such that the vectors $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ lie in the image space of T. It will turn out in the following subsections that the kernel of P and the image space of T is explicitly known [2]. This will enable us to characterize the vectors $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ with a maximum number of degrees of freedom while simultaneously satisfying equations (4), (5), (7). Once the first and second order derivatives of the boundary curves at the vertices, $\boldsymbol{r}^1$, $\boldsymbol{r}^2$, are known, the whole curve network is also known. The construction of the cross-boundary tangents and the inner patches can then be continued as in the 4-split method, see chapter 2 in order to obtain a smooth polyhedral mesh modeling method.

3.3 First order derivatives.

Let us now explain how the boundary curves can be determined. They will consist of piecewise (two pieces) cubic curves in order to keep the scheme local by separating first and second derivatives from one vertex to its neighbouring vertices. The final triangular patches will be given explicitly in Bézier form, let us therefore use the Bernstein-Bézier notations [4] for the boundary curves as well.

Let $\boldsymbol{x}(t) = \sum_{i=0}^{3} \boldsymbol{b}_i B_i^3(t)$, $t \in [a, b]$ be a cubic Bézier curve. Its derivatives are given by $\boldsymbol{x}^r(t) = \frac{n!}{r!}\frac{1}{(b-a)^r}\sum_{i=0}^{n-r}\Delta^r\boldsymbol{b}_i B_i^{n-r}(t)$. At the end point $t = a$ one obtains for the first and second derivatives

$$\begin{aligned} \boldsymbol{x}'(a) &= \tfrac{3}{b-a}(\boldsymbol{b}_1 - \boldsymbol{b}_0), \\ \boldsymbol{x}''(a) &= \tfrac{6}{(b-a)^2}(\boldsymbol{b}_2 - 2\boldsymbol{b}_1 + \boldsymbol{b}_0). \end{aligned}$$

For the boundary curve pieces incident to the vertex $\boldsymbol{p} = \boldsymbol{b}_0^i$ we adopt the notations as illustrated in fig. 2. The joining curve pieces are determined from the neighbouring vertices $\boldsymbol{p}_i$ (locality of the scheme). $\boldsymbol{b}_0^i = \boldsymbol{p}$ is already known (interpolation of the mesh vertices), the vector $\boldsymbol{r}^1$ is therefore determined by the control points $\boldsymbol{b}_1^i$ and $\boldsymbol{r}^2$ is determined by the control points $\boldsymbol{b}_2^i$, $i = 1, \ldots, n$. From (5) and (7) follows that the vector $\boldsymbol{r}^1$ should lie in the kernel of P and in the image space of T. P and T are cyclic matrices [DAV79]. The kernel of P is spanned by two vectors

$$\boldsymbol{k}^a = \begin{bmatrix} 1 \\ \vdots \\ \cos\left(\frac{2i\pi}{n}\right) \\ \vdots \\ \cos\left(\frac{2(n-1)\pi}{n}\right) \end{bmatrix}, \quad \boldsymbol{k}^b = \begin{bmatrix} 0 \\ \vdots \\ \sin\left(\frac{2i\pi}{n}\right) \\ \vdots \\ \sin\left(\frac{2(n-1)\pi}{n}\right) \end{bmatrix}. \quad (8)$$

$\boldsymbol{k}^a$ and $\boldsymbol{k}^b$ lie also in the image space of T, because two vectors $\bar{\boldsymbol{k}}^a$ and $\bar{\boldsymbol{k}}^b$ exist such that

$$T\bar{\boldsymbol{k}}^a = \boldsymbol{k}^a \quad \text{and } T\bar{\boldsymbol{k}}^b = \boldsymbol{k}^b,$$

where

$$\bar{\boldsymbol{k}}^a = \begin{bmatrix} \vdots \\ \cos\left(\frac{2i\pi}{n}\right) - \tan\left(\frac{\pi}{n}\right)\sin\left(\frac{2i\pi}{n}\right) \\ \vdots \end{bmatrix},$$

and

$$\bar{\boldsymbol{k}}^b = \begin{bmatrix} \vdots \\ \tan\left(\frac{\pi}{n}\right)\cos\left(\frac{2i\pi}{n}\right) + \sin\left(\frac{2i\pi}{n}\right) \\ \vdots \end{bmatrix}.$$

The vector $\boldsymbol{r}^1$ can therefore be written as a linear combination of $\boldsymbol{k}^a$ and $\boldsymbol{k}^b$

$$\begin{bmatrix} \boldsymbol{r}^1 \end{bmatrix}_{n\times 3} = \begin{bmatrix} \boldsymbol{k}^a \end{bmatrix}_{n\times 1} (\ \boldsymbol{a}\ )_{1\times 3} + \begin{bmatrix} \boldsymbol{k}^b \end{bmatrix}_{n\times 1} (\ \boldsymbol{b}\ )_{1\times 3} \qquad (9)$$

where the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are the degrees of freedom and can be chosen arbitrarily.

Geometrical interpretation:

Equations (8) and (9) can be interpretated geometrically in order to indicate the correct way for choosing $\boldsymbol{a}$ and $\boldsymbol{b}$.

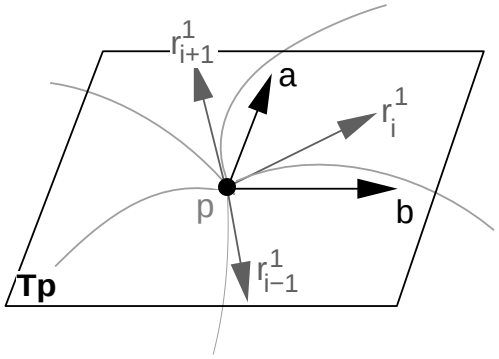

Figure 6: Geometrical interpretation of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

The components $\boldsymbol{r}_i^1$ are the tangents of the boundary curves at the common vertex $\boldsymbol{p}$ which are all lying in the same plane, which is the tangent plane $\boldsymbol{T_p}$ at $\boldsymbol{p}$ (see G^1-condition (4)). $\{\boldsymbol{a}, \boldsymbol{b}\}$ is the basis of the tangent plane and $\boldsymbol{k}_i^a, \boldsymbol{k}_i^b$ are the coordinates of $\boldsymbol{r}_i^1$ in this basis, see fig. 6. These coordinates, $\boldsymbol{k}_i^a = \cos\frac{2\pi i}{n}$, $\boldsymbol{k}_i^b = \sin\frac{2\pi i}{n}$ imply that the first derivative control points $\boldsymbol{b}_1^i = \boldsymbol{b}_0^i + \frac{1}{6}\boldsymbol{r}_i^1$ of the boundary curves form an affine transformation of a plane regular n-gon. This affine transformation is explicitly given by the choice of the tangent plane basis $\boldsymbol{a}$ and $\boldsymbol{b}$.

For practical use of these theoretical considerations it is now important to choose $\boldsymbol{a}$ and $\boldsymbol{b}$ such that the n tangents of the boundary curves in $\boldsymbol{p}$ fit the best. For this purpose we first determine "optimal tangents", denoted $\boldsymbol{r}_{opt}^1$, by a simple heuristic rule and then approach them in a least squares sense. The heuristic rule [12] consists of choosing the optimal tangent vectors $\boldsymbol{r}_{i_{opt}}^1$ as lying in the plane spanned by the edge $\boldsymbol{p}\bar{\boldsymbol{p}}_i$ and a mean normal vector of the mesh facets at $\boldsymbol{p}$, see fig. 7.

These are the tangents which we want to reach by using a linear least squares approximation

$$\|\boldsymbol{r}^1(\boldsymbol{a}, \boldsymbol{b}) - \boldsymbol{r}_{opt}^1\|^2 \to \min_{\boldsymbol{a},\boldsymbol{b}}$$

in order to determine $\boldsymbol{a}$ and $\boldsymbol{b}$.

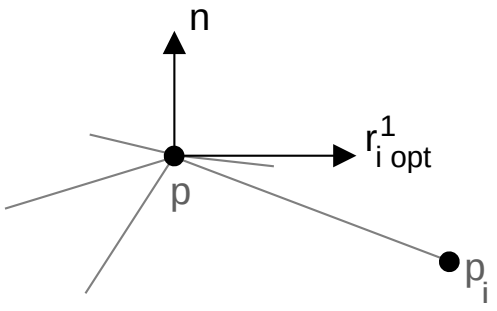

Figure 7: "Optimal" first derivatives.

3.4 Second order derivatives.

A further requirement on the boundary curves is to have second order derivatives at the end points lying in the image space of the matrix T. We can therefore choose any vector $\boldsymbol{r}^2$, such that

$$\boldsymbol{r}^2 = T\bar{\boldsymbol{r}}^2, \qquad (10)$$

where $\bar{\boldsymbol{r}}^2$ is an arbitrary $(n \times 3)$ vector. T is a $(n \times n)$ cyclic matrix. If n (order of the vertex $\boldsymbol{p}$) is odd, then T is invertible, and thus n second derivatives for the n boundary curves $\boldsymbol{r}^2$ can be chosen arbitrarily. And if n is even, T has only rank $n-1$, therefore some chosen n second derivatives, denoted $\boldsymbol{r}^2_{opt}$ can only be approximated, for example through a linear least squares approximation

$$\|\boldsymbol{r}^2 - \boldsymbol{r}^2_{opt}\|^2 \to \min.$$

Since the matrix T has nearly full rank, this least squares minimization approaches quite good the desired second derivatives. As an example, see fig. 12, where the input mesh consists only of vertices of even order.

Note, that the total amount of n resp. $n-1$ vector valued degrees of freedom are available at each vertex. Once these second order derivatives are fixed, i.e. the control points $\boldsymbol{b}_2^i$ are fixed, the whole boundary curve network is entirely fixed. Remember, that each boundary curve is a piecewise cubic curve, which is required to be C^1 continuous, therefore the control points $\boldsymbol{b}_3^i$ and $\boldsymbol{b}_3^k$ of a boundary curve between the vertex $\boldsymbol{p}$ and its neighbour vertex $\boldsymbol{p}_i$ have to be identical and $\boldsymbol{b}_3^i = \boldsymbol{b}_3^k = \frac{1}{2}(\boldsymbol{b}_2^i + \boldsymbol{b}_2^k)$.

3.5 Target middle points & target derivatives.

It's not quite evident how to choose the "optimal" second order derivatives at a vertex $\boldsymbol{p}$ for the boundary curves. On one hand, the scheme should keep a local one, i.e. both curve pieces of a boundary curve have to be constructed independent from each other. On the other hand, fixing $\boldsymbol{b}_2^i$ and $\boldsymbol{b}_2^k$ (k is the index of $\boldsymbol{p}$ in the neighbourhood of $\boldsymbol{p}_i$) determines finally the shape of the boundary curve between $\boldsymbol{p}$ and $\boldsymbol{p}_i$.

In order to cope with both problems we propose the following solution. For each boundary curve we choose a middle control point, called *target point* and a derivative at that point, see fig. 8, which we want to approach or interpolate by the curve. This is equivalent to choosing $\boldsymbol{b}_2^i$ and $\boldsymbol{b}_2^k$. It keeps the scheme local, but allows at the same time a control of the whole curve.
Furthermore, these control handles offer an intuitive way for controlling the curve network. Optimal values can be found either by using a heuristic rule, or by minimizing some energy norm.

Note, that for odd order vertices, these optimal target handles can be interpolated exactly, while otherwise they have to approximated in a least squares sense.

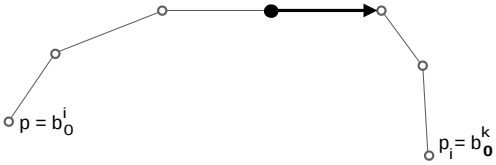

Figure 8: Target point and target vector.

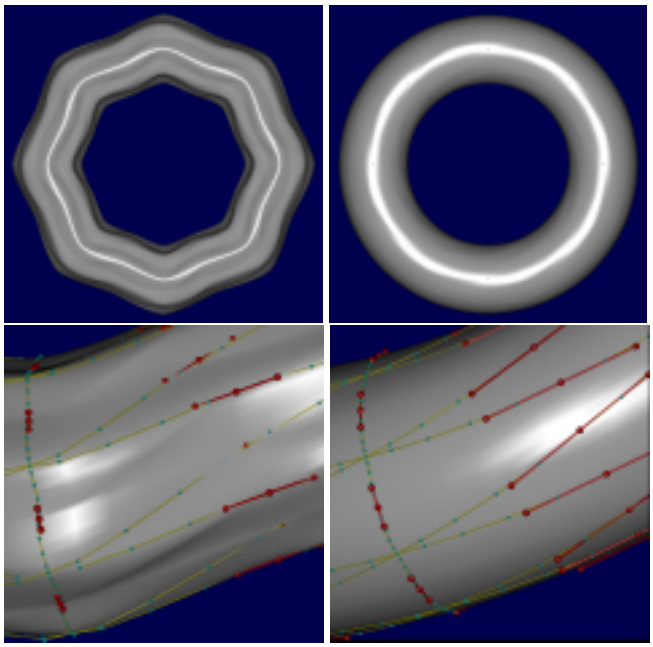

Figure 9: Torus mesh, left with bad target points, right with correct target points.

Figure 9 illustrates how important it is to have a maximum number of degrees of freedom for the second order derivatives. On the right some optimal target points and target derivatives have been approached. They are shown in red with respect to a cubic parametrization. The smooth surface is also shown together with the control polygon of the boundary curves (blue). They are shown as degree 5 curves (two times degree elevated). The sensitivity of the interpolation scheme is shown on the left figures, where some "bad" target points and tangent have been chosen.

3.6 Twist information.

Once the vectors $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ are determined the system (T) can be solved for the twist vectors $\boldsymbol{t}$. An explicit expression can be given as follows

$$\boldsymbol{t} = \Phi^1(\bar{\boldsymbol{k}}^a \boldsymbol{a} + \bar{\boldsymbol{k}}^b \boldsymbol{b}) + \Phi^0 \bar{\boldsymbol{r}}^2. \tag{11}$$

This expression is used in the next step of the algorithm, which is the construction of the cross-boundary tangent ribbons along the boundary curve network. Here it is important to know the twists explicitly, because otherwise it would not be possible to provide an explicit expression of the final surface. For this purpose and the last setp of filling-in the macro-patches C^1-continuously one proceeds analogously to the 4-split method, see sections 2.3 and 2.4.

Remark 1: normal interpolation Since we are able to exactly choose the tangent plane basis vectors, all one has to do in order to interpolate a given normal vector at the vertex is to choose these two tangent plane vectors orthogonal to that given normal.

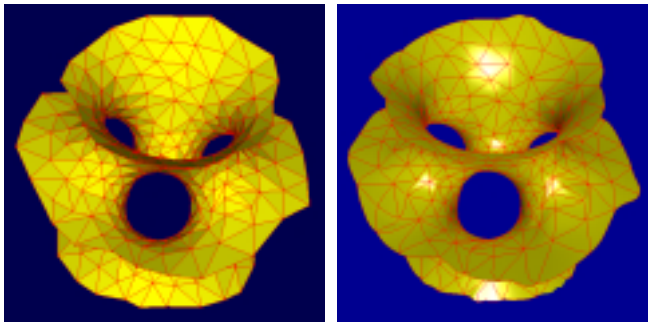

Figure 10: Polyhedral mesh – smooth surface with boundary curve network.

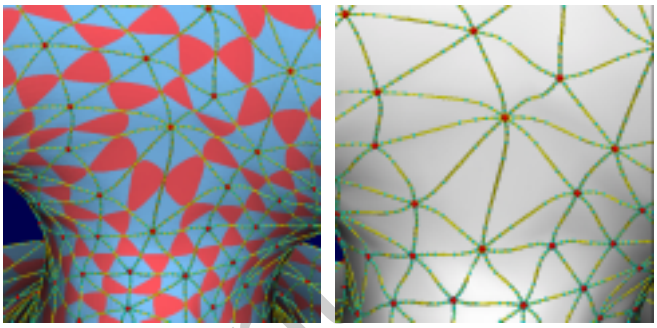

Figure 11: control polygon – smooth surface with boundary curves.

Remark 2: Locality of the scheme The scheme is local in the sense that moving one vertex will only affect the macro-patches around that vertex, and more over will leave the tangent ribbons of the opposite edges to that vertex unchanged.

4 RESULTS

The first example stresses the ability to interpolate meshes with arbitrary topology. The input mesh is available at `ftp://ftp.cs.washington.edu/pub/graphics/meshes`. Fig. 10-left shows the input mesh. Fig. 10-right shows the resulting G^1 surface together with the boundary curves. The order of the vertices ranges from 4 to 8 in this example. Fig. 11-left shows the control polygons of each of the triangular Bézier patches. Different colors have been used in order to distinguish the four Bézier patches per macro-patch. The curve control polygons are also highlighted, while the interpolated points are shown in red. Fig. 11-right shows the surface together with the boundary curve control polygons. It should be pointed out that we used a very crude treatment for the surface border and some additional work should be done on how to handle open meshes. Fig. 12, 13 show at the top an input mesh, in the middle the resulting surface together with the interpolated points in red, and the bottom the control polygons. For the torus input mesh in Fig. 14 a heuristic rule in order to find optimal second derivatives $\boldsymbol{r}^2_{opt}$ (see sect. 3.4) has been used. Even though all vertices have an even order, which means less degrees of freedom in the choice of the second derivatives, the torus is nearly reproduced.

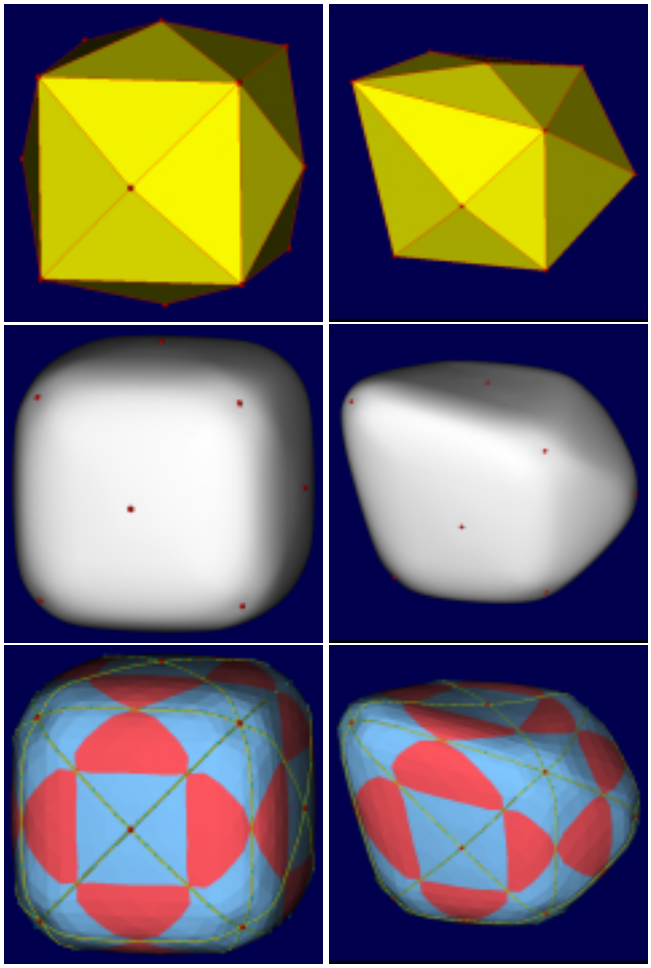

Figure 12: Figure 13:
Top to bottom: polyhedral mesh, smooth surface, control polygon.

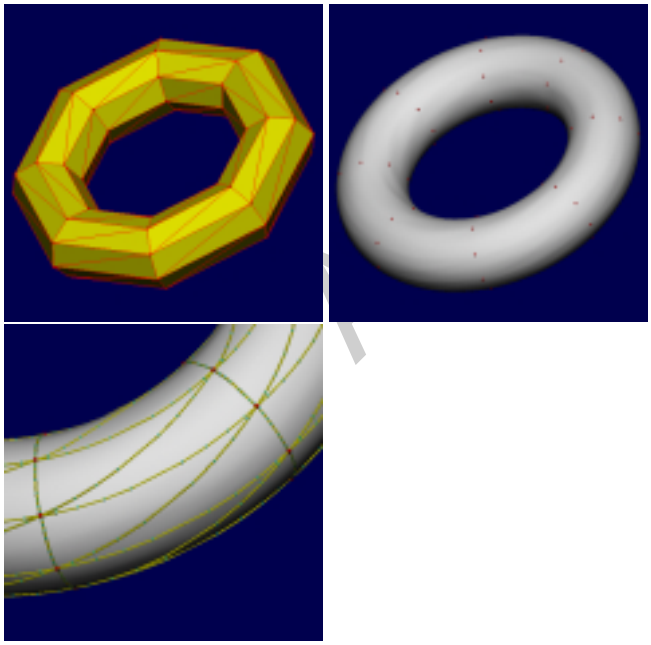

Figure 14: Polyhedral mesh – smooth surface – smooth surface with boundary curves.

References

[1] C. Bajaj. Smoothing polyhedra using implicit algebraic splines. *Computer Graphics*, 26(2):79–88, 1992.

[2] P. Davis. *Circulant Matrices*. Wiley, 1979.

[3] G. Farin. A construction for visual C^1 continuity of polynomial surface patches. *Computer Graphics and Image Processing*, 20:272–282, 1982.

[4] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, New York, 1997.

[5] J. A. Gregory. *N-sided surface patches*, pages 217–232. Clarendon Press, Oxford, 1986.

[6] H. Hagen. Geometric surface patches without twist constraints. *Computer Aided Geometric Design*, 3:179–184, 1986.

[7] S. Hahmann and G-P. Bonneau. Triangular G^1 interpolation by 4-splitting domain triangles. *Computer Aided Geometric Design*, to appear.

[8] C. Loop. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11:303–330, 1994.

[9] S. Mann, C. Loop, M. Lounsbery, D. Meyers, T. DeRose J. Painter and, and K. Sloan. *A survey of parametric scattered data fitting using triangular interpolants*, pages 145–172. SIAM, 1992.

[10] G. Nielson. *A transfinite, visually continuous, triangular interpolant*, pages 235–246. SIAM, 1987.

[11] J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7:221–246, 1991.

[12] B. Piper. *Visually smooth interpolation with triangular Bézier patches*, pages 221–233. SIAM, 1987.