Localizing the 4-split method for G^1 free-form surface fitting

Stefanie Hahmann, Georges-Pierre Bonneau, Riadh Taleb

to be published in Computing, 2001

Abstract. *One common technique for modeling closed surfaces of arbitrary topological type is to define them by piecewise parametric triangular patches on an irregular mesh. This surface mesh serves as a control mesh which is either interpolated or approximated. A new method for smooth triangular mesh interpolation has been developed. It is based on a regular 4-split of the domain triangles in order to solve the vertex consistency problem. In this paper a generalization of the 4-split domain method is presented in that the method becomes completely local. It will further be shown how normal directions, i.e. tangent planes, can be prescribed at the patch vertices.*

---

1. INTRODUCTION

Numerous areas of application such as geometric modeling, scientific visualization, medical imaging need to pass a surface through a set of data points. Not all of them need smooth surfaces. In geometric design, however, it is often desirable to produce visually smooth surfaces, i.e. surfaces with continuously defined tangent planes. Closed surfaces of arbitrary topological type can generally not be defined as a map of a domain on $\mathbb{R}^2$ into $\mathbb{R}^3$ without introducing undesirable singularities. Defining a surface on a triangulated mesh where every patch is the image of one domain triangle allows for arbitrary topological types.

The problem of constructing a parametric triangular G^1 continuous surface interpolating an irregular mesh in space has been considered by many. All methods are local and can be classified depending on how they solve the vertex consistency problem, which occurs

Laboratoire LMC-CNRS, University of Grenoble, B.P. 53, F-38041 Grenoble cedex 9, France, Tel: +33 4 76 63 57 88, Fax +33 4 76 63 12 63, `Stefanie.Hahmann@imag.fr`

when joining with G^1 continuity an even number of C^2-patches around a vertex: there are Clough-Tocher domain splitting methods [2, 9, 15, 16], convex combination schemes [4, 5, 6, 13], boundary curve schemes [14, 10], algebraic methods [1], singular parametrizations [12], quasi G^1 interpolants [11].

Recently another type of triangular interpolation schemes has been developed [7], which can be called *triangular 4-split method.* A regular domain triangle 4-split leads to the construction of four quintic Bézier patches which form a macro-patch in one-to-one correspondence to a mesh face. They have one polynomial degree less than Loop's scheme [10] but one degree more than Piper's [15] or Shirman-Séquin's methods [16]. The triangle 4-split is a new approach in parametric triangular mesh interpolation and has several advantages, as explained in chapter 3.1. It is also a local scheme in that changes of a vertex in the surface mesh only modify a small number of surface patches. But it is not *completely local.* Complete locality would mean that changes of a mesh vertex only affect the patches incident to this vertex. This is a very important property, because more the scheme is local, the more it is well adapted for use in an interactive design system. Real-time modifications of a complex object require minimum computation and display time.

The main aim of the present paper is to generalize the triangular 4-split method in order to make it completely local. A welcome side effect is that interpolation of tangential data will now be possible. In section 2, the vertex consistency problem is described, and some notations are introduced. In section 3, it is shown how the G^1 interpolation/approximation scheme introduced in [7] can be made completely local by using a virtual neighbourhood for each input vertex. Section 4 shows how the complete locality can be used to interpolate tangent planes, or to optimize the shape of the output G^1 surface. Eventually, section 5 gives some examples.

2. THE VERTEX CONSISTENCY PROBLEM

2.1 Notations

For the purpose of this paper, a *control mesh* $\mathcal{M}$ is a set of vertices, edges and triangular faces that describe an oriented 2-manifold in $\mathbb{R}^3$. The number of faces or edges incident to a vertex is referred to as the *order* of a vertex. The set of vertices sharing the edges incident to a vertex is called *vertex neighbourhood.*

In one-to-one correspondence to the mesh faces a collection of triangular patches joining each other with tangent plane continuity is constructed. They are called *macro-patches* M^i and consist of four C^1 continuous triangular Bézier patches. The resulting piecewise polynomial surface $\mathcal{S}$ can either be an approximation to the control mesh or it can interpolate the vertices of $\mathcal{M}$. In both cases, it can optionally interpolate given normal directions at the patch vertices. The reader is supposed to be familiar with Bézier curves and triangular Bézier patches, otherwise details can be found in [3].

2.2 G^1-conditions

When constructing a network of polynomial patches with G^1 continuity special attention has to be paied to what happens at the patch vertices. For this reason, the parameterization of the macro-patches has been chosen as illustrated in Fig. 1. Each macro-patch M^i is the image of the unit triangle in $I\!R^2$.

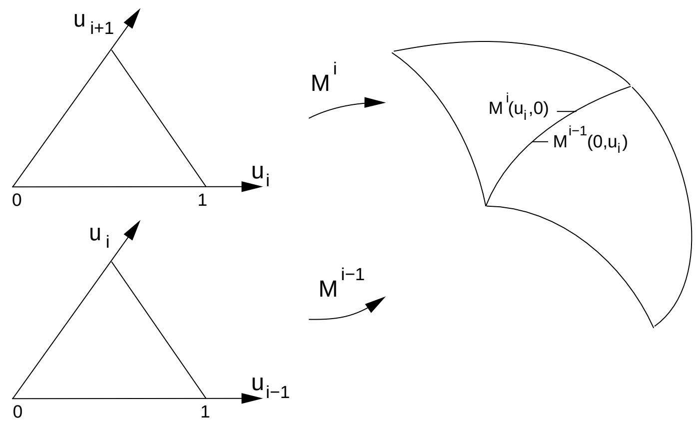

Figure 1: *parameterization.*

The index $i = 1, \ldots, n$ is taken modulo n, where n is the order of the mesh vertex corresponding to $M^i(0,0)$.

Let $M^{i-1}(u_{i-1}, u_i)$ and $M^i(u_i, u_{i+1})$ be two adjacent patches that share a common boundary, i.e. $M^{i-1}(0, u_i) = M^i(u_i, 0)$ for $u_i \in [0,1]$. M^{i-1} and M^i meet with G^1 continuity if there exists a scalar function Φ_i such that

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

These simplified G^1-conditions are used in order to keep the scheme of as low degree as possible.

Difficulties can now arise when joining several polynomial patches together around a common vertex with G^1 continuity. This problem has been mentioned by several authors and can be called *vertex consistency problem* [14] or *twist compatibility problem* [17]. At a vertex, where n patches meet, G^1 continuity can generally not be achieved by simply solving the linear system of n equations (C). The G^1 continuity at such a vertex is directly related to the twists. For polynomial patches, which lie in the continuity class C^2, both twists are identical:

$$\frac{\partial^2 M^i}{\partial u_i \partial u_{i+1}}(0,0) = \frac{\partial^2 M^i}{\partial u_{i+1}\partial u_i}(0,0)\ , \qquad i = 1,\ldots,n\ .$$

Therefore, additional conditions at the patch corner, which involve the twists, have to be satisfied for G^1 continuity of a network of patches:

$$\frac{1}{2}\,\frac{\partial^2 M^i}{\partial u_i \partial u_{i+1}}(0,0) + \frac{1}{2}\,\frac{\partial^2 M^{i-1}}{\partial u_{i-1}\partial u_i}(0,0) = \Phi^1\,\frac{\partial M^i}{\partial u_i}(0,0) + \Phi^0\,\frac{\partial^2 M^i}{\partial u_i \partial u_i}(0,0), \quad i = 1,\ldots,n \tag{1}$$

where $\Phi^0 := \Phi_i(0)$ and $\Phi^1 := \Phi_i'(0)$ are further simplifying assumptions to the G^1-conditions in the present paper. System (1) is obtained by differentiating (C) with respect to u_i taken at $u_i = 0$.
In matrix notation, the system (1) states as follows

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

where

$$T = \begin{bmatrix} \frac{1}{2} & 0 & \cdots & & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \cdots & & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \qquad \bar{\boldsymbol{r}}^1 = \begin{bmatrix} \frac{\partial M^1}{\partial u_1}(0,0) \\ \vdots \\ \frac{\partial M^n}{\partial u_n}(0,0) \end{bmatrix}, \qquad \bar{\boldsymbol{r}}^2 = \begin{bmatrix} \frac{\partial^2 M^1}{\partial u_1 \partial u_1}(0,0) \\ \vdots \\ \frac{\partial^2 M^n}{\partial u_n \partial u_n}(0,0) \end{bmatrix},$$

and $\bar{\boldsymbol{t}}$ is the vector of the twists.

This system can have singularities when n is even, due to the circulant structure of the matrix T, when solving for the twists. In that case, special solutions of the first and second derivatives of the boundary curves at the vertex have to be found in order to get a solution of system (T). In the present paper a similar solution to [7, 10] is given. It consists of determining patch boundary curves, such that the first and second derivative vectors $\bar{\boldsymbol{r}}^1, \bar{\boldsymbol{r}}^2$ lie in the image space of the matrix T. More details in chapter 3.2.

3. GENERALIZED TRIANGLE 4-SPLIT INTERPOLATION METHOD

3.1 Domain 4-split

Let us first introduce the basic idea of the present method as introduced in [7]. We aim to construct a piecewise polynomial surface interpolating or approximation the control mesh $\mathcal{M}$ by a *complete local method*. The vertex consistency problem will be solved by constructing first a network of boundary curves subject to (T). An obvious requirement on these curves is therefore, that their first and second derivatives at the vertices should be independent from each other. This implies that a boundary curve corresponding to an edge of $\mathcal{M}$ has to be of degree 5 at least. Another important requirement on the curve network is to keep the polynomial degree as low as possible.

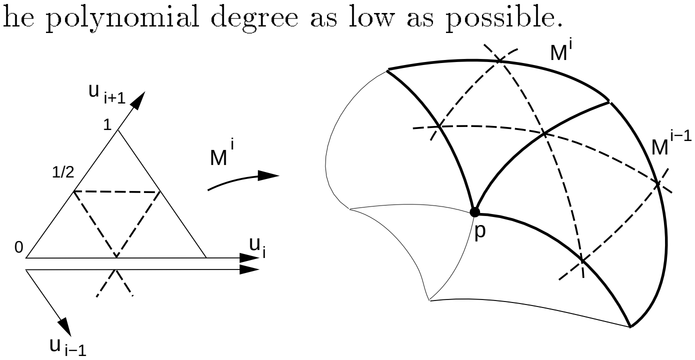

Figure 2: *4-split of the domain triangles.*

To this intend a regular 4-split of the domain triangle is introduced as shown in Fig. 2. It enables us to take piecewise polynomial boundary curves of degree 3 instead of 5. The twist compatibility system (T) can now be solved for each vertex independently. This means, that for each vertex of $\mathcal{M}$ the cubic boundary curve pieces corresponding to the edges incident to that vertex can be constructed independently from the joining curves pieces.

In general, the 4-split allows to construct not only a low degree curve network but also low degree cross-boundary tangents, which finally leads to a piecewise quintic surface spline. Each macro-patch M^i is composed of four quintic Bézier triangles.

3.2 Boundary curve network and cross-boundary tangents

A network of curves being the boundary curves of the macro-patches is constructed in correspondence to the edges of $\mathcal{M}$. These curves are so-called *twist compatible* curves because they satisfy twist compatibility conditions (T).

Each boundary curve between two adjacent patches is a piecewise (2 pieces) cubic Bézier curve parameterized on $\{0, \frac{1}{2}, 1\}$. Around each vertex of $\mathcal{M}$ the control points $\boldsymbol{b}_0^i, \boldsymbol{b}_1^i, \boldsymbol{b}_2^i$, $i = 1, \ldots, n$, of all incident boundary curves are constructed independently from the joining curve piece of the opposite vertices. The "midpoints" $\boldsymbol{b}_3^i$ are then constructed in order to have C^1 boundary curves. See fig. 3 for the notations.

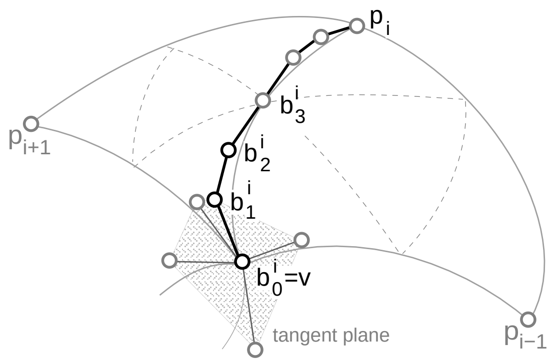

Figure 3: *control points of the boundary curves at vertex* $\boldsymbol{v}$.

At a vertex $\boldsymbol{v}$ the Φ_i-functions which are defined on the incident edges to $\boldsymbol{v}$, are first determined by calculating $\Phi_i(0)$ and $\Phi_i(1)$ from system (C) by solving it for $u_i = 0$ and $u_i = 1$ resp., which gives $\Phi^0 = \Phi_i(0) = \cos(\frac{2\pi}{n})$ and $\Phi_i(1) = 1 - \cos(\frac{2\pi}{n_i})$. The domain 4-split now enables to seperate vertex derivatives and to take the Φ_i-function piecewise linear

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

Let us now adopt a matrix notation for the boundary curve control points between $\boldsymbol{v}$ and

$\boldsymbol{p}_i$, $i = 1, \ldots, n$:

$$\bar{\boldsymbol{b}}_0 := \begin{bmatrix} \boldsymbol{b}_0^1 \\ \vdots \\ \boldsymbol{b}_0^n \end{bmatrix}, \quad \bar{\boldsymbol{b}}_1 := \begin{bmatrix} \boldsymbol{b}_1^1 \\ \vdots \\ \boldsymbol{b}_1^n \end{bmatrix}, \quad \bar{\boldsymbol{b}}_2 := \begin{bmatrix} \boldsymbol{b}_2^1 \\ \vdots \\ \boldsymbol{b}_2^n \end{bmatrix}, \quad \bar{\boldsymbol{p}} := \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_n \end{bmatrix}, \quad \bar{\boldsymbol{v}} := \begin{bmatrix} \boldsymbol{v} \\ \vdots \\ \boldsymbol{v} \end{bmatrix},$$

where $\bar{\boldsymbol{p}}$ is referred to as the *vertex neighborhood* of $\boldsymbol{v}$.

The following choice for the boundary curve Bézier points near the vertex $\boldsymbol{v}$ satisfies the G^1-conditions (C). Simultanously, these control points lie in the image space of matrix T, and therefore allow to solve the system (T) for the twists. See also [7, 10]:

$$\begin{aligned} \bar{\boldsymbol{b}}_0 &= \alpha\bar{\boldsymbol{v}} + B^0\bar{\boldsymbol{p}}\,, \\ \bar{\boldsymbol{b}}_1 &= \alpha\bar{\boldsymbol{v}} + B^1\bar{\boldsymbol{p}}\,, \\ \bar{\boldsymbol{b}}_2 &= \big[(\gamma_0+\gamma_1)\alpha + \frac{\gamma_2}{3}\big]\bar{\boldsymbol{v}} + B^2\bar{\boldsymbol{p}}\,, \quad \gamma_0+\gamma_1+\gamma_2 = 1, \end{aligned} \tag{3}$$

where B^0, B^1, B^2 are $n \times n$ matrices defined by

$$\begin{aligned} B^0_{ij} &= \frac{1-\alpha}{n} \\ B^1_{ij} &= \frac{1-\alpha+\beta\cos\left(\frac{2\pi(j-i)}{n}\right)}{n} \\ B^2_{ij} &= \frac{(\gamma_0+\gamma_1)(1-\alpha)+\gamma_1\beta\cos\left(\frac{2\pi(j-i)}{n}\right)}{n} + \gamma_2 \begin{cases} 1/6 & \text{if } j = i-1, i+1 \\ 1/3 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

The free parameters $\alpha, \beta, \gamma_1, \gamma_2$ control the interpolation/approximation of mesh vertices, the first and second derivatives. In [8] it is shown how they can be set optimally. The control points of the joining curves pieces $\boldsymbol{b}_0^k, \boldsymbol{b}_1^k, \boldsymbol{b}_2^k$ and $\boldsymbol{b}_3^k = \boldsymbol{b}_3^i$ are found by applying the formulas (3) and (4) to the neighbouring mesh points $\boldsymbol{p}_i$ of $\boldsymbol{v}$. k is the index of $\boldsymbol{v}$ relative to the neighborhood of $\boldsymbol{p}_i$.

The cross-boundary tangents are subject to the G^1 conditions (C), the vertex consistency constrains (T) and the curve network and are set to be equal

$$\begin{aligned} \frac{\partial M^i}{\partial u_{i+1}}(u_i, 0) &= \Phi_i(u_i)\,\frac{\partial M^i}{\partial u_i}(u_i, 0) + \Psi_i(u_i)\boldsymbol{V}_i(u_i), \\ \frac{\partial M^{i-1}}{\partial u_{i-1}}(0, u_i) &= \Phi_i(u_i)\,\frac{\partial M^i}{\partial u_i}(u_i, 0) - \Psi_i(u_i)\boldsymbol{V}_i(u_i). \end{aligned} \tag{5}$$

The scalar function Ψ_i and the vector function $\boldsymbol{V}_i$ are built of minimal degree so as to interpolate the values of the cross-derivatives and the twists at the vertices $\boldsymbol{v}$ and $\boldsymbol{p}_i$:

$$\begin{aligned} \Psi_i(u_i) &= \sin\frac{2\pi}{n}(1-u_i) + \sin\frac{2\pi}{n_i}u_i & & (\textit{linear}) \\ \boldsymbol{V}_i(u_i) &= \sum_{k=1}^{n} \boldsymbol{v}_i^k B_k^2(2u_i) \quad u_i \in [0, \frac{1}{2}]\,, & & (\textit{piecewise quadratic}) \end{aligned} \tag{6}$$

where

$$\bar{\boldsymbol{v}}_0 = V^0\bar{\boldsymbol{p}}, \quad \bar{\boldsymbol{v}}_1 = V^1\bar{\boldsymbol{p}}, \quad \bar{\boldsymbol{v}}_2 = \frac{1}{2}\bar{\boldsymbol{v}}_1 + \frac{1}{2}\bar{\boldsymbol{v}}_1^k. \tag{7}$$

The $n \times n$ matrices V^0 and V^1 are given by

$$V_{ij}^0 = \frac{6\beta}{n}\sin(\frac{2\pi(j-i)}{n}), \quad i,j = 1,\dots,n,$$

$$\begin{aligned} V_{ij}^1 = {} & \frac{1}{\psi_i^0}\left[(6\phi^1 - 48\phi^0 + 24\phi^0)\tan(\frac{\pi}{n}) - 6\psi_i^1\right]\frac{\beta}{n}\sin(\frac{2\pi(j-i)}{n}) \\ & + \frac{4}{\psi_i^0}\gamma_2\phi^0 \begin{cases} 1 & \text{if } j = i+1 \\ -1 & \text{if } j = i-1 \end{cases}, \end{aligned} \tag{8}$$

where $\Phi^0 = \Phi_i(0), \Phi^1 = \Phi_i'(0)$ and $\Psi_i^1 = \Psi_i'(0)$ are known from (2) and (6).

Although the boundary curves and the cross-boundary tangents are piecewise cubic, the marco-patches will be piecewise quintic. With quartic patches a vertex consistency problem could occur at the boundary mid-points which are supplementary vertices of order 6 (see domain triangle 4-split). This problem is automatically solved by the special choice of the cross-boundary tangents (7).

The explicit Bézier representation of the boundary curves is already known. In order to obtain quintic curves two degree elevations have to be performed of (3). Some further simple calculations combining (5)-(8) with (3) are necessary to get the first inner row of Bézier points of the macro-patches from the cross-boundary tangents. The formulas are explicitly given in [7].

3.3 Filling-in the macro-patches

Each macro-patch is composed of four quintic triangular Bézier patches. The boundary curves of a macro-patch are the twice degree elevated curves of section 3.2. The cross-boundary tangents of sect. 3.2 determine the first inner row of control points after one degree elevation. The remaining 15 inner control points, which are highlighted in fig. 4, are used for joining the four inner patches with C^1 continuity. Six of them can be chosen arbitrarily.

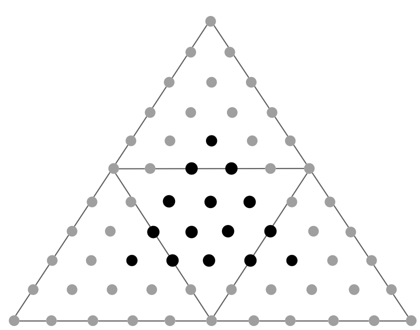

Figure 4: *15 free inner control points making macro-patches C^1.*

3.4 Complete locality

The present triangle 4-split method has several properties. It is an *affine invariant* scheme because only affine combinations of mesh vertices of $\mathcal{M}$ are used. An *explicit closed form* Bézier representation of the quintic patches is known. Several *shape parameters* and free control points are available for local shape modifications or shape optimizations. And the present method is *local* in that changes of a mesh vertex of $\mathcal{M}$ only affect a small number of patches.

Locality is a very important property for interpolation schemes. The advantages of the locality are obvious: the algorithms are generally numerically stable, no linear systems of equations have to be solved here. The algorithms are fast, because local modifications of the input data imply only local updates of the interpolating surface. Interactive real-time modeling of 3D objects with a large and complicate input mesh becomes possible.

It is easy to see, that the 4-split method is local: the algorithm works in two steps. First the boundary curves and cross-boundary tangents are constructed piecewise around each vertex and then joined together. For each vertex $\boldsymbol{v}$ the incoming curve pieces (fig. 5) are calculated by using only the local neighbourhood points $\bar{\boldsymbol{p}} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]^T$ of $\boldsymbol{v}$, see equations (3) and (7) and the icosahedon example in fig. 6(a). Once the boundary curves and cross-boundary tangents are fixed a second step of the algorithm consists of calculating the remaining inner control points for each macro-patches locally, i.e. independent from the neighbouring macro-patches.

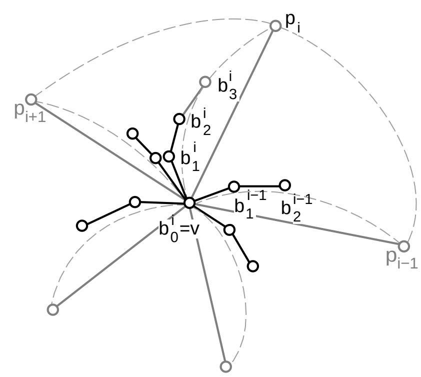

Figure 5: *boundary curves incident to $\boldsymbol{v}$. A first step of the algorthm consists of calculating these curve piece for each vertex, idem for the cross-boundary tangents, and of joining them together in the middle.*

Modification of a mesh vertex $\boldsymbol{v}$ has therefore influence on all the macro-patches having $\boldsymbol{v}$ in common, and on all the macro-patches having the neighbourhood points $\bar{\boldsymbol{p}} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]^T$ of $\boldsymbol{v}$ in common. This is due to the boundary curve construction, as mentioned below. For the icosahedron example, fig. 6(b) shows all boundary curves which are concerned, when vertex $\boldsymbol{v}$ is modified.

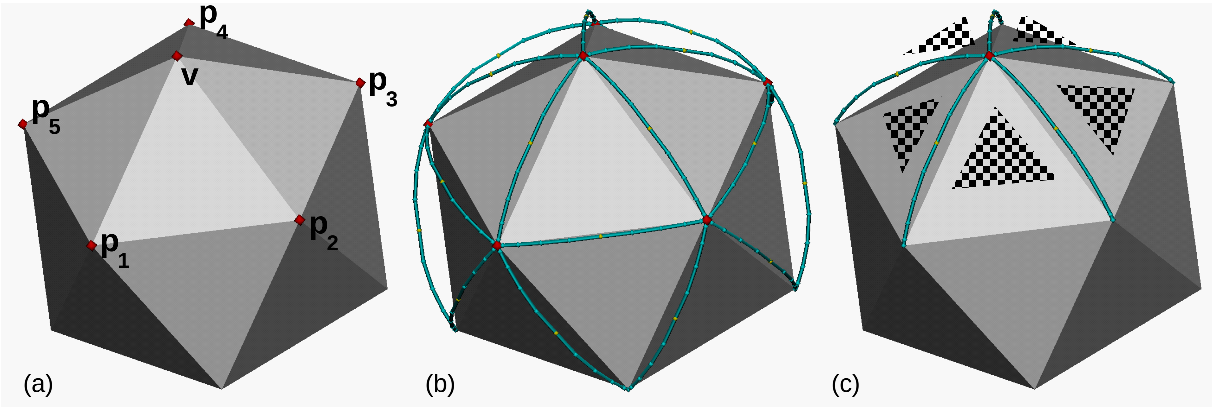

Figure 6: *The input mesh is a regular polyhedra, an icosahedron. (a) local neighbourhood points $\boldsymbol{p}_i$ of a mesh vertex $\boldsymbol{v}$. (b) boundary curves which depend on vertex $\boldsymbol{v}$. The control polygons of the piecewise degree five curves are shown. (c) macro-patches and boundary curves depending on vertex $\boldsymbol{v}$ when using the concept of virtual neighbourhoods in the algorithm.*

It turns out, that this method is not as local as it is desirable and useful. It will now be shown, that it is possible to *generalize the 4-split method* in order to make it a complete local interpolation scheme. *Complete local* means here, that modifications of a mesh vertex $\boldsymbol{v}$ only influence the n (order of $\boldsymbol{v}$) macro-patches incident to $\boldsymbol{v}$, see fig. 6(c). To this end the control points of the boundary curves, $\bar{\boldsymbol{b}}_0, \bar{\boldsymbol{b}}_1, \bar{\boldsymbol{b}}_2$, and the control points of the cross-boundary tangents $\bar{\boldsymbol{v}}_0, \bar{\boldsymbol{v}}_1$ should be made independent of the vertex neighbourhood $\bar{\boldsymbol{p}}$ of $\boldsymbol{v}$. It can be observed that satisfying the G^1-condition for $\bar{\boldsymbol{b}}_0, \bar{\boldsymbol{b}}_1, \bar{\boldsymbol{b}}_2$ and $\bar{\boldsymbol{v}}_0, \bar{\boldsymbol{v}}_1$ doesn't depend on a particular choice of $\bar{\boldsymbol{p}}$. The curve control points $\boldsymbol{b}_1^i$ for example satisfy the G^1-conditions because they are the result of an first order Fourier approximation of n distinct points. Thus they make the first derivative of the boundary curves lying in the image space of (T). Similarly for the others. Furthermore, the construction of the boundary curve pieces (3) and the cross-boundary tangent pieces (7) is local around a mesh vertex $\boldsymbol{v}$. The vertex neighbourhood $\bar{\boldsymbol{p}}$ can therefore be replaced by another new "virtual" neighbourhood $\bar{\boldsymbol{p}}^* = [\boldsymbol{p}_1^*, \ldots, \boldsymbol{p}_n^*]^T$. The following equations replace (3) and (7) in the algorithm.

New boundary curve's Bézier points:

$$\begin{aligned} \bar{\boldsymbol{b}}_0 &= \alpha\bar{\boldsymbol{v}} + B^0\bar{\boldsymbol{p}}^*\,, \\ \bar{\boldsymbol{b}}_1 &= \alpha\bar{\boldsymbol{v}} + B^1\bar{\boldsymbol{p}}^*\,, \\ \bar{\boldsymbol{b}}_2 &= \big[(\gamma_0+\gamma_1)\alpha + \frac{\gamma_2}{3}\big]\bar{\boldsymbol{v}} + B^2\bar{\boldsymbol{p}}^*\,, \quad \gamma_0+\gamma_1+\gamma_2 = 1, \end{aligned} \tag{9}$$

$$\bar{\boldsymbol{v}}_0 = V^0\bar{\boldsymbol{p}}^*, \quad \bar{\boldsymbol{v}}_1 = V^1\bar{\boldsymbol{p}}^*, \quad \bar{\boldsymbol{v}}_2 = \frac{1}{2}\bar{\boldsymbol{v}}_1 + \frac{1}{2}\bar{\boldsymbol{v}}_1^k, \tag{10}$$

where the matrices B^0, B^1, B^2, V^0, V^1 are given by (4) and (8). Doing this for all mesh vertices finally leads to a complete local mesh fitting scheme.

4. CHOICE OF VIRTUAL VERTEX NEIGHBOURHOOD

Up to now the 4-split method calculates first and second order derivative informations of the surface $\mathcal{S}$ at a vertex $\boldsymbol{v}$ by using the n neighbourhood points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ of $\boldsymbol{v}$. They are vertices of the input mesh and are therefore not free. In the generalized method, presented in the previous chapter, this set of n points can be chosen arbitrarily for each mesh vertex. How these novel degrees of freedom can be used in order to obtain pleasing shapes or in order to create shape design handles is now shown in the following subsections.

4.1 How many degrees of freedom ?

By replacing the true neighbourhood points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ by the virtual neighbourhood points $\boldsymbol{p}_1^*, \dots, \boldsymbol{p}_n^*$ additional degrees of freedom are created at ewach vertex. They can be used either for normal vector interpolation at the mesh vertices or for surface fairing methods. The number of degrees of freedom depends on how the points $\boldsymbol{p}_i^*$ are combined in equation (9) and (10). The number of degrees of freedom for calculating the first derivative points in (9) for example is equal to the rank of the matrix B^1, which is equal 2. How these two vector valued degrees of freedom can be employed is subject of the next sections.

4.2 Interpolation of normal vector input

The tangent plane of $\mathcal{S}$ at a vertex $\boldsymbol{v}$ is spanned by the points $\boldsymbol{b}_1^i$, $i = 1, \dots, n$ and $\boldsymbol{v}$. These $n+1$ points are all lying in the same plane, since the boundary curve's first deriveatives

$$\frac{\partial M^i}{\partial u_i}(0,0) = 6(\boldsymbol{b}_1^i - \boldsymbol{v})$$

satisfy the G^1 conditions at the vertex $\boldsymbol{v}$. The Bézier control points are obtained from a weighted averaging of the virtual neighbourhood points $\boldsymbol{p}_i^*$ given by:

$$\bar{\boldsymbol{b}}_1 = \begin{bmatrix} \boldsymbol{b}_1^1 \\ \vdots \\ \boldsymbol{b}_1^n \end{bmatrix} = \boldsymbol{v} + \frac{\beta}{n} \begin{bmatrix} \sum_{j=1}^n \cos(\frac{2\pi(j-1)}{n})\boldsymbol{p}_j^* \\ \vdots \\ \sum_{j=1}^n \cos(\frac{2\pi(j-n)}{n})\boldsymbol{p}_j^* \end{bmatrix}.$$

In other words, the normal vector of $\mathcal{S}$ at a vertex $\boldsymbol{v}$ is the weighted combination of the normal vectors of the n planes spanned by $\{\boldsymbol{v}, \boldsymbol{p}_i^*, \boldsymbol{p}_{i+1}^*\}$. The weights come from the G^1 conditions, but the points $\boldsymbol{p}_i^*$ are free. It is therefore possible to interpolate a given normal vector at the mesh vertices. To this end, the oriented points $\boldsymbol{p}_i^*$ have to lie in a plane together with $\boldsymbol{v}$, which is orthogonal to the given normal vector.

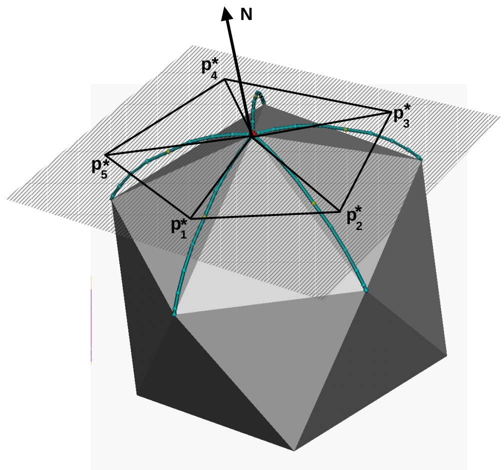

Figure 7: *the virtual neighbourhood points $\boldsymbol{p}_i^*$ lie in a plane together with the vertex $\boldsymbol{v}$ orthogonal to $\boldsymbol{N}$ in order to make the surfae interpolating the given normal vector $\boldsymbol{N}$.*

4.3 Shape optimization

By replacing the true vertex neighbourhood of a mesh vertex $\boldsymbol{v}$ by the new points $\boldsymbol{p}_i^*$, $i = 1, \ldots, n$, which are free, extranous degrees of freedom have been created for the whole surface scheme. If normal direction interpolation is not desired, the points $\boldsymbol{p}_i^*$ can be determined by some optimization process on the curve network.

The shape of the resulting surface depends mainly on the shape of the boundary curves. A "well shaped" curve network should f.ex. avoid undulations. The free virtual neighbourhood $\bar{\boldsymbol{p}}^*$ and the free curve shape parameters $\beta, \gamma_1, \gamma_2$ are available for each mesh vertex. They can be determined by local or global optimization on the curve network by using some minimum norm criteria, like energy functionals. Based on this concept of virtual neighbourhood points, the paper [8] proposes and tests various appropriate criteria for shape optimizations.

5. COMPLETE LOCAL FITTING OF ARBITRARY MESHES

A first example, fig. 8, simply illustrates the surface construction steps: input mesh, boundary curves, filling-in the macro-patches. The input mesh is a triangulated regular polyhedron with 12 vertices on the unit sphere, called icosahedron. The boundary curves are computed first, see fig. 8a. The parameter α is set to one, the mesh vertices are therefore interpolated. The macro-patches are then filled-in with four quintic Bézier patches each. The resulting surface is shown in fig. 8b. The shape parameters and free inner control points have been chosen in order to approximate the unit sphere. The L_∞-error between the surface and the unit sphere is 0.0033. An isophote analysis in fig. 8c shows the **global smoothness** of the spline surface.

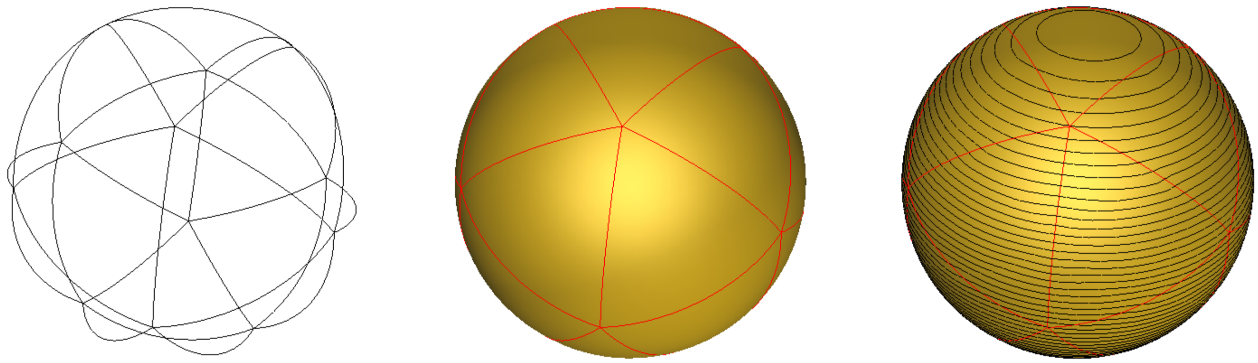

Figure 8: *interpolated icosahedron with isophote analysis.*

It is then possible to choose other **shape parameters**, which are stretching the boundary curves and flattening the macro-patches, see fig. 9a or rounding out the curves and patches, see 9b.

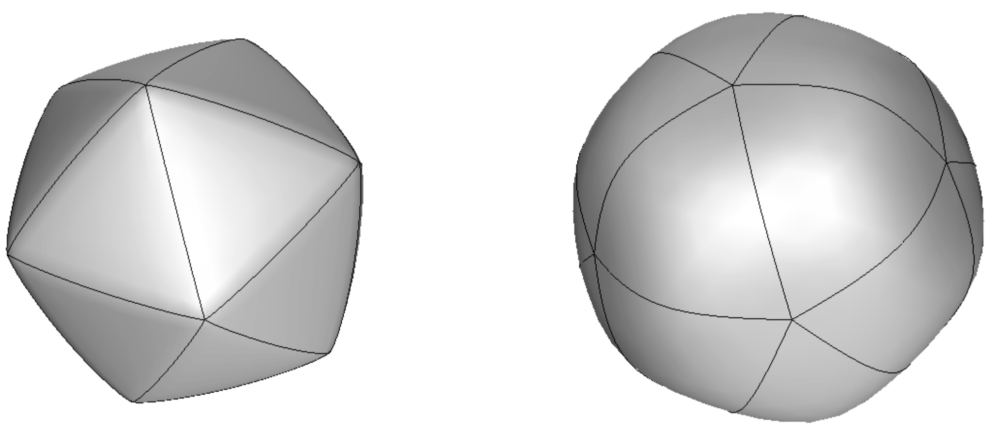

Figure 9: *interpolated icosahedron with different shape parameters.*

The **complete locality** of the surface scheme is illustrated on the icosahedron example in fig. 10. In both examples one mesh vertex has been modified and it can be observed that only the surface macro-patches incident to this vertex have been modified, see fig. 10b,d. The left image of each example shows the control nets of the Bézier patches. The four patches of each macro patch are colored individually, see fig. 10a,c.

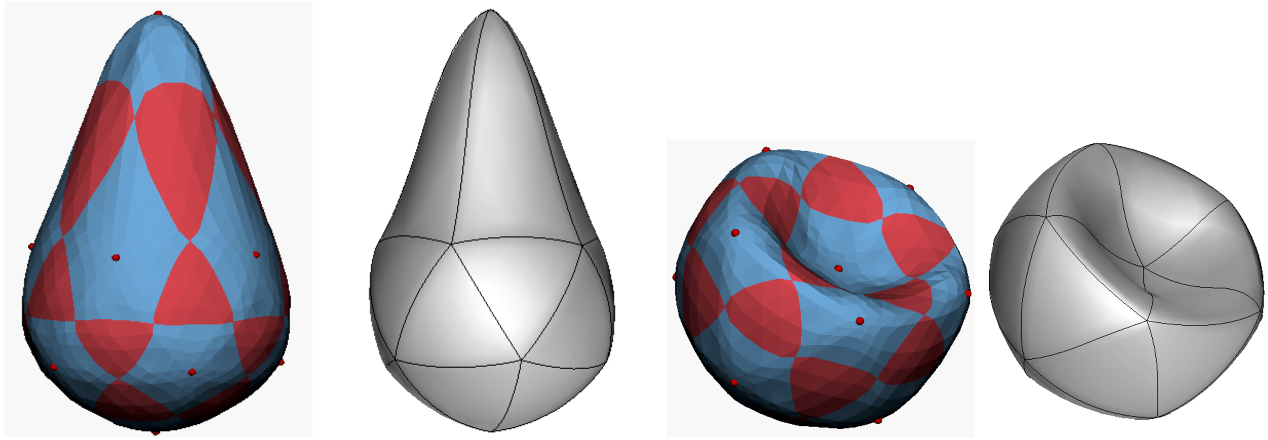

Figure 10: *locally modified icosahedron.*

The next example, fig. 11, shows another surface with vertices of order 6 and 4. Additional to the input mesh normal directions are interpolated at the mesh vertices. They are shown as gray lines in fig. 11. The shape parameters are fixed automatically by a local form optimization method (section 4.3).

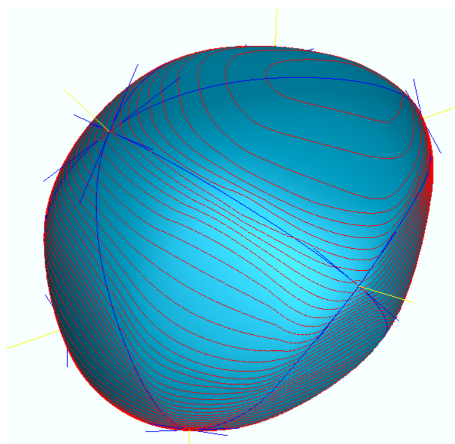

Figure 11: *normal interpolation, vertices of order 4 and 6.*

REFERENCES

1. Bajaj C., Smoothing polyhedra using implicit algebraic splines, Computer Graphics **26** (2) (1992), 79–88.
2. Farin G., A construction for visual C^1 continuity of polynomial surface patches, Computer Graphics and Image Processing **20** (1982), 272–282.
3. Farin G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, New York, 4th edition, (1997).
4. Gregory J.A., N-sided surface patches, in *The Mathematics of Surfaces* J. Gregory (ed.), Clarendon Press, Oxford (1986), 217–232.

5. Hagen H., Geometric surface patches without twist constraints, Computer Aided Geometric Design **3** (1986), 179–184.
6. Hagen H., Pottmann H., Curvature continuous triangular interpolants, in *Mathematical Methods in Computer Aided Geometric Design* Tom Lyche, Larry L. Schumaker (eds.), Academic Press, New York (1989), 373–384.
7. Hahmann S., Bonneau G-P., Triangular G^1 interpolation by 4-splitting domain triangles, Computer Aided Geometric Design **17** (2000), 731–757.
8. Hahmann S., Bonneau G-P., Taleb R., Smooth irregular mesh interpolation, in *Curve and Surface Fitting: Saint-Malo 1999* Albert Cohen, Christophe Rabut, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, (2000), 237–246.
9. Jensen T., Assembling triangular and rectangular patches and multivariate splines, in *Geometric Modeling: Algorithms and new Trends*, G. Farin (ed.), SIAM (1987), 203–220.
10. Loop C., A G^1 triangular spline surface of arbitrary topological type, Computer Aided Geometric Design **11** (1994), 303–330.
11. Mann S., Surface approximation using geometric Hermite patches, Ph.D. dissertation, University of Washington, (1992).
12. Neamtu M., Pluger P., Degenerate polynomial patches of degree 4 and 5 used for geometrically smooth interpolation in $\mathbb{R}^3$, Computer Aided Geometric Design **11** (1994), 451–474.
13. Nielson G., A transfinite, visually continuous, triangular interpolant, in *Geometric Modeling: Algorithms and new Trends*, G. Farin (ed.), SIAM (1987), 235–246.
14. Peters J., Smooth interpolation of a mesh of curves, Constructive Approximation **7** (1991), 221–246.
15. Piper B.R., Visually smooth interpolation with triangular Bézier patches, in *Geometric Modeling: Algorithms and new Trends*, G. Farin (ed.), SIAM (1987), 221–233.
16. Shirman L.A., Séquin C.H., Local surface interpolation with Bézier patches, Computer Aided Geometric Design **4** (1987), 279–295.
17. Van Wijk J.J., Bicubic patches for approximating non-rectangular control meshes, Computer Aided Geometric Design **3** (1986), 1–13.