Smooth Irregular Mesh Interpolation

Stefanie Hahmann, Georges-Pierre Bonneau,
and Riadh Taleb

Abstract. The construction of a smooth surface from an irregular mesh in space is considered. The mesh vertices can either be interpolated or approximated as a control net. A collection of triangular Bézier patches results from a local, affine invariant and visually smooth interpolation scheme that can represent surfaces of arbitrary topological type. It is based on a domain 4-split. Beside the surface construction scheme, the optimal employment of the numerous degrees of freedom is crucial for an overall pleasing shape. Different local minimum norm criteria are tested to see if they produce satisfactory shapes.

§1. Introduction

The general problem of constructing a parametric triangular G^1 continuous surface interpolating an irregular mesh in space has been considered by many authors. In [5] a survey of such schemes is given, and it is concluded that local polynomial interpolants have similar shape defects due to the absence of an optimization strategy for using the free parameters (a special solution has been proposed in [6] for one of these schemes).

A different method has been developed by Loop [4] producing a collection of patches that meet each other with G^1 continuity. The vertex enclosure problem, which occurs when joining with G^1 continuity an even number of polynomial patches around a vertex, is solved by first constructing C^2-consistent boundary curves and cross-boundary tangents and then filling in the patches. In one-to-one correspondence to the mesh faces, sextic triangular Bézier patches are constructed, which lead to a very small number of degrees of freedom. One scalar value per vertex controls the length of the tangents of the boundary curves at the end points and one control point per patch is free. This is not enough for sufficient control of the shape of a sextic patch. In [5], it was stated that well-shaped boundary curves are a necessary condition. Loop's scheme doesn't provide any influence on the second derivatives of the boundary curves, which can lead to undulations. It was therefore proposed to

Curve and Surface Fitting: Saint-Malo 1999
Albert Cohen, Christophe Rabut, and Larry L. Schumaker (eds.), pp. 237–246.

ISBN 0-8265-1357-3.

relax the interpolation condition, which leads to an extra free parameter per vertex controlling the distance from the patch vertices to their corresponding mesh vertices. This is clearly improving the shape.

Recently, another triangular interpolation scheme has been developed [3]. A regular domain 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, but one degree more than Piper's or Shirman-Séquin's method [8,9]. The domain 4-split is a new approach in triangular mesh interpolation, and has several obvious advantages: the boundary curves and cross-boundary tangents are piecewise polynomial. They can therefore be of low degree and simultaneously separate first and second derivatives of the boundary curves of the macro-patch corners from the neighbours. The scheme is completely local. Furthermore, the 4-split leads to four patches per macro-patch which leaves enough control points free for inner shape control. Finally this scheme offers two parameters per vertex for controlling first and second derivatives of the boundary curves, and six free control points inside the macro-patch. Additionally, the interpolation condition can also be relaxed to gain one more free parameter per vertex.

The present paper investigates the problem of how the free parameters and control points of the 4-split domain method can be employed optimally. The challenge is to get an overall satisfactory shape, which is a global requirement, while maintaining the locality property of the scheme. Various geometric and variational criteria are proposed and compared.

§2. Triangular G^1 Interpolation by 4-splitting Domain Triangles

2.1 Notations

The surface mesh $\mathcal{M}$ is input, and consists of a list of vertices and edges describing a 2-manifold triangulated mesh in $\mathbb{R}^3$. The surface $\mathcal{S}$ which interpolates the vertices of $\mathcal{M}$ is composed of triangular macro-patches M^i which are in one-to-one correspondence with the mesh facets. It is therefore convenient for the construction of $\mathcal{S}$ to choose a parameterization of the macro-patches M^i around a common vertex, sharing pairwise a common boundary as illustrated in Fig. 1. All subscripts $i = 1, \ldots, n$ are taken modulo n, where n is the order of the mesh vertex corresponding to $M^i(0,0)$. The parameter u_i lies in the interval $[0,1]$.

The fundamental idea of the present triangular interpolation scheme is to subdivide the domain triangle into four subtriangles by joining the edge midpoints together, see Fig. 1. Each macro-patch M^i will therefore be a piecewise polynomial image of the unit triangle in $\mathbb{R}^2$, composed of four quintic Bézier triangles [2] each. The macro-patches will join together with G^1 continuity. The resulting surface $\mathcal{S}$ will also be G^1.

The G^1 conditions which are used in this paper are subject to some simplifying assumption in order to keep the interpolation scheme of low degree. Two adjacent patches $M^{i-1}(u_{i-1}, u_i)$ and $M^i(u_i, u_{i+1})$ join at a common boundary

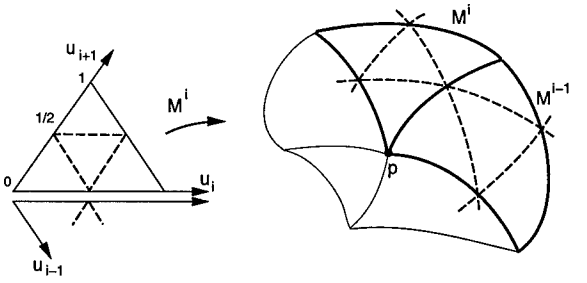

Fig. 1. Parameterization and domain 4-split.

with G^1 continuity if there exists a scalar function Φ_i such that

$$\Phi_i(u_i)\, M^i_{u_i}(u_i, 0) = \frac{1}{2}\, M^i_{u_{i+1}}(u_i, 0) + \frac{1}{2}\, M^{i-1}_{u_{i-1}}(0, u_i)\,, \tag{1}$$

where n is the order of the vertex corresponding to $u_i = 0$. $M^i_{u_i}$ denotes the partial derivative of M^i with respect to u_i.

The algorithm for constructing the spline surface consists of three steps

- constructing boundary curves,
- constructing cross-boundary tangents,
- filling in the patches,

which will be briefly presented in the following three subsections. For more details and complete explanations of this method, the reader is referred to [3].

It is important to keep all these functions of the lowest degree possible. The main contribution to this comes from the domain 4-split. It allows for piecewise polynomial functions of low degree while simultaneously fulfilling all other requirements, such as continuity and localness.

2.2. Boundary curves and vertex consistency

First the boundary curves of the macro-patches are constructed in correspondence to the edges of $\mathcal{M}$ by interpolating the mesh vertices at the end points by satisfying the G^1 conditions at the vertices and by keeping the surface scheme local. They are called C^2-consistent.

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 $\boldsymbol{b}^i_0, \boldsymbol{b}^i_1, \boldsymbol{b}^i_2$, $i = 1, \ldots, n$, of all incident boundary curves are constructed independently from the joining curve piece of the opposite vertices. The "midpoints" $\boldsymbol{b}^i_3$ are then constructed in order to have C^1 boundary curves. See Fig. 2 for the notation.

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 (1) by solving it for $u_i = 0$ and $u_i = 1$ resp., which gives $\Phi^0 = \Phi_i(0) = \cos\left(\frac{2\pi}{n}\right)$

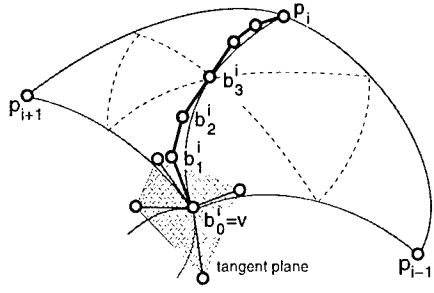

Fig. 2. Control points of the boundary curves at vertex $\boldsymbol{v}$.

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 to be piecewise linear:

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

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 $\boldsymbol{v}$ enables us to find a solution to system (1) which at the same time solves the vertex consistency problem [3,4]:

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

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

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

The free parameters $\alpha, \beta, \gamma_1, \gamma_2$ control the interpolation and the first and second derivatives. In Section 3 it will be 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 $\boldsymbol{v}$.

This curve network construction is local in that changes of one mesh vertex only affect the boundary curve pieces relative to the neighbourhood of that vertex.

2.3. Cross-boundary tangents

The cross-boundary tangents are subject to the G^1 conditions (1), the vertex consistency constraints, and the curve network of Sect. 2.2, and are set to be equal

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

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 $\boldsymbol{p}$ and $\boldsymbol{p}_i$:

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

where

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

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

$$V^0_{ij} = \frac{6\beta}{n} \sin\Big(\frac{2\pi(j-i)}{n}\Big), \quad i, j = 1, \ldots, n,$$

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

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

2.4. Filling-in the macro-patches

Each macro-patch is composed of four C^1 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

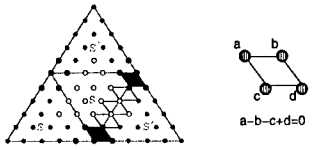

Fig 3. C^1-conditions between two adjacent quintic Bézier patches.

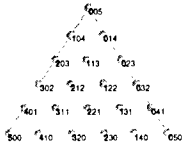

Fig 4. Labelling the control points of a quintic Bézier patch.

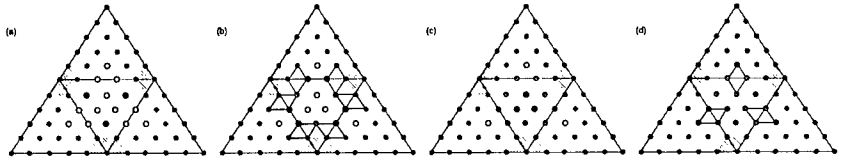

Fig. 5. Four steps for filling in the macro-patch M with C^1-continuity.

inner row of control points after one degree elevation [3]. The remaining 15 inner control points, which are highlighted in Fig. 5a, 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. 3: all pairs of adjacent triangles must form a parallelogram. In [3] it was shown that the first and last pairs of adjacent triangles in Fig. 3 already form parallelograms.
It remains to compute the free Bézier points such that the other three pairs of triangles along each edge inside the macro-patch also form parallelograms. This is done in four steps:

- choose the three twists points of the internal Bézier patch arbitrarily; these are free shape parameters (see Fig. 5a),
- compute the third and fourth Bézier points along each internal curve joining two Bézier patches using the second and fourth parallelogram conditions (see Fig. 5b),
- choose the remaining three unknown Bézier points of the central patch arbitrarily; these are free shape parameters (see Fig. 5c),
- compute the three remaining unknown Bézier points of the outer patches using the third parallelogram condition along each edge (see Fig. 5d).

§3. Local Optimization of the Boundary Curves

The present triangular interpolation method offers several degrees of freedom for shape control. They can be set manually or by using simple heuristics. An interactive design system can allow for manually adjusting these parameters in order to improve the shape. This procedure seems not to be sufficient if the given triangulated point set is very large, or if the data points are irregularly distributed.

Our goal is to investigate some optimization techniques. Two groups of degrees of freedom have to be distinguished. First there are 4 scalar parame-

ters per vertex controlling the curve network, then 6 free inner Bézier control points are available for each macro-patch. Let us first concentrate on the curve network. It was stated in [5] that triangular interpolants often suffer from undulating curve networks. It can be confirmed here that a "well shaped" curve network is not sufficient, but is necessary for the construction of a pleasing shape. As pointed out in Sect. 2, the 4-split method is local. This property should not be altered by an optimization procedure. Local optimization criteria are therefore needed. This localness requirement conflicts in some sense with the global requirement of a "well shaped" surface. Every local scheme has to accept this conflict, otherwise it loses its localness property. Nevertheless, it will be shown that good results can be obtained.

In detail, four curve parameters per vertex are available, see (3) and (4). α is not really free. It allows us to switch between interpolation and approximation of the surface mesh. At first, we only consider the interpolation problem and set $\alpha = 1.0$. β affects the length of the tangent vectors of the boundary curves at the vertex $\boldsymbol{v} = \boldsymbol{b}_0^i$. The control points $\bar{\boldsymbol{b}}_1$ are obtained by a first order Fourier approximation of the neighbourhood $\boldsymbol{p}_i$, $i = 1, \ldots, n$ of $\boldsymbol{v}$. In other words, the $\boldsymbol{b}_1^i$ are an affine image of a regular n-gon whose centroid is $\boldsymbol{b}_0^i$. Too short tangent vectors lead to sharp corners at the patch vertices, while too long tangent vectors can lead to unwanted undulations. γ_1, γ_2 control the second derivatives of the vertices. The control points $\boldsymbol{b}_2^i$ depend linearly on them. But they don't depend linearly on $\beta, \gamma_1, \gamma_2$.

Due to the previous observations, the optimization of $\beta, \gamma_1, \gamma_2$ should mainly avoid undulations and allow for more or less bent or stretched curves. If for computation-time reasons one wants to perform only linear optimization in a least-squares sense, as we do, it should be done in two steps separating β from γ_1, γ_2.

In the following, the computation of optimal $\beta, \gamma_1, \gamma_2$ is separated into two steps. Otherwise, the problem would become non-linear. Each boundary curve consists of two cubic pieces joining with C^1 continuity. For locality reasons of the whole scheme, the pieces have to be constructed independently one another. In a *first step* the points $\boldsymbol{b}_1^i$ of the boundary curves incident in $\boldsymbol{v}$, Fig. 2, are determined. Each boundary curve corresponds to an edge of the surface mesh. In this case we are looking at the edge connecting $\boldsymbol{v}$ and $\boldsymbol{p}_i$. In order to reproduce the shape inherent to the underlying surface mesh, geometrical considerations imply that $\boldsymbol{b}_1^i$ would optimally lie in the plane spanned by this edge and a vector between $\boldsymbol{v}_0$ and the orthogonal projection of $\boldsymbol{p}_i$ on the tangent plane in $\boldsymbol{b}_0^i$, as Piper does in [8]. We call these points $\bar{\boldsymbol{b}}_1^* = [\boldsymbol{b}_1^{1*}, \cdots, \boldsymbol{b}_1^{n*}]^T$. The tangent planes should be estimated first. The constraints on the boundary curves in the present method don't allow for setting $\bar{\boldsymbol{b}}_1$ equal $\bar{\boldsymbol{b}}_1^*$, this is why these points are approximated in a least-squares sense. The key point here is that the locality of the equations (3) and (7) enables us to replace the true neighbourhood points $\boldsymbol{p}_i$ of $\boldsymbol{v}$ in these equations by new "virtual" neighbourhood points $\boldsymbol{p}_i^*$ that are only used in these equations, i.e. to compute the boundary curves, and the cross-boundary tangents. Therefore, we are able to solve the following linear least-square

problem in order to compute the "virtual" points $\boldsymbol{p}_i^*$:

$$\sum_{i=1}^{n} \|\boldsymbol{b}_i^{1*} - \boldsymbol{b}_i^1\|^2 \longrightarrow \min. \tag{9}$$

The new, optimal, control points are now given by

$$\bar{\boldsymbol{b}}_1^{optimal} = \alpha\bar{\boldsymbol{v}} + B^1\bar{\boldsymbol{p}}^*. \tag{10}$$

In a *second step*, $\bar{\boldsymbol{b}}_2$ has to be determined. $\bar{\boldsymbol{p}} := \bar{\boldsymbol{p}}^*$ and β are already fixed. Two parameters per vertex, γ_1, γ_2 are left free for optimization. $\boldsymbol{b}_3$ is then fixed as the midpoint between $\boldsymbol{b}_2^i$ of the two curve pieces. The requirements on $\bar{\boldsymbol{b}}_2$ are twofold: avoid undulations and bend the curve on request. The second requirement concerns the whole boundary curve between $\boldsymbol{v}$ and $\boldsymbol{p}_i$, and depends on the choice of $\bar{\boldsymbol{b}}_2$. The idea is to cope with that problem by introducing a target point $\boldsymbol{t}$ for each boundary curve. The control points $\boldsymbol{b}_2^i$ of each curve piece are then determined so that $\boldsymbol{b}_3 = \frac{1}{2}(\boldsymbol{b}_2^i + \tilde{\boldsymbol{b}}_2^i)$ approaches the target point by minimizing an appropriate energy functional on each curve piece locally. The introduction of the target point allows for global control of the boundary curves, while still keeping the scheme local.

The target point is fixed by a subdivision rule in terms of $\boldsymbol{b}_0^i, \boldsymbol{b}_1^i$ and $\tilde{\boldsymbol{b}}_0^i, \tilde{\boldsymbol{b}}_1^i$ of the joining curve piece, such as

$$\boldsymbol{t} = \frac{1}{16}(-\boldsymbol{b}_0 + 9\boldsymbol{b}_1^i + 9\tilde{\boldsymbol{b}}_1^i - \tilde{\boldsymbol{b}}_0). \tag{11}$$

$\bar{\boldsymbol{b}}_2$, which depends linearly on the free parameters γ_1, γ_2, is now determined by minimizing the linearized version of the bending energy combined with a curve length component [1]

$$E_\omega = \int_0^1 \|X''(t)\|^2 dt + \omega \int_0^1 \|X'(t)\|^2 dt\,, \quad \omega \geq 0. \tag{12}$$

The solution of a linear 2×2 system gives the optimal values for γ_1, γ_2 for $\bar{\boldsymbol{b}}_2$.

Different ways for finding an optimal, i.e. well shaped, curve network have been studied. Within the local schemes, the concept of target points can be replaced by target tangent vectors. This leads to a $2n \times 2n$ linear system of equations per vertex. $\beta, \gamma_1, \gamma_2$ can also be determined by a non-linear optimization method in only one step. When relaxing the localness requirement of the scheme, plenty of curve network schemes are possible, like a variant of Nielson's MNN [7] or the integration of given curvature values or second fundamental forms at the mesh vertices in the optimization process. This is not a subject of the present paper.

§4. Minimum Norm Criteria for Macro-patches

Once the curve network and the cross-boundary tangents are constructed, 15 inner Bézier control points remain for each macro-patch. They are related to each other by the C^1 continuity conditions which are imposed between the four quintic sub-patches. Six of them are completely free. They are drawn as full black dots in Figs. 5a and 5b. The remaining 9 points depend linearly on

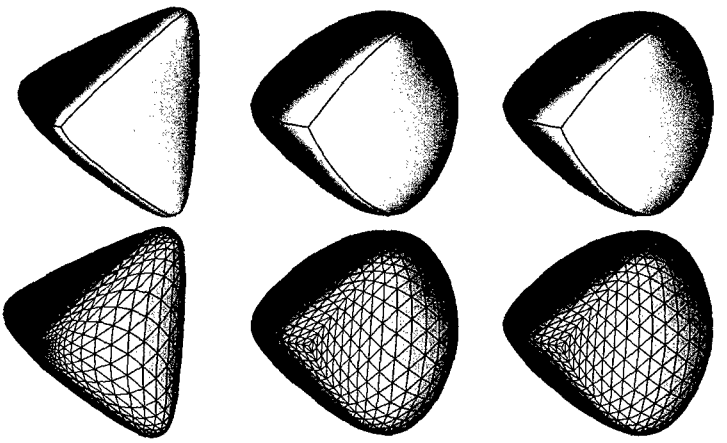

Fig. 6. *Left*: manually fixed free parameters, *Middle*: optimized boundary curves, *Right*: optimized boundary curves and patches.

them. It is therefore possible to use one of the two quadratic functionals $E_k(X,X)$, where $k = 2,3$ and

$$E_k(X,Y) = \int_{\triangle} \left(\frac{\partial^k X}{\partial u^k} \cdot \frac{\partial^k Y}{\partial u^k} + \frac{\partial^k X}{\partial v^k} \cdot \frac{\partial^k Y}{\partial v^k} + \frac{\partial^k X}{\partial w^k} \cdot \frac{\partial^k Y}{\partial w^k} \right) dudvdw.$$

The free points $\bar{\boldsymbol{s}}$ are determined by minimizing

$$\sum_{j=0}^{3} E_k(\boldsymbol{S}^j, \boldsymbol{S}^j) = \sum_{j=0}^{3} (M_j \bar{\boldsymbol{s}} + \boldsymbol{a}_j)^T A (M_j \bar{\boldsymbol{s}} + \boldsymbol{a}_j)\,, \quad k = 2,3, \tag{13}$$

where $\boldsymbol{S}^j = \sum_{|\mathbf{i}|=5} \boldsymbol{b}_{\mathbf{i}} B^5_{\mathbf{i}}(u,v,w)$ denote the four quintic Bézier sub-patches, and $\bar{\boldsymbol{s}} = [\boldsymbol{s}^0_{113}, \boldsymbol{s}^0_{131}, \boldsymbol{s}^0_{311}, \boldsymbol{s}^0_{122}, \boldsymbol{s}^0_{212}, \boldsymbol{s}^0_{221}]^t$ denotes the vector of the 6 unknown control points of the middle patch (see Figs. 4 and 3). The (21×6) matrices M_j and the vectors $\boldsymbol{a}_j$ contain the linear relations between control points of the 4 sub-patches and the 6 unknown points. The (21×21) matrix A is given by $A_{\mathbf{ij}} = E_k(B_{\mathbf{i}}, B_{\mathbf{j}})$ for $|\mathbf{i}| = 5$, $|\mathbf{j}| = 5$.

§5. Results

Fig. 6 shows the interpolation of a tetrahedron by our method. This very simple example was chosen because it illustrates clearly the influence of the free parameters and control points. The upper row shows three surfaces with the boundary curves of the macro-patches, while the lower row shows their iso-parametric lines. The left surface is obtained by manually setting $\alpha = 1.0$, $\beta = 0.15$, $\gamma_0 = -1.0$, $\gamma_1 = 2.0$, $\gamma_2 = 0.0$. These values are identical for all vertices due to the regularity of the surface mesh. The free inner control points are set by a rule combining the mesh face normal, cross-boundary tangents

and the boundary curves. The middle surface has optimized boundary curves, (9)-(12). Optimal "virtual" neighbour points are calculated, and the optimal parameters are $\gamma_0 = -1.598$, $\gamma_1 = 2.393$, $\gamma_2 = 0.205$. Face energy without minimization is equal to 732.2. The right surface has the same boundary curves as the previous example, but the free inner control points of the macro-patches are obtained by minimizing the face energy (13) with $k = 3$. The energy decreases to 309.0. The connections between the macro-patches are sharper for the manual setting. An overall more smooth surface results from the optimized parameter setting of this paper. The distribution of the iso-parametric lines shows a positive side-effect: it is more regular at the patch vertices for the optimized surfaces.

References

1. Bonneau, G.-P., and H. Hagen, Variational design of rational Bézier curves and surfaces, in *Curves and Surfaces in Geometric Design*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters, Wellesley MA, (1994), 51–58.
2. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, New York, 4th edition, 1997.
3. Hahmann, S., and G.-P. Bonneau, Triangular G^1 interpolation by 4-splitting domain triangles, to appear in Computer Aided Geometric Design.
4. Loop, C., A G^1 triangular spline surface of arbitrary topological type, Computer Aided Geometric Design **11** (1994), 303–330.
5. Mann, S., C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan, A survey of parametric scattered data fitting using triangular interpolants, in *Curve and Surface Design*, H. Hagen (ed.), SIAM (1992), 145–172.
6. Mann, S., Using Local optimization in surface fitting, in *Mathematical Methods for Curves and Surfaces* Morten Dæhlen, Tom Lyche, Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville & London, 1995, 323–332.
7. Nielson, G., A method for interpolating scattered data based upon a minimum norm network, Mathematics of Computation **40** (1983), 253–271.
8. Piper, B. R., Visually smooth interpolation with triangular Bézier patches, in *Geometric Modeling: Algorithms and New Trends*, G. Farin (ed.), SIAM (1987), 221–233.
9. Shirman, L. A., and C. H. Séquin, Local surface interpolation with Bézier patches, Computer Aided Geometric Design **4** (1987), 279–295.

S. Hahmann, G.-P. Bonneau, and R. Taleb
Laboratoire de Modélisation et Calcul, CNRS-IMAG, B.P. 53
F-38041 Grenoble Cedex 9, France
hahmann@imag.fr