Fair Surface Reconstruction Using Quadratic Functionals

Andreas Kolb[1] Helmut Pottmann[2] Hans-Peter Seidel[1]

[1] IMMD IX, Graphische Datenverarbeitung, Universität Erlangen, Germany
Email: {kolb,seidel}@informatik.uni-erlangen.de

[2] Institut für Geometrie, Technische Universität Wien, Austria
Email: pottmann@egmvs2.una.ac.at

Abstract

An algorithm for surface reconstruction from a polyhedron with arbitrary topology consisting of triangular faces is presented. The first variant of the algorithm constructs a curve network consisting of cubic Bézier curves meeting with tangent plane continuity at the vertices. This curve network is extended to a smooth surface by replacing each of the networks facets with a split patch consisting of three triangular Bézier patches. The remaining degrees of freedom of the curve network and the split patches are determined by minimizing a *quadratic* functional. This optimization process works either for the curve network and the split patches separately or in one simultaneous step. The second variant of our algorithm is based on the construction of an optimized curve network with higher continuity. Examples demonstrate the quality of the different methods.

1 Introduction

The reconstruction of a surface from a set of (a priori unorganized) points as well as the design of surfaces with arbitrary shape (or topology) are of major concern for numerous applications in CAD/CAM-based technologies. Veltkamp [19] gives an overview on existing methods concerning the first topic. In this paper we will address the following surface reconstruction problem which is closely related to both topics:

Given: A non-degenerate polyhedron in 3-space with triangular faces.

Goal: A smooth surface having the same shape (i.e. the same topology) as the polyhedron and interpolating its vertices.

In this context "smooth" means tangent-plane continuous. This problem has been intensively investigated in the last decade. One very popular way of solving the reconstruction problem consists of two steps:

- Each edge of the given polyhedron is replaced by a space curve such that curves with a common end point have the same tangent plane, i.e. they meet with G^1- or with a higher degree of continuity.
- The constructed curve network is filled with triangular Bézier patches interpolating the curve network, such that neighboring patches meet smoothly.

Lounsbery et al [10], Mann [11] and Peters [15] give an overview on methods that are based on these construction steps. Basically, the existing methods can be divided into *heuristic methods* and *optimizing methods.*

The main characteristic of the heuristic methods is that all free parameters of the curves and patches that are not determined by any of the constraints (interpolation conditions/smoothness conditions at vertices and along patch-boundaries) are set by heuristic rules. The resulting algorithms are, in general, very easy to implement. Furthermore they are relatively fast. The major drawback of the heuristic methods is that the resulting surfaces are often of very poor quality (see Mann [11]).

On the other hand, optimizing methods try to overcome the shortcomings of the traditional heuristic methods utilizing the idea of variational design. Nielson [14] proposes a method applying the basic ideas of the minimum norm networks known from functional scattered data interpolation (see [13] and Section 2.2) to the general situation. Nielson uses a quadratic functional to optimize a G^1 curve network. This reduces the optimization process to solving a linear system (see Section 2.1). The curve network is filled with so-called transfinite interpolants, a very special class of rational surfaces not supported by any CAD system. The underlying representation of the curve network yields a relatively large number of unknowns for the optimization. Moreover, this representation can not be extended to curve networks with higher continuity. These seem to be the major disadvantages of this method.

Moreton and Séquin [12] present a method using functional optimization to generate fair surfaces. The results of their scheme is, in general, very appealing. Moreton and Séquin use exact formulas to describe the non-linear constraints necessary to represent a G^2 curve network (curves with a common end point have the same tangent plane and agree with the same second fundamental form at this point; see [6]) and a smooth surface. Optimization based on a non-quadratic objective functional is used to generate an optimized curve network and a smooth surface. As a result of the exact description of the constraints and the non-quadratic objective functional, only iterative methods can be used to determine the optimal parameters. This optimization process is relatively slow.

The algorithm presented in this paper uses a representation of the curve network that not only reduces the number of unknowns for the optimization but is also extensible to curve networks of higher continuity. Furthermore the curve network is filled with Bézier patches using surface optimization. Both, the optimization of the curve network and of the surface, is realized using quadratic functionals. This reduces the optimization process to solving a linear system (see Section 2.1), and significantly speeds up the computation. The resulting algorithm is a combination of the heuristic and the optimization approach.

This paper is organized as follows: In the following section some basic topics are discussed that closely relate to the upcoming algorithm. The main algorithm is presented in Section 3. In Section 3.1 we describe a method for generating an optimized G^1 curve network. A method for fitting an interpolating patchwork to a curve network using surface optimization is given in Section 3.2. Furthermore, we discuss simultaneous fairing of curve network and surface (Section 3.3). In Section 3.4 we describe the construction of a G^2 curve network and its extension to a smooth surface. In Section 4 some examples are given demonstrating the quality of the different schemes. Conclusions are presented in Section 5.

2 Preliminaries

In this section we briefly discuss previous work which is related to material in the subsequent sections. First we describe optimization processes based upon quadratic functionals. In the second subsection the basic idea of *functional* minimum norm networks is given. Finally we review the problem of constructing smooth surfaces starting with a given curve network.

2.1 Quadratic Functionals

Our goal is to reduce the process of optimization to that of solving a linear system in the unknown free parameters. This can be done by using a quadratic functional to approximate the exact, but highly non-linear functional that, e.g., describes the curvature or the variation of curvature of a curve or surface. Greiner [5] shows how these approximative functionals can be chosen in order to achieve a result close to that of the exact functionals.

Moreover, we must guarantee that our curve network or surface depends *linearly* on the free parameters which we want to optimize.

If both requirements are fulfilled, the problem of finding the set of parameters having optimal value with respect to the quadratic functional reduces to the problem of finding the solution to a linear system of equations (see [3]).

Further on, optimization always means optimization with quadratic functionals and linear dependency of the free parameters.

2.2 Functional Minimum Norm Networks

The functional *Minimum Norm Network (MNN)* method is designed to solve the surface reconstruction problem in the functional situation, i.e., in the case that the polyhedron is given by a triangulation in the (x,y)-plane plus a set of real numbers describing the z coordinate of each vertex. The underlying triangulation serves as global parameterization of the curve network. Using this parameterization yields a representation of the curve network by means of the (unknown) partial derivatives of the curve network at the vertices. This representation of the curve network depends linearly on these unknowns, which is an essential property needed for the optimization process (see Section 2.1).

2.3 Smooth Surface Fitting

The problem of fitting an overall smooth surface consisting of Bézier patches to a given curve network consisting of Bézier curves is well-known. Peters [15] classifies the different algorithms solving this fitting problem. Naturally, one wants to solve the problem with minimal degree triangular Bézier patches and with as few patches as possible. If the underlying curve network is G^1, the problem of fitting one non-degenerate Bézier patch to each facet of the curve network has, in general, no solution (see [16]). One way to overcome this is to fill each facet with several Bézier patches, e.g. by using the split scheme described by Shirman and Séquin [18]. Alternatively one can try to generate a G^1 curve network with admissible data, i.e. a curve network that guarantees a solution consisting of one non-degenerate Bézier patch per facet. Loop [9] describes a method doing this. Yet another possibility is to use networks with higher continuity. A G^2 network can always be extended to a smooth surface with only one Bézier patch per facet (see [15]). In general, this implies a higher polynomial degree of the curve network and thus the total degree of the surface increases as well.

3 Our Algorithm

As mentioned above, we restrict ourselves to the case of interpolating the vertices and the topology of a polyhedron with triangular faces. The first variant of our algorithm is a two-step approach: First, the construction of a curve network and then the fitting of Bézier patches in the holes to this curve network.

The representation of our G^1 curve network is based on the idea of a functional MNN (see Section 2.2). Due to the unrestricted topology of our polyhedron we can not, in general, parameterize our curve network globally. The introduction of *local coordinates* for each vertex solves this problem (see Section 3.1). This yields a representation of a G^1 curve network consisting of cubic Bézier curves which is linearly dependent on the remaining degrees of freedom.

The G^1 curve network is extended to an overall smooth surfaces using the well-known *Shirman-Séquin* split scheme [18] (see Section 3.2). This scheme fits *three* degree *four* triangular Bézier patches, i.e. a 3×4 split patch, to each opening of the curve network. Investigating the Shirman-Séquin split scheme we find that there are three vector-valued degrees of freedom (cross-boundary derivatives) for each 3×4 split patch which can be utilized for the surface optimization.

As an alternative to the two-step method, it is possible to optimize the curve network and the surface simultaneously, i.e. all free parameters (those for the curve network and those for the 3×4 split patch) are determined by one optimization process (see Section 3.3).

The final variant of our algorithm presented in Section 3.4 is based on a G^2 curve network. Cross-boundary vector-fields are introduced into the representation of the curve network yielding a more surface-like network description. The G^2 curve network is filled with one degree seven triangular Bézier patch per facet.

3.1 The G^1 Curve Network

In this section we describe the details of the construction and optimization of the cubic G^1 curve network. Our representation of the curve network is based on the idea of local coordinates. For each vertex of the polyhedron we estimate a normalized vector* $\hat{\mathbf{w}}$ that represents the local z coordinate of the curve network. $\hat{\mathbf{w}}$ is estimated by computing the weighted sum of the normals of all triangles meeting at the considered vertex. The weights are either the sizes of the triangles (or their reciprocal values) or their angles at the vertex. In the plane normal to $\hat{\mathbf{w}}$ two perpendicular vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ (*local coordinates*) are defined. The $(\hat{\mathbf{u}}, \hat{\mathbf{v}})$-plane is referred to as *local parameter plane.*

Remember that in the case of a functional MNN the underlying triangulation serves as parameterization of the curve network. Similarly, we define a local parameterization of the curve network in each vertex. The actual tangent plane of the curve network is described with help of partial derivatives $s_{\hat{\mathbf{u}}}, s_{\hat{\mathbf{v}}}$ in the local coordinates $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ and the vectors $\vec{\mathbf{u}}^* = \hat{\mathbf{u}} + s_{\hat{\mathbf{u}}}\hat{\mathbf{w}}$ and $\vec{\mathbf{v}}^* = \hat{\mathbf{v}} + s_{\hat{\mathbf{v}}}\hat{\mathbf{w}}$ (see Figure 1). Thus, the actual normal vector of the curve network is given by

$$\hat{\mathbf{n}} = \frac{\hat{\mathbf{w}} - (s_{\hat{\mathbf{u}}}\hat{\mathbf{u}} + s_{\hat{\mathbf{v}}}\hat{\mathbf{v}})}{\sqrt{1 + s_{\hat{\mathbf{u}}}^2 + s_{\hat{\mathbf{v}}}^2}}.$$

Furthermore, a tangent vector $\vec{\mathbf{t}}^*$ with parametric direction $\vec{\mathbf{t}} = \alpha\hat{\mathbf{u}} + \beta\hat{\mathbf{v}}$ is determined by $\vec{\mathbf{t}}^* = \vec{\mathbf{t}} + (\alpha s_{\hat{\mathbf{u}}} + \beta s_{\hat{\mathbf{v}}})\hat{\mathbf{w}}$ (see Figure 1).

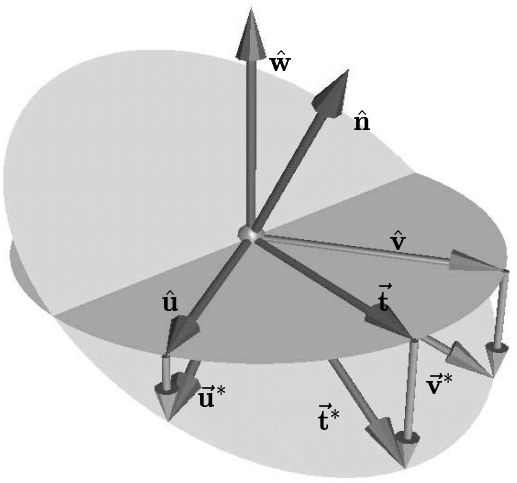

Figure 1: Local coordinates.

In the following we describe the different methods used to determine the parametric directions $\vec{\mathbf{t}}$ of the curve corresponding to the edge $\overline{\mathbf{V}_0\mathbf{V}_1}$ at the vertex $\mathbf{V}_0$. This process is split into two steps:

*$\hat{\mathbf{v}}$ denotes a vector of unit length, $\vec{\mathbf{v}}$ an arbitrary vector and $\mathbf{V}$ a point in 3-space

(1) determine the direction $\hat{\mathbf{t}}$ of $\vec{\mathbf{t}}$ and

(2) determine the length of $\vec{\mathbf{t}}$.

Known methods to compute the direction $\hat{\mathbf{t}}$ are:

Normal Projection: The edge $\overline{\mathbf{V}_0\mathbf{V}_1}$ is projected onto the local parameter plane at $\mathbf{V}_0$.

Plane Curve: A plane is defined containing both the edge $\overline{\mathbf{V}_0\mathbf{V}_1}$ and the vector $\frac{1}{2}(\hat{\mathbf{w}}_0 + \hat{\mathbf{w}}_1)$; $\hat{\mathbf{t}}$ is computed as the intersection of this plane with the local parameter plane at $\mathbf{V}_0$ (see Figure 2).

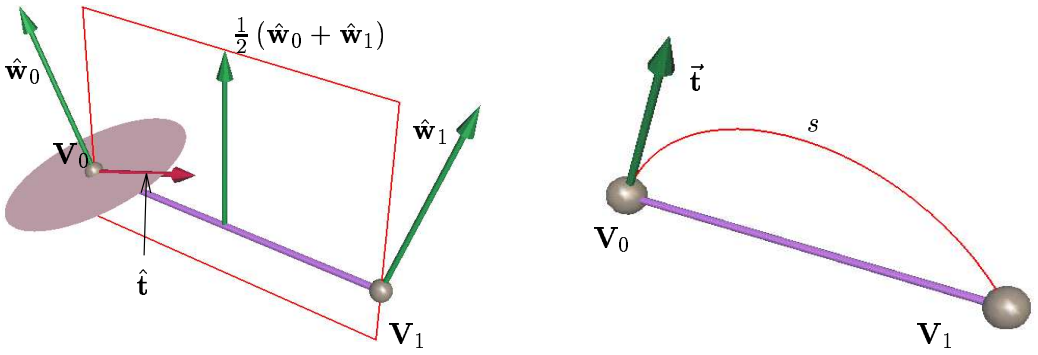

Figure 2: The plane curve- and the circle segment method.

To set the length of $\vec{\mathbf{t}}$ methods can be used as follows:

Chord Length: Set $\left\|\vec{\mathbf{t}}\right\| = \|\overline{\mathbf{V}_0\mathbf{V}_1}\|$.

Circle Segment: The positions $\mathbf{V}_0$, $\mathbf{V}_1$ and the direction $\hat{\mathbf{t}}$ define a unique circular segment s; we set $\left\|\vec{\mathbf{t}}\right\| = arc(s)$ (see Figure 2).

DeBoor-Höllig-Sabin: This method requires additional information about the curvature at $\mathbf{V}_0$ and $\mathbf{V}_1$; a detailed discussion of the construction of this type of curve is given by de Rose and Mann [2]; we use the norm of the tangents of the DeBoor-Höllig-Sabin-curve to set the length of $\vec{\mathbf{t}}$.

The curves of the network are defined as cubic Bézier curves interpolating the corresponding vertex-positions and the tangents with parameteric directions as described above. This representation of the curve network depends linearly on the not yet specified partial derivatives $s_{\hat{\mathbf{u}}}, s_{\hat{\mathbf{v}}}$ in the local coordinates.

The unknowns $s_{\hat{\mathbf{u}}}, s_{\hat{\mathbf{v}}}$ in our curve network representation are set by applying a minimization process using a quadratic functional. A widely used quadratic functional to optimize a curve $\mathbf{C}$ is

$$\sigma(\mathbf{C}) = \int \|\mathbf{C}''(t)\|^2 \, dt.$$

The functional used to optimize the network is simply the summation of the functionals for each curve. Optimization is then performed by solving a linear system for the unknown derivatives $s_{\hat{\mathbf{u}}}$ and $s_{\hat{\mathbf{v}}}$ for each vertex.

We iterate this process by using the normals calculated in the optimization as new $\hat{\mathbf{w}}_i$'s for the next iteration step. Practice shows that this iteration converges in a small number of steps independently of the initial estimation of the vectors $\hat{\mathbf{w}}_i$.

3.2 Interpolatory Surface Fitting

The optimized G^1 curve network as described in Section 3.1 serves as input for the process of surface fitting. We use the 3×4-split-patches given by Shirman and Séquin [18] for this job.

The tangent plane continuity between adjacent split patches across a network curve is enforced by the Chiyokura-Kimura constraints [1]. Farin's method [4] is used to ensure the smooth patch-patch joints in the interior of the 3×4-split-patch. Analyzing Shirman and Séquin's method reveals that there is a total of five scalar and three vector-valued degrees of freedom not set by the smoothness constraints. The three vector-valued degrees of freedom $\vec{\mathbf{c}}_0, \vec{\mathbf{c}}_1$ and $\vec{\mathbf{c}}_2$ are used to define the cross-boundary vector field of the 3×4 split patch along the corresponding network curve (see Figure 3). For this reason neighboring split patches share this degrees of freedom. The split-patch depends linearly on the $\vec{\mathbf{c}}$'s and thus these parameters can be used for the optimization process.

We modify only three of the scalar-valued degrees of freedom: $\Delta_0, \Delta_1, \Delta_2$. These scalars determine the position of the first control point $\mathbf{S}_i$ of an inner *cubic* boundary curve (see Figure 3), i.e.

$$\mathbf{S}_i = (1 - 2\Delta_i)\mathbf{V}_i + \Delta_i(\mathbf{B}_i + \mathbf{C}_i).$$

Unfortunately, the split-patch does not depend linearly on the Δ_i's. Thus we have to set these values heuristically. In the original paper of Shirman and Séquin, we find $\Delta_i = \frac{1}{3}$, which places the control point $\mathbf{S}_i$ on the centroid of $\mathbf{V}_i, \mathbf{B}_i$ and $\mathbf{C}_i$. This setting works fine in situations where the curve network imposes a nearly planar surface. In cases of strongly bending curves, the resulting surface tends to flatten in the center. We correct this by scaling Δ_i by the fraction $\|s\| \,/\, \|l\|$, where l is the line connecting $\mathbf{V}_i$ and the mid-point $\mathbf{M}_i$ of the opposite triangle edge, and s is the circular segment defined by the data $\mathbf{V}_i$, $\mathbf{F}_i(0.5)$ and the direction $\mathbf{S}_i - \mathbf{V}_i$. $\mathbf{F}_i$ denotes the curve opposite to the considered vertex $\mathbf{V}_i$ (see Figure 3).

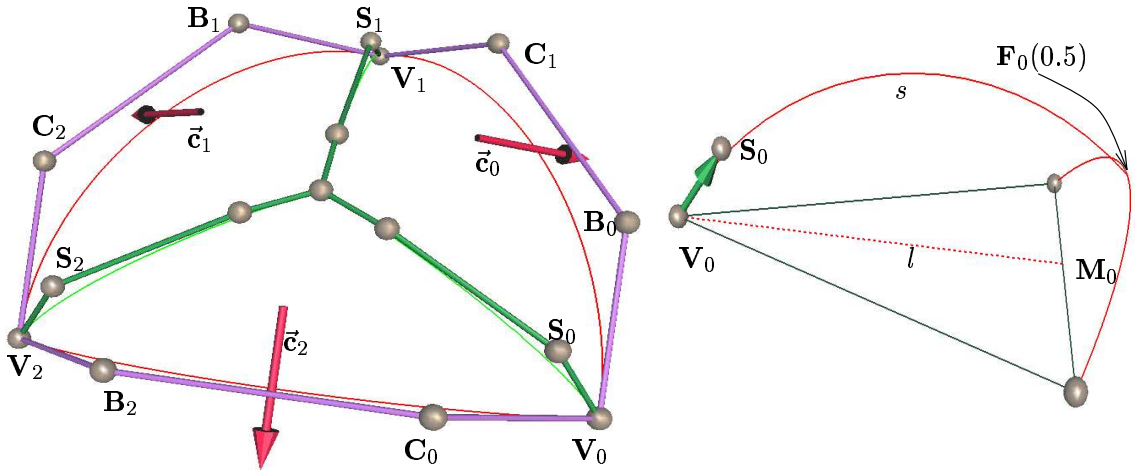

Figure 3: Degrees of freedom of the Shirman-Séquin split patch and placing of $\mathbf{S}_i$.

The remaining free parameters $\vec{\mathbf{c}}_i$ are specified by minimizing a quadratic functional. The following non-quadratic functionals are widely used in the context of surface fairing:

$$\sigma_1(\mathbf{S}) = \int (\kappa_1)^2 + (\kappa_2)^2 \; d\omega, \qquad , \qquad \sigma_2(\mathbf{S}) = \int \left(\frac{\partial \kappa_1}{\partial \hat{\mathbf{e}}_1}\right)^2 + \left(\frac{\partial \kappa_2}{\partial \hat{\mathbf{e}}_2}\right)^2 \; d\omega, \tag{1}$$

where κ_i denote the principle curvatures and $\hat{\mathbf{e}}_i$ the corresponding directions of principle curvature of the surface $\mathbf{S}$ (see [6]). The functional σ_2 is known as *Minimum Variation of Curvature (MVC)* functional. This functional has proved to yield surfaces with very good shape (see [5, 12]). The quadratic approximation to σ_1 is based on the Laplace-Beltrami operator for a surfaces $\mathbf{S}$ w.r.t the reference surface $\mathbf{S}_0$ (see [5]):

$$\Delta_{\mathbf{S}_0}(\mathbf{S}) = \frac{1}{\sqrt{g}} \left(\frac{\partial}{\partial u} \left(\sqrt{g} I_{\mathbf{S}_0}^{-1} \begin{pmatrix} \mathbf{S}_u \\ \mathbf{S}_v \end{pmatrix} \right) + \frac{\partial}{\partial v} \left(\sqrt{g} I_{\mathbf{S}_0}^{-1} \begin{pmatrix} \mathbf{S}_u \\ \mathbf{S}_v \end{pmatrix} \right) \right),$$

where $I_{\mathbf{S}_0}$ denotes the first fundamental form of $\mathbf{S}_0$ and $g = \det(I_{\mathbf{S}_0})$. Note that this operator is linear for the surface $\mathbf{S}$. Thus, the funtional

$$\sigma(\mathbf{S}) = \int \langle\, \Delta_{\mathbf{S}_0}(\mathbf{S}) \mid \Delta_{\mathbf{S}_0}(\mathbf{S}) \,\rangle \, d\omega_0,$$

where $d\omega_0$ denotes the area element of the reference surface $\mathbf{S}_0$.

The quadratic approximation to σ_2 yields a similar expression involving higher order derivatives and is referred to as *simplified MVC* functional (see [5]). We use both quadratic functionals as objective functional for our optimization. Since the representation of the 3×4 split patch is linear in the free parameters $\vec{\mathbf{c}}_i$ the optimization process is linear.

3.3 Simultaneous Optimization

After analysis, we find that the split-patch depends linearly on the partial derivatives used to represent the curve network. Thus we can combine the optimizations described in Section 3.1 and 3.2 to perform a simultaneous optimization. Not surprisingly, the results of this combined optimization step are often almost identical to fairing curve network and surface separately.

3.4 The G^2 Curve Network

In this section we describe the construction of a G^2 curve network. In the case of functional MNN such extensions to higher order curve networks already exist (see [8, 17]). Furthermore, Pottmann [17] introduces cross-boundary derivatives into the optimization of the curve network which leads to a more surface-like network description.

The basic idea of the representation of a G^2 curve network is very much the same as for the G^1 curve network. The tangents of the curve meeting at a vertex are computed in the same way as for the G^1 curve network.

Additionally, the second derivative of the curves have to agree with a common second fundamental form at the vertex. Therefore second order partial derivatives $s_{\hat{\mathbf{u}}\hat{\mathbf{u}}}, s_{\hat{\mathbf{u}}\hat{\mathbf{v}}}, s_{\hat{\mathbf{v}}\hat{\mathbf{v}}}$ are introduced as unknowns to the local coordinates $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. With respect to the base $\{\vec{\mathbf{u}}^* = \hat{\mathbf{u}} + s_{\hat{\mathbf{u}}}\hat{\mathbf{w}}, \vec{\mathbf{v}}^* = \hat{\mathbf{v}} + s_{\hat{\mathbf{v}}}\hat{\mathbf{w}}\}$ of the tangent plane (see Section 3.1), the second fundamental form is given by

$$\Pi = \frac{1}{\sqrt{1 + s_{\hat{\mathbf{u}}}^2 + s_{\hat{\mathbf{v}}}^2}} \begin{pmatrix} s_{\hat{\mathbf{u}}\hat{\mathbf{u}}} & s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} \\ s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} & s_{\hat{\mathbf{v}}\hat{\mathbf{v}}} \end{pmatrix}.$$

The second derivative vector $\vec{\mathbf{s}}^*$ of a curve have to satisfy the relation $\langle \vec{\mathbf{s}}^* | \hat{\mathbf{n}} \rangle = (\alpha, \beta)\, \Pi \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, where $\vec{\mathbf{t}} = \alpha\hat{\mathbf{u}} + \beta\hat{\mathbf{v}}$ is the parametric direction of the tangent $\vec{\mathbf{t}}^*$.

Introducing the parametric direction $\vec{\mathbf{s}} = \gamma\hat{\mathbf{u}} + \delta\hat{\mathbf{v}}$, where $\gamma = \langle\vec{\mathbf{s}}^*|\hat{\mathbf{u}}\rangle$ and $\delta = \langle\vec{\mathbf{s}}^*|\hat{\mathbf{v}}\rangle$, the second derivative vector $\vec{\mathbf{s}}^*$ is determined as:

$$\vec{\mathbf{s}}^* = \vec{\mathbf{s}} + \left(\gamma s_{\hat{\mathbf{u}}} + \delta s_{\hat{\mathbf{v}}} + (\alpha,\beta)\begin{pmatrix} s_{\hat{\mathbf{u}}\hat{\mathbf{u}}} & s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} \\ s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} & s_{\hat{\mathbf{v}}\hat{\mathbf{v}}} \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix}\right)\hat{\mathbf{w}}.$$

The parametric directions $\vec{\mathbf{s}}$ of the second derivative vectors of the curves can be determined as follows:
Cubic Curve: A cubic curve $\mathbf{C}$ interpolating the position and the parametric directions $\vec{\mathbf{t}}$ of the considered edge is defined; $\vec{\mathbf{s}}$ is set to the projection of $\mathbf{C}''(0)$ onto the local parameter plane (see Figure 4). The curves of the G^2 curve network are defined as quintic polynomials interpolating the corresponding position, as well as the tangents and second derivatives with the parametric directions $\vec{\mathbf{t}}$ and $\vec{\mathbf{s}}$ as described above.
In addition we define a cross-boundary vector-field $\vec{\mathbf{q}}_{ij}$ for each edge $\overline{\mathbf{V}_i\mathbf{V}_j}$ of the polyhedron. The parametric direction $\hat{\mathbf{d}}$ of $\vec{\mathbf{q}}_{ij}(0)$ at a vertex is set orthogonally to the parametric direction $\vec{\mathbf{t}}$ of the corresponding curve (see Figure 4).

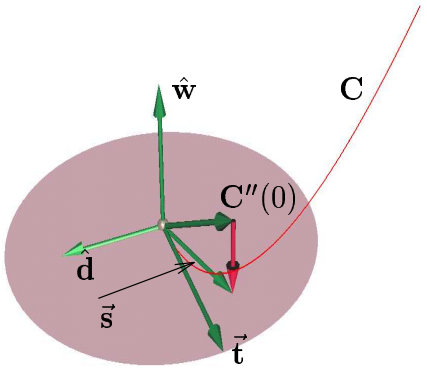

Figure 4: Second derivative and cross-boundary vector-field.

The first derivative $\vec{\mathbf{q}}'_{ij}(0)$ has to agree with the corresponding second fundamental form Π, i.e. for $\hat{\mathbf{d}} = \overline{\alpha}s_{\hat{\mathbf{u}}} + \overline{\beta}s_{\hat{\mathbf{v}}}$:

$$\langle\vec{\mathbf{q}}'_{ij}(0)|\hat{\mathbf{n}}\rangle = (\overline{\alpha},\overline{\beta})\ \Pi\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad\Longrightarrow\quad \langle\vec{\mathbf{q}}'_{ij}(0)|\hat{\mathbf{w}}\rangle = (\alpha,\beta)\begin{pmatrix} s_{\hat{\mathbf{u}}\hat{\mathbf{u}}} & s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} \\ s_{\hat{\mathbf{u}}\hat{\mathbf{v}}} & s_{\hat{\mathbf{v}}\hat{\mathbf{v}}} \end{pmatrix}\begin{pmatrix} \overline{\alpha} \\ \overline{\beta} \end{pmatrix}$$

The $\hat{\mathbf{u}}$- and $\hat{\mathbf{v}}$-components of $\vec{\mathbf{q}}'_{ij}(0)$ are set to zero. The cross-boundary vector-field is defined as cubic vector-field interpolating the corresponding values and first derivatives.

This representation of the G^2 curve network depends linearly on the unknown partial derivative $s_{\hat{\mathbf{u}}}, s_{\hat{\mathbf{v}}}, s_{\hat{\mathbf{u}}\hat{\mathbf{u}}}, s_{\hat{\mathbf{u}}\hat{\mathbf{v}}}, s_{\hat{\mathbf{v}}\hat{\mathbf{v}}}$ in the local coordinates. These unknowns are specified applying a minimization process using a quadratic functional as described in Section 3.1. Again, this process can be iterated as in case of the G^1 curve network.

Finally, we fit one triangular Bézier patch to each hole of the curve network interpolating also the cross-boundary vector-fields. Thus, for the cross-boundary vector-field $\mathbf{S}_v$ of a surface patch $\mathbf{S}$ along a network curve $\mathbf{C}_{ij}$ we have: $\mathbf{S}_v(t) = a_{ij}(t)\mathbf{C}'_{ij}(t) + b_{ij}(t)\vec{\mathbf{q}}_{ij}(t)$, for some scalar weight polynomials a_{ij} and b_{ij}. To meet the twist-constraints at each vertex, we need two scalar-valued degrees of freedom. This plus the interpolation constraints at the end-points gives the restrictions: $deg(a), deg(b) \geq 1, deg(a) + deg(b) \geq 4$. We choose $deg(a) = 1, deg(b) = 3$ which sets the degree of $\mathbf{S}$ to 7.

Up until this point we have not used any optimizing techniques to determine these Bézier patches. Rather we use a simple method to set the leftover degrees of freedom. Even without any surface fairing, the method based on the optimized G^2 curve network yields good results.

4 Examples

Several input polyhedra are used to examine the methods described in the prior sections. The quality of the different schemes is illustrated with color or gray-scale curvature plots displaying the *Gaussian*

curvature of the surface. For color curvature plots the color at a surface point is obtained by linearly mapping a predefined interval of curvature values to a range of colors going from red (minimal curvature) to blue (maximal curvature). Generally, surfaces with well distributed curvature, i.e. surfaces of good quality, exhibit smooth color distributions, whereas wrinkles and other artifacts result in strongly varying colors. The gray-scale images are simply obtained by scaling the color values using a standard method.

The first input polyhedron is a regular octahedron. We find that the original Shirman-Séquin scheme produces flat Bézier patches yielding a very non-uniform curvature distribution (see Figure 5 b)). The results of the reconstruction based on the optimized G^1- and G^2 curve network based schemes exhibit much better curvature distribution (see Figure 5 c) and d)). The parametric directions $\vec{\mathbf{t}}$ of the G^1 curve network have been determined using the DeBoor-Höllig-Sabin method and the plane curve method. The split patches are optimized using the simplified MVC functional. The G^2 curve network is constructed utilizing the circle segment method to determine the length of the parametric direction $\vec{\mathbf{t}}$ and the cubic curve method to determine the parametric direction $\vec{\mathbf{s}}$ of the second derivative.

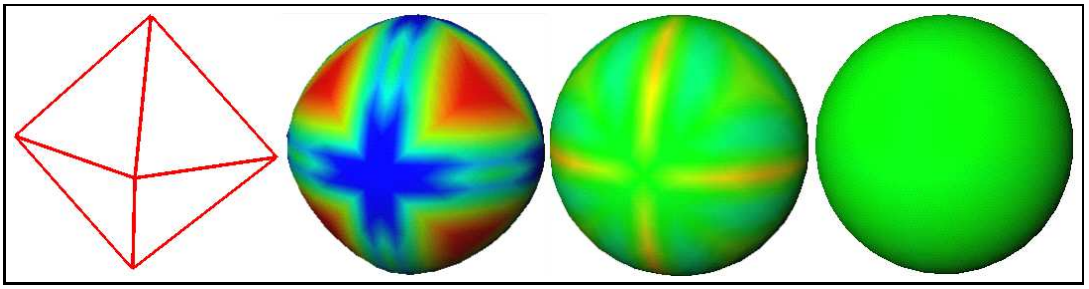

Figure 5: Reconstructing a sphere: a) the octahedron; b) Shirman-Séquin's method; c) the optimized G^1 curve network and surface and d) the optimized G^2 curve network scheme.

Our next example consist of the so-called *Washington-test-torus*. For comparison we reproduced the torus with the original Shirman-Séquin scheme (Figure 6 b)). The plane curve- and the circle segment method are used to determine the parametric directions $\vec{\mathbf{t}}$ for the G^1 curve network. The split patches are optimized with the Laplace-Beltrami functional. The reconstructed surface in Figure 6 c), already exhibits a very improved curvature distribution. The reconstruction based upon the optimized G^2 curve network yields a surface with even smoother curvature distribution. In this case the plane curve- and the circle segment method are used to determine the parametric directions $\vec{\mathbf{t}}$. The parametric directions $\vec{\mathbf{s}}$ of the second derivative are determined with the cubic curve method.

Finally we give a very difficult example: A part of a car is designed with a commercial CAD system. The part is further processed by using a local deformation. This deformation is simulated on a regular grid of points. To make the result of the simulation again available for the CAD system, we use some data reduction to get rid of redundancies and reconstruct the surface. The input polyhedron contains some long and thin triangles making the reconstruction process numerically very difficult to handle. First we use Shirman-Séquin's method to reconstruct the car part (see Figure 7 b)). The Gaussian curvature plot reveals severe problems in areas where the surface has relatively small curvature radii. Most of these problems where removed using the G^1- and G^2 curve network based reconstruction schemes (see Figure 7 c) and d), respectively). For the G^1 curve network the parametric direction $\vec{\mathbf{t}}$ is determined by the plane curve- and the circle segment method. The split patches are optimized using the Laplace-Beltrami functional. The methods used to determine $\vec{\mathbf{t}}$ and $\vec{\mathbf{s}}$ for the G^2 curve network are: the plane curve- with the circle segment method and the cubic curve method for $\vec{\mathbf{t}}$ and $\vec{\mathbf{s}}$, respectively.

5 Concluding Remarks

An algorithm has been presented for constructing a smooth surface interpolating the vertices and the topology of an input polyhedron with triangular faces. The first variant of the algorithm generates an optimized cubic G^1 curve network and fits optimized split patches consisting of three degree four triangular Bézier patches to the holes of the curve network. The optimization can be realized either in two steps (for the curve network and the split patches separately) or in one simultaneous step determining all free parameters (for the curve network and the split patches). The second variant of the algorithm generates a G^2 curve network which is extended to a smooth surface with only one degree seven triangular Bézier patch per facet. The resulting surfaces are smooth (tangent plane continuous) and exhibit very smooth distribution of curvature.

Acknowledgments

The authors would like to acknowledge the comments and detailed suggestions by two anonymous referees.

References

[1] CHIYOKURA, H. Localized surface interpolation method for irregular meshes. *Advanced Computer Graphics* (1986), 3–19.

[2] DEROSE, T., AND MANN, S. An approximately G^1 cubic surface interpolant. In *Math. Methods in CAGD II* (1992), T. Lyche and L. Schumaker, Eds., Academic Press, pp. 185–196.

[3] DIERCKX, P. *Curve and surface fitting with splines.* Clarendon Press, 1993.

[4] FARIN, G. A construction for visual continuity of polynomial surface patches. *Computer Graphics and Image Processing20* (1982), 272–282.

[5] GREINER, G. Variational design and fairing. In *Proc. EUROGRAPHICS '94* (1994), vol. 13, Eurographics, Blackwell Publishers.

[6] KLINGENBERG, W. *A course in differential geometry.* Springer-Verlag, Berlin-Heidelberg, 1978.

[7] KOLB, A., POTTMANN, H., AND SEIDEL, H.-P. Surface reconstruction based upon minimum norm networks. In *Math. Methods for Curves and Surfaces* (Ulvik, Norway, 1995), M. Dæhlen, T. Lyche, and L. Schumaker, Eds., Vanderbuilt University Press.

[8] KOLB, A., AND SEIDEL, H.-P. Interpolating scattered data with C^2 surfaces. *Computer Aided Design27:4* (1995), 277–282.

[9] LOOP, C. A G^1 triangular spline surface of arbitrary topological type. *Computer-Aided Geom. Design11* (1994), 303–330.

[10] LOUNSBERY, M., MANN, S., AND DEROSE, T. Parametric surface interpolation. *IEEE Computer Graphics & Appl.12* (1992), 45–52.

[11] MANN, S. *Surface approximation using geometric Hermite patches.* PhD thesis, University of Washington, 1992.

[12] MORETON, H., AND SÉQUIN, C. Functional optimization for fair surface design. In *Proc. SIGGRAPH* (1992), vol. 26, pp. 167–176.

[13] NIELSON, G. A method for interpolating scattered data based upon a minimum norm network. *Math. Comp.40* (1983), 253–271.

[14] NIELSON, G. Interactive surface design using triangular network splines. In *Proceedings 3rd Int. Conf. on Engineering Graphics and Descriptive Geometry* (Vienna, 1988), vol. 2, pp. 70–77.

[15] PETERS, J. Local smooth surface interpolation: a classification. *Computer-Aided Geom. Design7* (1990), 191–195.

[16] PETERS, J. Smooth interpolation of a mesh of curves. *Constr. Approx.7* (1991), 221–246.

[17] POTTMANN, H. Scattered data interpolation based upon generalized minimum norm networks. *Constr. Approx.7* (1991), 247–256.

[18] SHIRMAN, L., AND SÉQUIN, C. Local surface interpolation with Bézier patches. *Computer-Aided Geom. Design4* (1987), 279–295.

[19] VELTKAMP, R. *Closed object boundaries from scattered points.* PhD thesis, Erasmus-University Rotterdam, The Netherlands, 1992.