Fifth Hungarian Conference on Computer Graphics and Geometry, Budapest, 2010

Constrained Fairing of Freeform Surfaces

Péter Salvi [1], Tamás Várady [2]

[1] Loránd Eötvös Science University, Budapest
[2] Geomagic Inc., Budapest

Abstract

New algorithms are introduced to create fair B-spline surfaces that also satisfy continuity constraints, inherited from neighboring surfaces. The proposed procedure comprises a stepwise method to assure approximate curvature continuity and adapts various existing fairing methods. A new fairing algorithm based on curvature approximation is also presented. The discussion is not restricted to four-sided patches, but an extension to handle n-sided surfaces is also provided.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

The goal of fairing is to create visually pleasing curves and surfaces in Computer Aided Geometric Design. This is also crucial in Digital Shape Reconstruction[13], where digital models are created from millions of data points, while retaining the geometric features of scanned objects. There is no unique algebraic expression to define aesthetic requirements; however, there is a general agreement in the styling industries that even curvature distribution with large, monotone curvature areas is the most important to achieve high-quality of surfaces. For example, take reflection lines that are frequently used in the automotive industries to indicate curvature imperfections.

Approaches to surface fairing can be divided into two categories. There are *variational methods*, that simultaneously minimize least-squares distances and various smoothness functionals during surface fitting, and there are *post-processing methods* that apply changes on already existing geometries. Both methods minimize various fairness measures, and control the extent of deviations from the original data points. In many cases though, a higher priority is assigned to the fairness of the surface approximation.

1.1. Problem

There are several, traditional methods that work quite well when primary surfaces are fitted independently of their environment; however, when a complete object composed of many connected surfaces needs to be created, not only the smoothing of the individual surfaces is needed, but the continuity between the adjacent surfaces must be taken into account. Fairing dependent connection surfaces, such as fillets or corner patches, thus poses a new problem. Several methods have been published that deal with this problem using the variational method, see for example Lai et al.[8] or Hsu et al.[6].

In this paper we address the problem from the post-processing perspective. Suppose that we have a model consisting of several surfaces with dependency relations as it has been defined in the functional decomposition paradigm[13]. We would like to perform fairing in a hierarchical order, first fairing the primary surfaces, then fillets, and finally corner patches. In order to achieve this, we need algorithms that create fair surfaces while assuring smooth connections to the adjacent surface elements. For fillets, smooth connection to two primary surfaces must be satisfied; for corner patches, smooth connection to n surrounding surface elements is needed. Our goal is to deal primarily with the latter problem of constrained fairing, which includes a solution for fillets, as well.

1.2. Goals

We can break this problem into two separate subproblems.

The first one is to assure continuous connections. In CAGD, continuity constraints between surfaces are typically

relaxed to *geometric continuity* instead of matching the parametric derivatives. G^1 continuity enforces common tangent planes along the shared boundaries, G^2 continuity enforces common surface curvatures. We want to tweak a surface S to match the fixed *master* surfaces M_i by *numerical* (tolerance driven) G^1 and G^2 constraints.

The second one is to perform the actual fairing. We need algorithms that smooth the surface S and retains the continuity constraints to the master surfaces. This may be as simple as doing local fairing in the "inside" region of S, or may mean special fairness measures that incorporate the constraints into the fairing process. We will evaluate different approaches later.

A combination of the above two algorithms yields a solution to our main problem.

1.3. Outline

The structure of the paper is the following. In Section 2 we briefly review some papers that are important for this work. In Section 3 a solution to the simplest configuration of four-sided corner patches is presented using an alternating sequence of constraining and fairing. The extension to n-sided patches is given in Section 4. The results are illustrated using a few examples in Section 5, and a few concluding remarks are added at the end of the paper.

2. Previous Work

There are several papers in the literature on continuity constraints and surface fairing, but a solution to constrained fairing of multiple surfaces in the post-processing context is not known to the authors. Space limitations prevent us to give a full review of all known approaches; instead we collected a few contributions that have been influential to our current project.

Concerning curvature continuity constraints — Pegna and Wolter[9] proved the *Linkage Curve Theorem,* which gives a necessary and sufficient condition for G^2 continuity between two surfaces that share common tangent planes. The theorem states (as rephrased in Hermann et al.[5], who also extended the theorem to G^n continuity) the following:

> Two surfaces tangent along a C^1-smooth linkage curve are curvature continuous if and only if at every point of the linkage curve, their normal curvature agrees for an arbitrary direction other than the tangent to the linkage curve.

This is the basis of our G^2 algorithm in Section 3.1.

Concerning curvature approximation — a paper by Greiner[3] gives a quadratic approximation of surface curvature using a reference surface (e.g. a least-square fit of the data points). The idea here is to compute part of the Hessian matrix from the reference surface, such that the computation still dependent on the real surface is only quadratic. Then the mean and Gauss curvatures can be computed from the Hessian. Greiner also introduces a data-dependent quadratic fairness functional, along with an algorithm minimizing this approximated curvature.

We use different fairing algorithms in our tests. These will be analyzed in more details in Section 3.2.

Knot-Removal and Reinsertion (KRR), proposed originally by Farin and Sapidis[2], is one of the simplest and most widely used local, control-point based fairing algorithm. It provides fair curves by removing and reinserting the *most offending knots*, thus the C^3 jumps for a cubic B-spline are reduced, which yields a nicer curve. The algorithm has a couple of variations; Hahmann[4] extended the method to surface fairing.

Another fairing method was proposed by Salvi et al.[11], that approximates a smoothed *target curvature*. This is an efficient and highly flexible algorithm; however, it includes a final fitting step which may harm locality to some extent.

Finally, for n-sided corner patches (see Section 4), we cannot use bi-parametric surface algorithms. Instead, we apply a mesh based fairing algorithm described by Kobbelt et al.[7], where the n-sided surface region is approximated by a mesh and discrete fairing is applied. A local quadratic approximation is used to compute the second-order partial derivatives and minimize the thin plate energy. Continuity constraints can be preserved at sampled data points around the boundary of the n-sided region.

3. Four-sided Patches

A B-spline surface $S(u,v)$ is defined by means of its control points P_{ij} ($i \in [0\dots n]$, $j \in [0\dots m]$) and its knot vectors U and V:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} N_i^{U,p}(u) N_j^{V,q}(v),$$

where p and q are the degrees of the surface in the u and v directions, respectively. The four boundaries of the surface are defined by the outermost control points — P_{0j}, P_{nj}, P_{i0} and P_{im}, respectively ($i \in [0\dots n]$, $j \in [0\dots m]$). In this context we call these together as the *positional frame*. Let us assume hereinafter that the outermost control points are fixed. The tangent planes at the points of the boundaries are indirectly determined by the first cross-boundary derivatives, i.e. by the inner control points P_{1j}, $P_{(n-1)j}$, P_{i1} and $P_{i(m-1)}$, respectively ($i \in [1\dots n-1]$, $j \in [1\dots m-1]$). We call these control points together the *tangential frame*. Finally, assume that control points of the positional and tangential frames are fixed. Then the surface curvatures at the points of the boundaries are indirectly determined by the second cross-boundary derivatives, i.e. by the inner control points P_{2j}, $P_{(n-2)j}$, P_{i2} and $P_{i(m-2)}$, respectively ($i \in [2\dots n-2]$, $j \in [2\dots m-2]$). We call these control points together the *curvature frame*.

It is also well-known, that when we enforce G^1 continuity independently for the individual boundaries, a so-called twist incompatibility condition must be satisfied for the mixed partial derivatives, which are indirectly determined by the twist control points P_{11}, $P_{1(m-1)}$, $P_{(n-1)1}$ and $P_{(n-1)(m-1)}$. After we enforce G^2 continuity for the individual boundaries, a similar compatibility condition must be satisfied to tweak the inner twist control points P_{22}, $P_{2(m-2)}$, $P_{(n-2)2}$ and $P_{(n-2)(m-2)}$.

Having said all these, the algorithm to set continuity constraints proceeds in the following way:

1. Insert some knots into the surface, if it has too few control points.
2. Fair the surface, while retaining C^0 continuity for each boundary, i.e. only control points within the positional frame are used for fairing.
3. Fix the positional frame and enforce G^1 continuity for each boundary.
4. Set the twist control points compatible, and repeat the G^1 computation.
5. Fair the surface, while retaining G^1 continuity, i.e. only control points within the tangential frame are used for fairing.
6. Fix the tangential frame and enforce G^2 continuity for each boundary.
7. Set the inner twist control points compatible, and repeat the G^2 computation.
8. Fair the surface, while retaining G^2 continuity, i.e. only control points within the curvature frame are used for fairing.

To sum it up, the sequence is always (i) constrain, (ii) set compatibility and (iii) apply fairing, until G^2 is achieved.

3.1. Three-Frame Method for Numerical G^1/G^2

We will treat the algorithm on a per side basis. The twists, however, need special attention, and they will be dealt with at the end of this section.

3.1.1. G^1 Continuity

There are two B-spline surfaces, M (master) and S (slave), that are (without loss of generality) joined along the $u = u_{\min}$ parameter line with C^0 continuity. We want to modify the corresponding side of the tangential frame, i.e. the second control row of S, such that the two surfaces will have numerical G^1 continuity. We also take sampled parameters v_k ($k \in [1 \dots K]$) along the border, not including the corner points. The normal vectors at these (u, v_k) points of the master surface are denoted by $\underline{n}_k$. If we now add displacement vectors $\underline{w}_j$ to the control points P_{1j} ($j \in [1 \dots m-1]$), they will modify the original tangents $\underline{e}_k$ in the following way:

$$\hat{\underline{e}}_k = \underline{e}_k + \dot{N}_1^{U,p}(u) \sum_{j=1}^{m-1} N_j^{V,q}(v_k)\underline{w}_j \quad (1)$$

$$= \underline{e}_k + \sum_{j=1}^{m-1} c_{k,j}\underline{w}_j. \quad (2)$$

Since we would like to avoid irrelevant control point movements, we minimize the deviation only in the surface normal direction. This leaves us with

$$\hat{\underline{e}}_k = \underline{e}_k - \underline{n}_k(\underline{e}_k\underline{n}_k). \quad (3)$$

Subtracting $\underline{e}_k$ from (2) and (3) gives the linear equation system $A\underline{x} = \underline{b}$, where

$$A = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1(m-1)} \\ c_{21} & c_{22} & \cdots & c_{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K1} & c_{K2} & \cdots & c_{K(m-1)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \underline{w}_1 \\ \underline{w}_2 \\ \vdots \\ \underline{w}_{m-1} \end{bmatrix}, \; \underline{b} = \begin{bmatrix} -\underline{n}_1(\underline{e}_1\underline{n}_1) \\ -\underline{n}_2(\underline{e}_2\underline{n}_2) \\ \vdots \\ -\underline{n}_K(\underline{e}_K\underline{n}_K) \end{bmatrix}.$$

We have K equations and $m-1$ unknowns. Solving the least-square equation $(A\underline{x} - \underline{b})^2 = 0$ leads to a linear system of $(A^TA)\underline{x} = A^T\underline{b}$.

3.1.2. G^2 Continuity

We have the same assumptions as in the G^1 case, but now we also have a numerical G^1 connection. Take K parameter points from the v domain: v_k, $1 \le k \le K$. For every v_k, calculate the normal curvature κ_k^M of M at $(u_{\min}, v_k)$ in the u direction. If we modify S such that its normal curvature in the u direction is the same, we will have G^2 continuity in $(u_{\min}, v_k)$, because of the Linkage Curve Theorem (see Section 2).

The normal curvature of a surface at (u, v) in some $\underline{d}$ direction can be calculated as:

$$\kappa(\lambda) = \frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2},$$

where E, F, G and L, M, N are the coefficients of the first and second fundamental form, respectively; $\lambda = \frac{d_v}{d_u}$ and $\underline{d} = d_u\underline{S}_u + d_v\underline{S}_v$. In the special cases where $\underline{d}$ is the u or v parametric direction, this is simplified into L/E and N/G, respectively.

Minimization

Instead of directly optimizing for surface curvature, we use the curvature of the surface curve $C_k(u) = S(u, v_k)$, which results in much simpler equations[12]. We know from

Meusnier's theorem[1] that at a given point, the curvature κ_C of a surface curve C and the normal curvature κ of a surface S in the tangent direction of C have the following relationship:

$$\kappa = \kappa_C \cos\theta = \frac{\left\|\underline{C}' \times \underline{C}''\right\|}{\left\|\underline{C}'\right\|^3} \cos\theta = \frac{\langle \underline{C}'', \underline{n} \rangle}{\left\|\underline{C}'\right\|^2},$$

where $\underline{n} = \frac{\underline{S}_u \times \underline{S}_v}{\|\underline{S}_u \times \underline{S}_v\|}$ is the surface normal and θ is the angle between $\underline{n}$ and the curve normal $(\underline{C}' \times \underline{C}'') \times \underline{C}'$.

Consequently, we have to solve equations of the form

$$\frac{\langle C_k''(u_{\min}), \underline{n}_k \rangle}{\left\|C_k'(u_{\min})\right\|^2} = \kappa_k^M. \tag{4}$$

Since

$$\begin{aligned} C_k(u) &= S(u, v_k) = \sum_{i=0}^{n} N_i^{U,p}(u) \left(\sum_{j=0}^{m} N_j^{V,q}(v_k) P_{ij} \right) \\ &= \sum_{i=0}^{n} N_i^{U,p}(u) \hat{P}_i, \end{aligned}$$

the first and second derivatives at the end are

$$\begin{aligned} C_k'(u_{\min}) &= \frac{p}{u_{p+1} - u_1}(\hat{P}_1 - \hat{P}_0), \\ C_k''(u_{\min}) &= \frac{p-1}{u_{p+1} - u_2} \left(\frac{p}{u_{p+2} - u_2}(\hat{P}_2 - \hat{P}_1) \right. \\ &\quad \left. - \frac{p}{u_{p+1} - u_1}(\hat{P}_1 - \hat{P}_0) \right), \end{aligned}$$

see Reference[10] (assuming that $u_0 = u_1 = \cdots = u_p = u_{\min}$).

If we want to modify the P_{2j} control point by a vector $\underline{d}_j$, we can reformulate Eq. 4 as

$$\begin{aligned} \Delta\kappa_k \left\|C_k'(u_{\min})\right\|^2 \frac{(u_{p+1} - u_2)(u_{p+2} - u_2)}{p(p-1)} \\ = \sum_{j=2}^{m-2} N_j^{V,q}(v_k) \left\langle \underline{d}_j, \underline{n}_k \right\rangle, \end{aligned} \tag{5}$$

where $\Delta\kappa_k = \kappa_k^M - \kappa_k^S$. Note that the second half of C_k'' is eliminated in the scalar product by the (perpendicular) surface normal.

We propose two different solutions for this equation system. In the first one, in order to avoid irrevelant control point movements, the P_{2j} points are only allowed to move in an *outward* direction ($\underline{o}_j$ for P_{2j}). Alternatively, we can require that the sum of the squared deviations should be minimal.

Solution by Fixed Directions

An easy and intuitive choice for outward directions is to get the cross product of the difference of the neighboring control points (Fig. 1), i.e. $\underline{o}_j = (P_{3j} - P_{1j}) \times (P_{2(j+1)} - P_{2(j-1)})$. If we define the deviation vectors $\underline{d}_j$ as $\chi_j \underline{o}_j$ and introduce

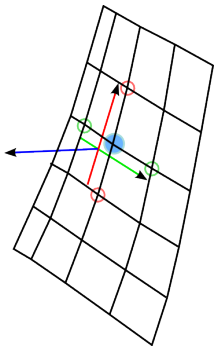


Figure 1: Outward direction

the constants α_{kj} and β_k, Eq. 5 can be rewritten as $\beta_k = \sum_{j=2}^{m-2} \alpha_{kj} \chi_j$.

Now we can create the overdefined equation system $A\underline{x} = \underline{b}$:

$$A = \begin{bmatrix} \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1(m-2)} \\ \alpha_{22} & \alpha_{23} & \cdots & \alpha_{2(m-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{K2} & \alpha_{K3} & \cdots & \alpha_{K(m-2)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \chi_2 \\ \chi_3 \\ \vdots \\ \chi_{m-2} \end{bmatrix}, \underline{b} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix}.$$

Solving the $(A^T A)\underline{x} = A^T \underline{b}$ equation results in a least-squares approximation, as earlier.

Solution by Minimal Deviation

We can also solve the equations while minimizing the squared deviation of the $\hat{P}_2$ control point of C_k, by requiring that it should change only in the $\underline{n}_k$ direction. This means that Eq. 5 becomes $\beta_k \underline{n}_k = \sum_{j=2}^{m-2} N_j^{V,q}(v_k) \underline{d}_j = \sum_{j=2}^{m-2} \gamma_{kj} \underline{d}_j$.

The equation system is now $A\underline{x} = \underline{b}$, where

$$A = \begin{bmatrix} \gamma_{12} & \gamma_{13} & \cdots & \gamma_{1(m-2)} \\ \gamma_{22} & \gamma_{23} & \cdots & \gamma_{2(m-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{K2} & \gamma_{K3} & \cdots & \gamma_{K(m-2)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \underline{d}_2 \\ \underline{d}_3 \\ \vdots \\ \underline{d}_{m-2} \end{bmatrix}, \underline{b} = \begin{bmatrix} \beta_1 \underline{n}_1 \\ \beta_2 \underline{n}_2 \\ \vdots \\ \beta_K \underline{n}_K \end{bmatrix}.$$

As before, $(A^TA)\underline{x} = A^T\underline{b}$ gives a least-squares approximation.

3.1.3. Twists

Every twist (or inner twist) control point of a frame has two values coming from the independent side constraints. In order to get a valid B-spline, we need a single twist control point, computed as halving of the two independent twist control points. After this, we repeat the continuity enhancement algorithm, now constraining only the $j \in [2 \dots m-2]$ control points for G^1 and the $j \in [3 \dots m-3]$ control points for G^2, in order to obtain the best positions in accordance with the new twist values. The use of halving points is a dubious choice and optimizing the twist control points is subject of further research.

3.2. Fairing Algorithms

In this section we will look at three different algorithms suitable for our purpose. The minimal condition is that fairing should not destroy the continuity already achieved.

3.2.1. Knot Removal and Reinsertion (KRR)

This fairing method operates on the control points and we choose the one where the C^3 jump is the greatest. We perform this operation in an iterative manner several times. This retains continuity in a trivial way. On the backside, KRR is primarily suited for curve fairing. We can choose between fairing only in the u or v parametric direction — or average the two values. This simple method gives quite nice results.

3.2.2. Fairing Based on Target Curvature

We have much more flexibility in this algorithm, as the use of a target curvature enables us to define curvature continuity constraints, which is particularly valuable in our present task. However, there are some problems. The last fitting phase may harm the precision of the continuity. In this case, we need to insert another continuity enhancement step. Another drawback is that the algorithm relies on fairing the isocurves of the surface. This may cause artefacts in complex saddle surfaces.

3.2.3. Fairing Based on Curvature Approximation

We will use the curvature approximation outlined in Section 2 to enhance the previous algorithms. Given a reference surface R and a twice differentiable scalar function $\tilde{h}$ defined on it, Greiner[3] shows that the Hessian matrix of $h = \tilde{h} \circ R$ is

$$\mathrm{Hess}_R(h) = \left(\sum_l g^{kl} (\partial_j \partial_l h - \sum_i \partial_i h \Gamma^i_{jl}) \right)_{kj},$$

where ∂_i is the partial derivative by the i^{th} argument, $\Gamma^i_{jl} = \sum_m g^{im} \langle \partial_j \partial_l R, \partial_m R \rangle$ and (g^{ij}) is the inverse of the first fundamental form of R (every index can take the values 1 and 2). Note that both Γ and g can be computed beforehand. The paper also shows that if we use the coordinate functions R_c ($c \in [1 \dots 3]$) as h, we have

$$\mathrm{Hess}_R(R_c) = \frac{1}{EG-F^2} \begin{bmatrix} G & -F \\ -F & E \end{bmatrix} \begin{bmatrix} L & M \\ M & N \end{bmatrix} n_c,$$

where $\underline{n}$ is the surface normal. This has several nice properties, for example it is easy to see that

$$\sum_c \mathrm{trace}\,(\mathrm{Hess}_R(R_c))^2 = (\kappa_1^R + \kappa_2^R)^2,$$
$$\sum_c \det(\mathrm{Hess}_R(R_c)) = \kappa_1^R \cdot \kappa_2^R.$$

The claim is that using S_c instead of R_c will result in good approximations of the curvatures of S:

$$\sum_c \mathrm{trace}\,(\mathrm{Hess}_R(S_c))^2 \approx (\kappa_1^S + \kappa_2^S)^2,$$
$$\sum_c \det(\mathrm{Hess}_R(S_c)) \approx \kappa_1^S \cdot \kappa_2^S.$$

The reader can consult the original paper[3] for more details.

In our case, we already have a very good reference surface — the original surface itself. We would like to do something similar to the algorithm in the previous section, i.e. set up a target curvature and find a surface with similar curvature. Our general scheme would be as follows:

1. Sample the original surface at intervals.
2. Compute Γ and g in these positions.
3. Set up a target curvature.
4. Minimize the deviation from the target curvature.

We can write up an equation system that depends linearly on the control points of S, so it will be easy to minimize. One option is to create a target mean curvature by smoothing the traces of Hessian matrices in the sampled points. Another alternative is to average every element of the Hessian matrices over the sampled points. Since the computation of the Hessian matrix from the control points is linear, these lead to overdefined linear equation systems, that can be solved in least-squares sense. An example is shown in Fig. 2.

As for the continuity restrictions, we can fix the outer frames as in the KRR algorithm. The advantage of this method is that it is independent of the parametric directions. However, we have to minimize a fairly large equation system, which makes its computational cost quite high.

4. n-sided Corner Patches

There are corner patches with three or more than four (usually five) connecting surfaces. The simplest representation of these is based on the so-called central split, where n quadrilateral surfaces are stitched together, see Fig. 5 for a five-sided example.

The main difficulty here is that the corner patch consists of more than one surface, so we have to ensure continuity not

(a) Before fairing

(b) After fairing

Figure 2: Isophotes of a car body panel faired by the curvature approximation method

only along the boundaries, but along the subdividing curves between the quadrilaterals as well. Moreover, for n-sided regions the previous fairing methods that were applied for bi-parametric surfaces cannot be used.

4.1. Extension of the Algorithm

In order to cope with these difficulties, we modify the procedure presented in Section 3:

1. Global fairing of the quadrilaterals (retaining C^0 or G^1 continuity on the perimeter).
2. Enforce G^1 continuity on the perimeter and between the quadrilaterals.
3. Local fairing of each quadrilateral, retaining G^1 continuity.
4. Enforce G^2 continuity on the perimeter and between the quadrilaterals.
5. Local fairing of each quadrilateral, retaining G^2 continuity.

The first step aims at creating a good base for the further operations. It can be omitted if the original surface has adequate quality. Unlike in the four-sided case, the first step here needs special attention, since multiple surfaces need to be processed simultaneously. A mesh-based fairing seems to be a natural approach.

4.2. Mesh-based Fairing

In order to avoid the parameterization problems, we discretize the corner patch, and create a triangle mesh as shown in Fig. 6. The actual fairing is done by the method suggested by Kobbelt[7], inheriting G^1 continuity at the boundaries. Finally, we refit the surfaces using the points of the faired mesh. Although this algorithm, due to precision loss, may be inferior to the others, it creates generally a good shape and is easy to use regardless of the number of quadrilaterals involved.

5. Examples

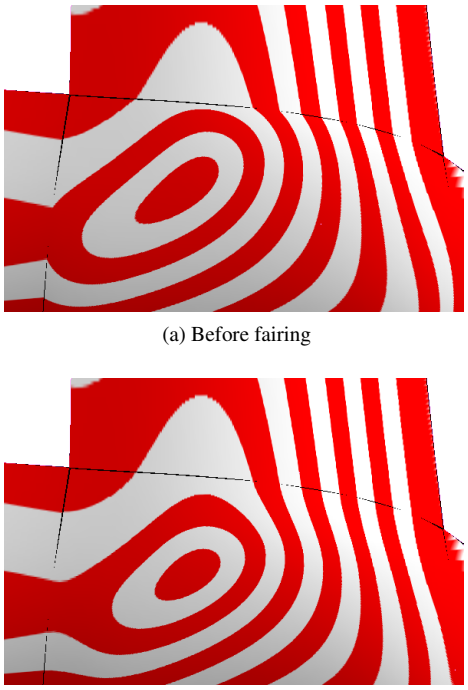
Figures 3 and 4 both show the effect of constrained fairing for four-sided corner patches. In these cases, the central surface is relatively simple; the original (a) has reasonable isophotes, but the isophote strips break when they reach the boundaries. After constrained fairing (b) the images show that numerical G^2 continuity has been achieved and fairing also nicely affected the interior without changing the master surfaces.

We have a different case in Fig. 7. Here the original corner patch (a) already had numerical G^2 that needed only very minor changes. On the other hand, fairing (b) increased the overall surface quality very much.

Conclusion and Future Work

A new approach that combines numerical G^2 connections between free-form surfaces with methods to create fair shapes was presented. Its primary application is to perfect connected surface elements produced in Digital Shape Reconstruction. The proposed algorithms have worked well for several models.

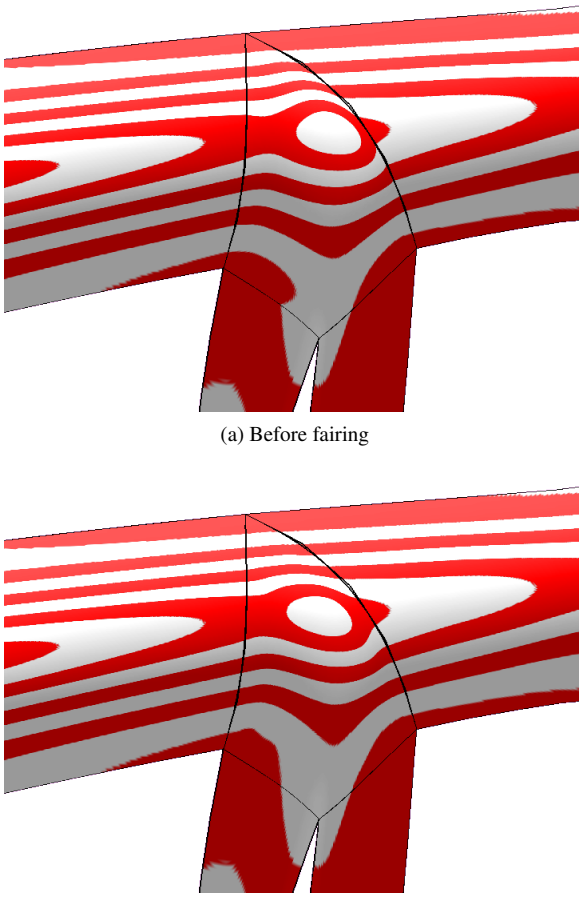
There is still room for improvements. For example, the algorithm that assures continuity should make a better use of the twist control points. We believe that better twist locations may significantly influence the quality of the modified surfaces. The averaging strategy of the KRR method applied for the u and v parametric directions should also be enhanced. The n-sided corner-patches should also be improved by applying such discrete fairing methods that guarantee discrete G^2 connections to the master surfaces. In fact, the discrete fairing step is considered only as a temporary solution, and alternative fairing techniques for n-sided corner patches are subject of our current investigations.



(a) Before fairing

(b) After fairing

Figure 3: Fairing an X-node



(a) Before fairing

(b) After fairing

Figure 4: Fairing another X-node

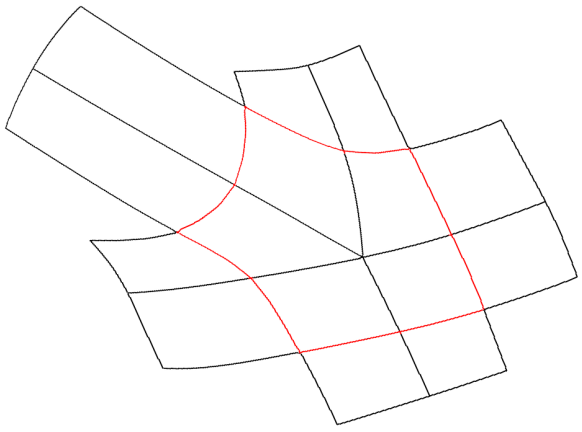


Figure 5: Five-sided corner patch consisting of five quadrilateral surfaces

References

1. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*. Academic Press, 5th Edition, 2002.
2. G. Farin, G. Rein, N. Sapidis, A. J. Worsey, *Fairing Cubic B-Spline Curves*, Computer Aided Geometric Design, Vol. 4, pp. 91–103, 1987.
3. G. Greiner, *Curvature Approximation with Application to Surface Modeling*, Teubner, pp. 241–252, 1996.
4. S. Hahmann, S. Konz, *Knot-Removal Surface Fairing Using Search Strategies*, Computer Aided Design, Vol. 30, pp. 131–138, 1998.
5. T. Hermann, G. Lukács, F-E. Wolter, *Geometrical Criteria on the higher order smoothness of composite surfaces*, Computer Aided Geometric Design, Vol. 16, pp. 907–911, 1999.
6. K.-L. Hsu, D.-M. Tsay, *Corner Blending of Free-Form N-Sided Holes*, IEEE Computer Graphics and Applications, Vol. 18, pp. 72–78, 1998.
7. L. P. Kobbelt, *Discrete Fairing and Variational Subdivision for Freeform Surface Design*, The Visual Computer, Vol. 16, pp. 142–158, 2000.
8. J.-Y. Lai, W.-D. Ueng, G^2 *Continuity for Multiple*

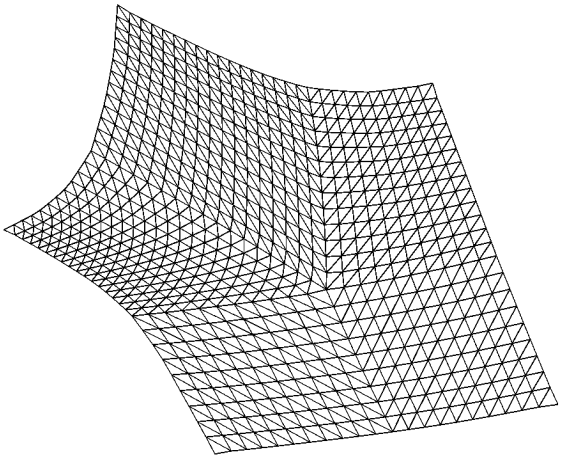
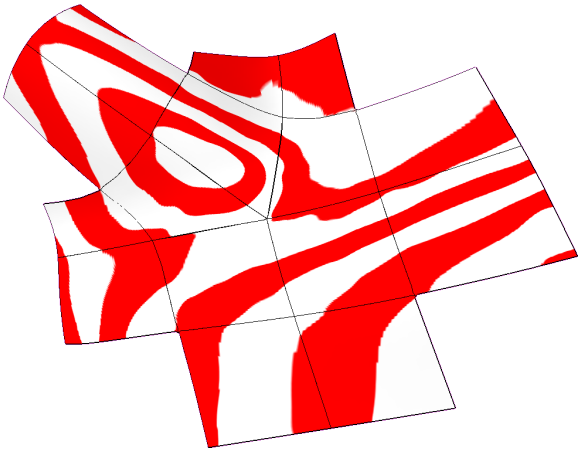
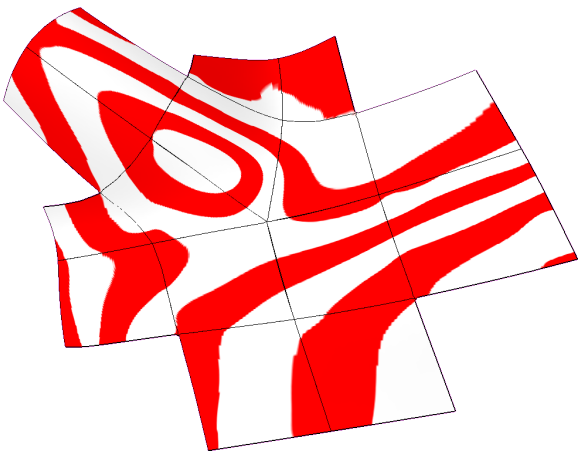


Figure 6: Triangle mesh of a five-sided corner patch



(a) Before fairing



(b) After fairing

Figure 7: Fairing a five-sided corner patch.

Surfaces Fitting, The International Journal of Advanced Manufacturing Technology, Vol. 17, pp. 575–585, 2001.

9. J. Pegna, F-E. Wolter, *Geometrical Criteria to Guarantee Curvature Continuity of Blend Surfaces*, Journal of Mechanical Design, Vol. 114, pp. 201–210, 1992.
10. L. Piegl, W. Tiller, *The NURBS Book*, Springer, 2nd Edition, 1997.
11. P. Salvi, H. Suzuki, T. Várady, *Fast and Local Fairing of B-Spline Curves and Surfaces*, Advances in Geometric Modeling and Processing, pp. 155–163, Springer, 2008.
12. T. Várady, T. Hermann, *Best Fit Surface Curvature at Vertices of Topologically Irregular Curve Networks*, Mathematics of Surfaces VI, Clarendon, pp. 411–428, 1996.
13. T. Várady, R. Martin, *Reverse Engineering*, Handbook of Computer Aided Geometric Design, Ch. 26, Elsevier, 2002.