Hierarchical Surface Fairing with Constraints

Péter Salvi
Eötvös Loránd Science University
salvipeter@gmail.com

Tamás Várady
Geomagic Inc.
varady@geomagic.com

ABSTRACT

The surfaces of complex free-form objects can typically be modeled by a hierarchy of primary surfaces, connecting surfaces and corner patches. Within the context of digital shape reconstruction, these surfaces simultaneously approximate measured data points, satisfy fairness criteria and adhere to continuity constraints according to their dependencies. A new framework algorithm is introduced to perfect existing B-spline surfaces; the algorithm alternates — in a stepwise manner — between continuity constraint satisfaction and fairing by the remaining degrees of freedom. Some well-known fairing methods are adapted to this framework; together with a new approach based on curvature approximation. Constrained fairing for n-sided corner patches, composed of quadrilaterals, is also briefly discussed. A few examples illustrate the results.

Keywords

Digital shape reconstruction, surface fairing, geometric continuity, constraints

1. INTRODUCTION

Digital Shape Reconstruction deals with the creation of CAD models based on measured data points. There are applications where the only goal is to obtain models with high accuracy. In other applications, achieving "visually pleasing" appearance dominates the process; even if tolerances need to be loosened. There is a strong industrial demand for (semi-)automatic methods to visually improve the quality of CAD models while the original, reconstructed features are also preserved.

There is no unique algebraic formulation to describe aesthetic perfection, nevertheless, it is generally agreed in the CAD community that even curvature distribution with large, monotone curvature areas is crucial for obtaining high-quality surface geometries.

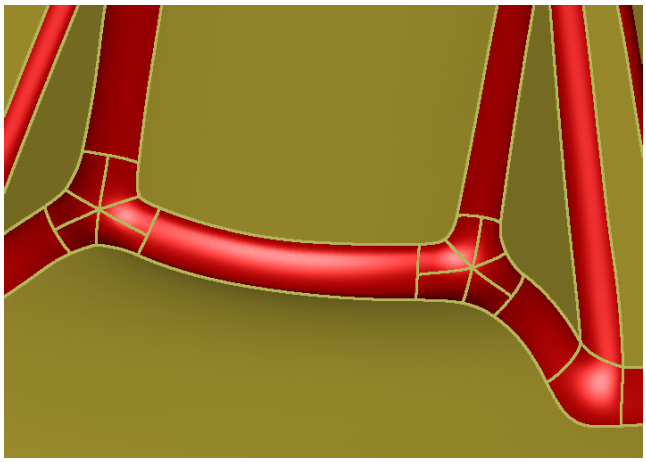


Figure 1: Primaries, connections, corner patches.

Several papers have been published on fairing individual curves and surfaces. These approaches can be divided into two categories. There are the *variational* methods, that simultaneously minimize least-squares distances and various smoothness functionals during surface fitting; and there are the *post-processing* methods that apply changes on already existing geometries.

Traditional methods work quite well when single surfaces are being faired independently of their environment. In this paper, we step forward and investigate the problem of *constrained fairing*, where surfaces must also be smoothly connected to other surface elements.

1.1 Basic Concept

According to the functional decomposition paradigm described in [13], a complex CAD model can be broken down into a set of surfaces with continuity constraints. Typically there is a hierarchy comprising (i) primary surfaces (ii) connecting surfaces, such as fillets, and (iii) corner patches. Surface reconstruction is also performed accordingly, providing continuity constraints from the previous phases. Primary surfaces are independent; fillets smoothly connect to two primary surfaces, and corner patches to several surfaces up in the hierarchy. These include not only fillets, but primaries as well, sharing common boundaries with corner patches at T-node junctions or when setbacks are applied [14], see Fig. 1.

We propose to fair a reconstructed model following the same hierarchical order. A general observation is that primary surfaces are relatively large and are supposed to preserve the original design intent, as opposed to the connections and corner patches, which are much smaller, and the related measured data points are less accurate, so some deviation is more permissible to assure continuity and fairness.

1.2 Goals

There are several papers in the literature that deal with continuity constraints or surface fairing. Generally, when an individual surface is faired, the original accuracy along the surface boundaries is lost; when continuity constraints are satisfied often undesirable curvature artifacts can be observed in the interior of the surface. Our goal is to maintain continuity and provide fair surfaces simultaneously in the post-processing context, which makes it possible to perfect dependent surface geometries of a complex CAD model. The connections between the adjacent surface elements have a large influence on the overall quality of the model, and at least numerical G^2 continuity is recommended. When dealing with trimmed free-form surfaces, algebraic continuity cannot be assured in general, and continuity must be interpreted in the numerical sense, i.e. controlled by tight tolerances.

The focus of our discussion is an algorithm that reaches the above goal in a stepwise manner. Assume that the surface we want to modify is connected to n neighboring surfaces up in the hierarchy (*master surfaces*) with C^0 continuity. Our proposal is to use two types of algorithms in an alternating manner: one sets geometric continuity to the master surfaces, and the other performs fairing retaining the continuity already achieved. For continuity setting, there are two phases: one going from C^0 to G^1, and another one from G^1 to G^2 continuity. The alternation with fairing is essential, because the continuity constraints may crease the surface near the sides, thus harming surface quality. Also, the known fairing methods need to be tweaked such that the established continuity is not destroyed.

1.3 Outline

The paper is structured as follows. In Section 2 we briefly review some papers that are important for this work. Section 3 explains the constrained fairing framework algorithm in details, both the simple four-sided configuration and its extension to n-sided patches. In Section 4 we present algorithms for setting numerical G^1 and G^2 continuity. Section 5 deals with the modification and application of existing fairing algorithms in the continuity-preserving context, along with a new, curvature approximation based method outlined in Section 5.3. The results are illustrated by some examples in Section 6.

2. PREVIOUS WORK

While publications on constrained surface fairing in the post-processing context are not known to the authors, variational constrained fairing methods have recently been investigated in Lai et al. [8] or Hsu et al. [6].

Space limitations prevent us to review the wide range of fairing and continuity fixing algorithms; instead we have collected a few contributions that have been influential to our current research.

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. This was later rephrased in Hermann et al. [5] for G^n continuity. The theorem states 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 5.3.

Concerning curvature approximation, a paper by Greiner [3] gives a quadratic approximation of surface curvature using a reference surface (e.g. a least-square fitting of the data points). The main 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, which will be discussed in more details in Section 5. 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 nicer curves. 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], where a smoothed *target curvature* is approximated with some tolerance control. This is an efficient algorithm using numerical integration. A discrete set of faired data points is computed, which are re-approximated to obtain the final faired surface.

For fairing n-sided corner patches (see Section 3.4), bi-parametric surface algorithms cannot be used. Instead, we adapt an algorithm proposed by Kobbelt [7], where an n-sided surface region is approximated by a triangular mesh and discretized fairing is applied. A local quadratic approximation is used to compute the second-order partial derivatives and minimize the thin plate energy. G^1 continuity constraints can be preserved at sampled data points around the boundary of the n-sided region.

3. THE MAIN ALGORITHM

Constrained surface fairing has to achieve two goals simultaneously: improve surface quality and ensure continuity constraints. In this section, we discuss how the framework algorithm proceeds, and later we will present the actual computations for G^1 and G^2 continuity, and the preferred

algorithms of fairing. Before the alternating, frame-based method is introduced, let us first define, what *frame* means in this context.

3.1 Frames

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 ($i \in [0 \dots n]$, $j \in [0 \dots m]$). 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*.

3.2 Basic Steps

The frame algorithm proceeds in the following way:

1. Insert additional knots into the knot vectors of the surface, if it had too few control points. This step is necessary in order to provide sufficient degrees of freedom for the forthcoming fairing steps in the interior.

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 set G^1 continuity for each boundary.

4. Fair the surface, while retaining G^1 continuity, i.e. only control points within the tangential frame are used for fairing.

5. Fix the tangential frame and set G^2 continuity for each boundary.

6. 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 an alternation of fairing and constraining, until a fair surface with G^2 continuity is achieved.

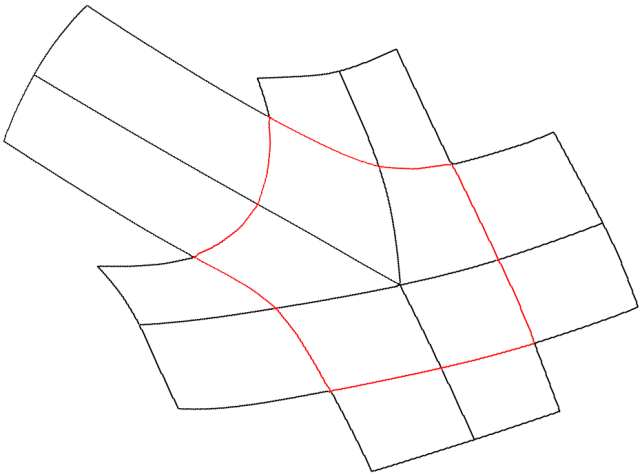


Figure 2: Five-sided corner patch consisting of five quadrilateral surfaces

3.3 Handling Twist Vectors

When we set G^1 continuity for individual boundaries, the so-called twist compatibility condition [1] must also be satisfied; this involves the mixed partial derivatives, which are indirectly determined by the so-called *twist* control points P_{11}, $P_{1(m-1)}$, $P_{(n-1)1}$ and $P_{(n-1)(m-1)}$. When we set G^2 continuity for individual boundaries, a similar compatibility condition needs to be satisfied to tweak the *inner twist* control points P_{22}, $P_{2(m-2)}$, $P_{(n-2)2}$ and $P_{(n-2)(m-2)}$.

With other words, when we compute the cross-derivative functions independently, then for every twist (or inner twist) control points we obtain two values that need to be united in order to get a valid B-spline. There is a range of methods how to determine a common value, though experience shows, that the two candidate control points coming from the adjacent boundaries generally lie very close to each other, so a simple averaging works well. Having tweaked the twist control points, we can repeat the continuity setting algorithms, now constraining only the inner $j \in [2 \dots m-2]$ control points for G^1 and the $j \in [3 \dots m-3]$ control points for G^2 continuity, respectively, which will yield the best possible frames that comprise the new twist values.

3.4 Extension to n-sided Corner Patches

There are corner patches with three or more than four (usually five or six) connecting surfaces. The simplest representation of these is based on the so-called central split, where n quadrilateral surfaces are stitched together, see Fig. 2 for a five-sided example. The main difficulty here is that corner patches consist of more than one biparametric surface, so we have to ensure continuity not only along the external boundaries, but along internal subdividing curves between the quadrilaterals as well. Moreover, no fairing algorithm is known to the authors that can handle multiple surfaces at the same time.

In order to cope with these difficulties, the procedure presented in Section 3.2 needs to be modified. First we fair the entire corner patch. The rest of the algorithm works on the quadrilateral level, so this is the only part, where the sub-

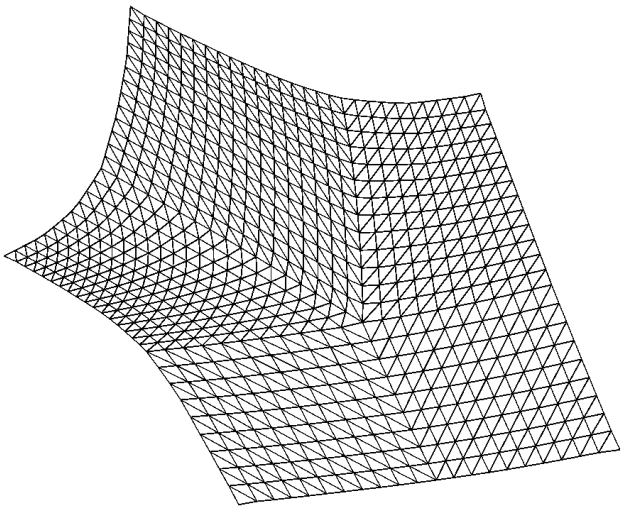


Figure 3: Triangle mesh of a five-sided corner patch

dividing boundaries are modified. It aims at creating a fair initial shape, and may be omitted, if the original patch was of sufficiently good quality.

We use discrete smoothing to achieve the global fairing effect, as described below. After sampling the n-sided region, a triangular mesh is obtained (see Fig. 3), and an algorithm suggested by Kobbelt [7] is performed. This can inherit G^1 continuity at discrete points of the boundaries, if needed. Using the faired n-sided mesh, we refit the quadrilateral surfaces, and fix the internal subdividing curves and related cross-derivative functions for the forthcoming internal continuity setting. This algorithm, due to precision loss, would not be suitable in the later steps of our algorithm, but it is easy to use regardless of the number of quadrilaterals.

Having the group of n surfaces, sharing the fixed subdividing boundaries, the framework algorithm is modified, as follows.

1. For each quadrilateral set G^1 continuity with its neighbors. (Adjust twists, recompute.)
2. For each quadrilateral perform local fairing, while retaining G^1 continuity.
3. For each quadrilateral set G^2 continuity with its neighbors. (Adjust inner twists, recompute.)
4. For each quadrilateral perform local fairing, while retaining G^2 continuity.

4. SETTING CONTINUITY

We set continuity between two B-spline surfaces M (master) and S (slave) by modifying the appropriate frame of S. The general idea is to draw up the equations for a number of sampled parameter points along the border and minimize the least-squares error of the equation system.

We will treat the algorithm on a per side basis. The equations here use index ranges that also include the twists. As explained in Section 3.3, these may need to be pinned down, but the necessary adjustments of the algorithms are straightforward.

4.1 Tangential Continuity

Let M and S be joined (without loss of generality) along the u_0 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. 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 superfluous control point movements, we minimize the deviation only in the surface normal direction. This leads to

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

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 $(A^TA)\underline{x} = A^T\underline{b}$.

4.2 Curvature Continuity

We have the same assumptions as in the G^1 case, but now we already 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_0, 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_0, 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:

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

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

$$\frac{\langle C_k''(u_0), \underline{n}_k \rangle}{\|C_k'(u_0)\|^2} = \kappa_k^M. \tag{4}$$

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

$$\begin{aligned} C_k'(u_0) &= \frac{p}{u_{p+1} - u_0} (\hat{P}_1 - \hat{P}_0), \\ C_k''(u_0) &= \frac{p-1}{u_{p+1} - u_0} \left(\frac{p}{u_{p+2} - u_0} (\hat{P}_2 - \hat{P}_1) \right. \\ &\quad \left. - \frac{p}{u_{p+1} - u_0} (\hat{P}_1 - \hat{P}_0) \right), \end{aligned}$$

see [10] (assuming that $u_0 = u_1 = \cdots = u_p$).

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

$$\begin{aligned} \Delta\kappa_k \left\|C_k'(u_0)\right\|^2 \frac{(u_{p+1} - u_2)(u_{p+2} - u_2)}{p(p-1)} \\ = \sum_{j=2}^{m-2} N_j^{V,q}(v_k) \left\langle \underline{d}_j, \underline{n}_k \right\rangle, \end{aligned} \tag{5}$$

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 superfluous control point movements, the P_{2j} points are only allowed to move in an approximate perpendicular direction ($\underline{o}_j$ for P_{2j}). Alternatively, we can require that the sum of the squared deviations should be minimal.

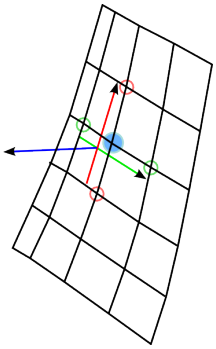


Figure 4: Approximate perpendicular direction computed from control points

Solution by Fixed Directions

An intuitive choice for perpendicular directions is the surface normal at the Greville abscissae corresponding to the control point. An easy alternative is to get the cross product of the difference of the neighboring control points (Fig. 4), 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 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, $(A^T A)\underline{x} = A^T \underline{b}$ gives a least-squares approximation.

5. FAIRING ALGORITHMS

In addition to continuity setting, we also need an algorithm to fair bi-cubic quadrilateral surface patches. We have several options, but there is also a restriction: the smoothing process should preserve G^1/G^2 continuity. After some comments on two existing methods, we will present a new alternative based on curvature approximation.

5.1 Knot Removal and Reinsertion

This fairing method operates on the control points, usually choosing the one where the C^3 jump is the greatest. We perform this operation in an iterative manner several times, combined with some heuristics like taboo search. We can restrict the choice of control points to those inside the fixed frame, which 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.

5.2 Target Curvature Based

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 setting step. Another drawback is that the algorithm relies on fairing the isocurves of the surface. This may cause artifacts in complex saddle-like surfaces.

5.3 Curvature Approximation Based

We will use the curvature approximation outlined in Section 2 to enhance the previous algorithm. 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

$$\begin{aligned} \sum_c \mathrm{trace}\left(\mathrm{Hess}_R(R_c)\right)^2 &= (\kappa_1^R + \kappa_2^R)^2, \\ \sum_c \det\left(\mathrm{Hess}_R(R_c)\right) &= \kappa_1^R \cdot \kappa_2^R. \end{aligned}$$

(a) Before fairing

(b) After fairing

Figure 5: Isophotes of a car body panel faired by the curvature approximation method

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

$$\begin{aligned} \sum_c \mathrm{trace}\left(\mathrm{Hess}_R(S_c)\right)^2 &\approx (\kappa_1^S + \kappa_2^S)^2, \\ \sum_c \det\left(\mathrm{Hess}_R(S_c)\right) &\approx \kappa_1^S \cdot \kappa_2^S. \end{aligned}$$

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.

(a) Before fairing

(b) After fairing

Figure 6: Fairing an X-node

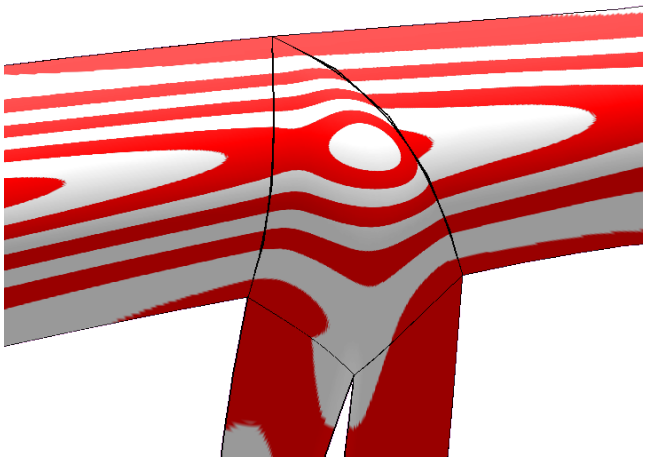
We need an equation system that depends linearly on the control points of S, so it will be easy to minimize. One option is to create a target mean curvature by smoothing the traces of Hessian matrices in the sampled points. Another alternative is to average every element of the Hessian matrices over 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. 5.

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.

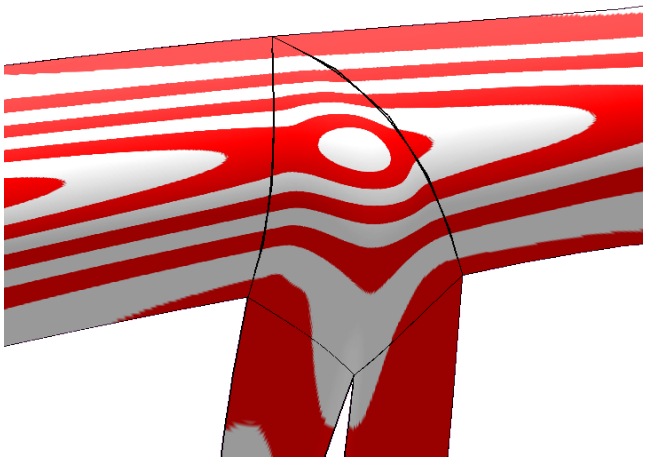
6. EXAMPLES

Figures 6 and 7 both show the effect of constrained fairing on four-sided corner patches. In these cases, the surface configuration is relatively simple — connecting four fillets. The original surfaces (a) show reasonable isophotes, but the stripes are not tangential at the common boundaries (only G^1 continuity). After constrained fairing (b) the images show smooth stripes, i.e. numerical G^2 continuity has been achieved, and the interiors have been nicely affected, smoothly joining the master surfaces.

We have a different case in Fig. 8. The configuration connects three fillets and two primary surfaces. Here the original corner patch (a) already had numerical G^2 that only needed very minor adjustments. The master surfaces have been retained, but the middle patch was faired (b) increasing surface quality to a great extent.



(a) Before fairing



(b) After fairing

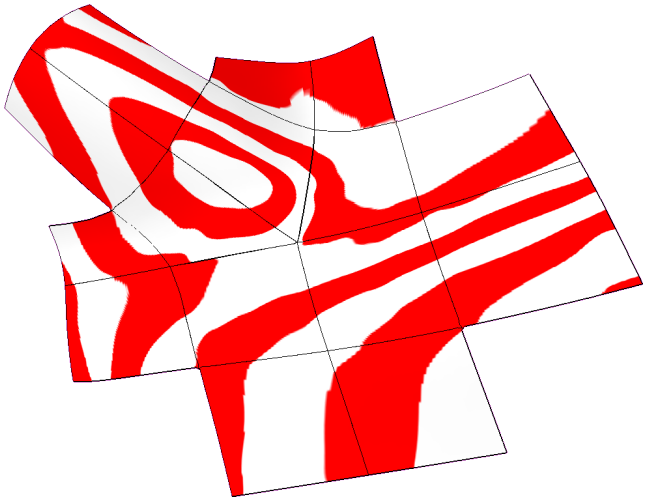
Figure 7: Fairing another X-node

These examples show individual corner patches that have been faired and constrained within the hierarchical framework of dependent surfaces, as described earlier in Section 1.1.

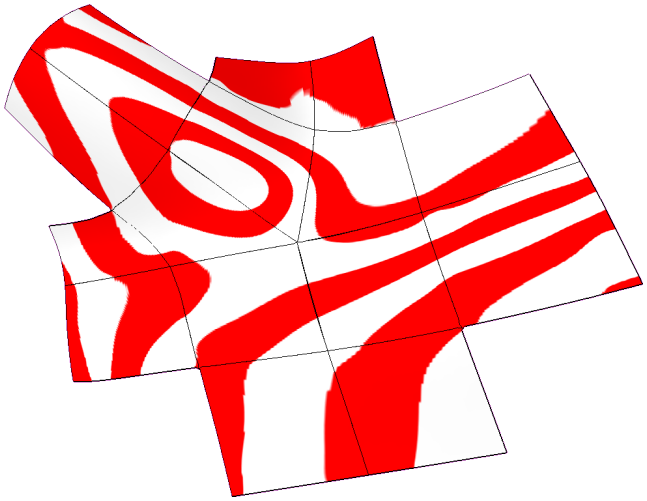
7. CONCLUSION AND FUTURE WORK

A new approach for fairing surface elements of complex CAD models was presented to support digital shape reconstruction based on measured data. The process combines fairing and continuity setting algorithms to perfect functionally decomposed surfaces following a hierarchical order.

Nonetheless, there is still room for future research and enhancements. For example, in the n-sided corner patch case, the discrete fairing step is going to be replaced by some continuous fairing technique, which is subject of our current investigations. The twist compatibility setting can also be replaced using a different system of equations, i.e. setting continuity not side by side, but for the entire frame. An-



(a) Before fairing



(b) After fairing

Figure 8: Fairing a five-sided corner patch.

other idea is to fair functionally decomposed surfaces locally, i.e. using selected portions of adjacent primaries, connections and corner patches and then apply hierarchical fairing which will affect only the selected areas.

8. ACKNOWLEDGEMENT

The authors would like to acknowledge the support of Geomagic, Inc., Research Triangle Park, North Carolina, that contributed to conduct this research and evaluate the results on industrial examples.

9. REFERENCES

[1] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, fifth edition, 2002.

[2] G. Farin, G. Rein, N. Sapidis, and A.J. Worsey. Fairing cubic B-spline curves. *Computer Aided Geometric Design*, 4(1–2):91–103, July 1987.

[3] G. Greiner. Curvature approximation with application to surface modeling. In J. Hoschek and P. Kaklis, editors, *Advanced Course on FAIRSHAPE*. B.G. Teubner, 1996.

[4] S. Hahmann and S. Konz. Knot-removal surface fairing using search strategies. *Computer-Aided Design*, 30(2):131–138, 1998.

[5] T. Hermann, G. Lukács, and F-E. Wolter. Geometrical criteria on the higher order smoothness of composite surfaces. *Computer Aided Geometric Design*, 16(9):907–911, 1999.

[6] K-L. Hsu and D-M. Tsay. Corner blending of free-form N-sided holes. *IEEE Computer Graphics and Applications*, 18(1):72–78, 1998.

[7] L. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3-4):142–158, 2000.

[8] J-Y. Lai and W-D. Ueng. G^2 continuity for multiple surfaces fitting. *The International Journal of Advanced Manufacturing Technology*, 17:575–585, 2001.

[9] J. Pegna and F. Wolter. Geometric criteria to guarantee curvature continuity of blend surfaces. *ASME Transactons, J. of Mech. Design*, 114, 1992.

[10] L. Piegl and W. Tiller. *The NURBS Book*. Monographs in visual communication. Springer, second edition, 1997.

[11] P. Salvi, H. Suzuki, and T. Várady. Fast and local fairing of B-spline curves and surfaces. In *Advances in Geometric Modeling and Processing*, pages 155–163. Springer, 2008.

[12] T. Várady and T. Hermann. Best fit surface curvature at vertices of topologically irregular curve networks. In *Proceedings of the 6th IMA Conference on the Mathematics of Surfaces*, pages 411–427. Clarendon Press, 1996.

[13] T. Várady and R. Martin. Reverse engineering. In G. Farin, J. Hoschek, and M-S. Kim, editors, *Handbook of Computer Aided Geometric Design*, chapter 26. Elsevier, 2002.

[14] T. Várady and A. Rockwood. Geometric construction for setback vertex blending. *Computer-Aided Design*, 29(6):413–425, 1997.