Knot-Removal Surface Fairing using Search Strategies

Stefanie Hahmann[1] *Stefan Konz*[2]

Revised Version, December 1997
to be published in CAD 1998

Abstract. *This paper presents two automatic fairing algorithms for parametric C^2-continuous bi-cubic B-spline surfaces. The fairing method consists of a knot removal and knot reinsertion step which locally smooths the surface. Search strategies like best-first-search and simulated-annealing are searching for the global minimum of the fairing measure. The best-first-search algorithm constructs only partially a search tree and reduces significantly the complexity of a systematic search. Simulated annealing is a heuristic algorithm which needs a probability function and some further parameters as input. Both methods can satisfy end constraints and tolerances. Their performance is discussed for two numerical experiments.*

Keywords: *B-spline surfaces, fairness, knot insertion, knot removal, simulated annealing, best-first-search*

1. INTRODUCTION

The free form surface has been becoming an indispensable part of powerful CAD-systems. Bicubic tensor product B-splines, which are the subject of this paper, are often used in geometric modelling due to their well known advantages which result basicly from the local support of the basis splines and their dependence on the knot vector. However, the designer is not always satisfied with the fairness or smoothness of the resulting surface obtained from interpolation or approximation of some data sets.

[1] Laboratoire LMC-IMAG, University of Grenoble, B.P. 53, F-38041 Grenoble cedex 9, France, Tel: +33 4 76 63 57 88, Fax +33 4 76 63 12 63, `hahmann@imag.fr`

[2] Fachbereich Informatik, Universität Kaiserslautern, D-67653 Kaiserslautern, Germany

In the literature a lot of different fairing criteria are offered which are either more aesthetically or more technically based. Finally it's the designer's subjective decision to accept a surface as fair or not. In most cases an appropriate graphical interrogation method [14,16] is used, which has the capability to emphasize surface imperfections and which allows a visual inspection of fairness or unfairness during the design process.

1.1 Principles of surface fairing

A fair surface can be obtained by two different ways:

- modelling surfaces with constraints
- post-processing surface fairing.

The first way includes the fairness criterion as a constraint in the design process. The degrees of freedom of the surface geometry are used as parameters in an optimization method. The so called *variational design of smooth surfaces*, consists of the minimization of a physical based fairness criteria (i.e. minimal bending energy of torsion and flexure, minimal jerk), which is incorporated in the interpolation or approximation process (see [15,3] for examples and more details).

Non-linear surface smoothing methods can be found in [22,9,24,11]. These methods are often time consuming due to the non-linear constraints, but they do not approximate the physical criteria.

The other way to obtain fair surfaces is to apply a *post-processing fairing* method to a given surface. Such a surface already has (approximatively) its final shape but some imperfections occur (digitization errors e.g.). The fair surface is not allowed to change its shape too much. (see [5,20,12] as examples).

1.2 Motivation

The fairing criteria are of very different kinds, like minimal energy criteria [13], criteria based on the surface areas [27], or aesthetic criteria, as light reflection behaviour of a surface [21]. We want to focus here first on a fairing principle special for cubic spline curves, which was successfully used by Farin/Sapidis [29] as a local scheme, and by Kjellander [19] as a global scheme. The method of Farin/Sapidis is based on the fairing principle that

(***C***) *a C^2-cubic B-spline curve is fair if the curvature function $\kappa(s)$ is continuous, has the appropriate sign, and is as close as possible to a piecewise monotone function with as few as possible monotone pieces [29].*

A curve x is therefore fairer than y at $t \in (a,b)$ if

$$|\kappa'_x(t^+) - \kappa'_x(t^-)| \leq |\kappa'_y(t^+) - \kappa'_y(t^-)|, \tag{$*$}$$

where κ'_x, κ'_y mean the derivative of the curvature of x and y with respect to the arc length. $(*)$ is always an equality except at the interior knots of the B-spline curve. Farin/Sapidis use then a *knot-removal-reinsertion* step at the most offending knot t^*, which makes the curve momentarily C^3-continuous at t^* before proceeding at other knots. The iterations continue as long as the global fairing measure (sum of the curvature discontinuities over all knots) decreases.

In the present paper we introduce a new fairing method for bicubic tensor product B-spline surfaces based on the knot-removal-reinsertion step combined with two search strategies, the *best-first-search* and the *simulated annealing.* This fairing method will be *local* in contrast to Kjellander's method for bicubic surfaces. The principal reason for incorporating search strategies into the fairing process is to overcome the drawback of Farin/Sapidis's algorithm which stops, when the global fairing measure achieves its first local minimum.

Section 2 begins by fixing the notations, recalls some knot-removal-reinsertion algorithms and describes Farin/Sapidis' curve fairing algorithm. In order to solve the problems mentioned above we want to present in section 3 two fairing algorithms for B-spline surfaces based on special search strategies like best-first-search and simulated-annealing. The heuristic *simulated-annealing algorithm* for example, depends on a probability function and some other parameters and searches for a global minimum of the fairness measure. Both algorithms are searching for the global minimum of the fairness criterion and result in an optimal surface. The paper then discusses with some examples the performance of both fairing methods for bicubic B-spline surfaces. Some concluding remarks indicate generalizations to preserve tolerances and end constraints.

2. KNOT REMOVAL FOR B-SPLINE CURVES AND SURFACES

2.1 Knot removal for curves

Given two positive integers n and k and $\mathbf{t} = (t_i)_{i=0}^{n+k}$ a sequence of real numbers with $t_i < t_{i+k}$ $(i = 0, \ldots, n)$. The $n+1$ B-spline basis functions of order k (degree $k-1$) associated with the *knot vector* $\mathbf{t}$ are denoted by $(N_{i,k,\mathbf{t}})_{i=0}^{n}$ (or simply by $N_{i,k}$) and are assumed to be normalized to sum to one. They span a linear space of functions: $\mathcal{S}_{k,\mathbf{t}}$.

A parametric *B-spline curve* $\mathbf{x}$ in $I\!R^2$ of order k, defined by a set of control points $\mathbf{d}_i \in I\!R^2$ and a knot vector $\mathbf{t}$, is given by

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{d}_i N_{i,k}(t)\,, \quad t \in [t_{k-1}, t_{n+1}] \tag{1}$$

where each component belongs to $\mathcal{S}_{k,\mathbf{t}}$. Note that the B-spline curve $\mathbf{x}(t)$ is only defined over $[t_{k-1}, t_{n+1}]$, i.e. over $n-k+2$ intervals, $t_k, \ldots, t_n$ are called *inner knots.* $\mathbf{x}$ is a piecewise spline which is at most C^{k-2}-continuous at the knots t_i, $(i = k, \ldots, n)$.

The knot removal step consists of approximating a given spline $\mathbf{x} = \sum_{i=0}^{n} \mathbf{d}_i N_{ik\mathbf{t}}$ by a spline $\tilde{\mathbf{x}} = \sum_{i=0}^{n-1} \tilde{\mathbf{d}}_i N_{ik\tilde{\mathbf{t}}}$ on a knot vector $\tilde{\mathbf{t}}$ which is a subsequence of $\mathbf{t}$ with one knot less.

It follows that $\tilde{\mathbf{x}}$ can be expressed in the same basis as $\mathbf{x}$, because of $\mathcal{S}_{k,\tilde{\mathbf{t}}} \subset \mathcal{S}_{k,\mathbf{t}}$, i.e.

$$\tilde{\mathbf{x}} = \sum_{i=0}^{n-1} \tilde{\mathbf{d}}_i N_{i,k,\tilde{\mathbf{t}}} = \sum_{i=0}^{n} \mathbf{D}_i N_{i,k,\mathbf{t}} \,. \tag{2}$$

The control points $\mathbf{D} = (\mathbf{D}_0, \ldots, \mathbf{D}_n)$ are obtained from the control points $\tilde{\mathbf{d}} = (\tilde{\mathbf{d}}_0, \ldots, \tilde{\mathbf{d}}_{n-1})$ by

$$\mathbf{D} := A\tilde{\mathbf{d}}\,, \tag{3}$$

where A is the $(n+1, n)$-matrix, called *knot insertion matrix* of order k from $\tilde{\mathbf{t}}$ to $\mathbf{t}$. We can assume without any restriction, that $\tilde{\mathbf{t}} = \mathbf{t} \backslash \{t_{j+1}\}$, $(j+1) \in \{k, \ldots, n\}$, then A is given by

$$A = \begin{bmatrix} I_{j-k+2} & & \\ & A_j & \\ & & I_{n-j} \end{bmatrix} \text{ with } A_j = \begin{bmatrix} (1-\alpha_{j-k+2}) & \alpha_{j-k+2} & & \\ & \ddots & \ddots & \\ & & (1-\alpha_j) & \alpha_j \end{bmatrix} \tag{4}$$

$\alpha_r = \frac{t_{r+1} - \tilde{t}_r}{\tilde{t}_{r+3} - \tilde{t}_r}$ ($\tilde{t}_r = t_r$ for $r = 0, \ldots, j$; $\tilde{t}_r = t_{r+1}$ for $r = j+1, \ldots, n-1$). I_m being the m-dimensional unit matrix [2].

While *knot insertion* does not change the shape of the curve, the "inverse" process of knot removal cannot be carried out in general without changing the shape, except the case where the curve is C^{k-1} at t_{j+1}, i.e. its continuity order is higher than it should be according to its multiplicity). In the future we will use knot removal for bicubic surfaces, therefore we restrict the following considerations to the cubic case, i.e. $k = 4$.

There are now different possibilities to determine approximated solutions of the *knot removal* problem: $\quad A\tilde{\mathbf{d}} = \mathbf{d} \quad$ ($\tilde{\mathbf{d}}$ unknown).

From the approximation point of view, knot removal can be solved by $\min \|\tilde{\mathbf{x}} - \mathbf{x}\| : \{\tilde{\mathbf{x}} \in \mathcal{S}_{k,\tilde{\mathbf{t}}}\}$ with respect to some appropriate norm $\|\cdot\|$. For more details see [23]. In general, all control points are involved in that knot removal procedure, which is therefore a global one.

We want our fairing methods to work as locally as possible. This means, that after reinsertion of the removed knot t_{j+1} (2), a minimum number of control points of the curve $\tilde{\mathbf{x}}$ should differ from the original control points $\mathbf{d}_i$. The knot removal problem can locally be solved by calculating an approximate solution of the overdetermined (3,2)-subsystem $A_j\tilde{\mathbf{d}} = \mathbf{d}$ (4), which states as follows:

$$\begin{bmatrix} \alpha_{j-2} & 0 \\ (1-\alpha_{j-1}) & \alpha_{j-1} \\ 0 & (1-\alpha_j) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{d}}_{j-2} \\ \tilde{\mathbf{d}}_{j-1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{j-2} - (1-\alpha_{j-2})\mathbf{d}_{j-3} \\ \mathbf{d}_{j-1} \\ \mathbf{d}_j - \alpha_j \mathbf{d}_{j+1} \end{bmatrix} \tag{5}$$

and $\tilde{\mathbf{d}}_i = \mathbf{d}_i$ for $(i = 0, \ldots, j-3)$ and $\tilde{\mathbf{d}}_i = \mathbf{d}_{i+1}$ for $(i = j+1, \ldots, n)$.
Different approximate solutions of this system are possible. A detailed description of these

knot removal algorithms can be found in [7,6].
We want to focus on two of them, because they can also be used for knot removal in the surface case:

Minimal region knot removal:
Solve the linear system (5) without the second equation. After reinsertion of t_{j+1} one gets $\mathbf{D}_{j-2} = \mathbf{d}_{j-2}$ and $\mathbf{D}_j = \mathbf{d}_j$ and only the control point $\mathbf{d}_{j-1}$ has moved. This knot-removal-reinsertion step changes the curve only in the interval $[t_j, t_{j+4}]$ and is therefore the most local knot removal procedure, see [7]. Nevertheless it should be mentioned, that the difference between both curves will generally be higher than in the following case.

Least squares knot removal:
$\tilde{\mathbf{d}}_{j-2}$ and $\tilde{\mathbf{d}}_{j-1}$ are determined as least squares solution of system (5). Hence, three control points have moved, such that $\sum_{i=0}^{2} \|\mathbf{D}_{j-i} - \mathbf{d}_{j-i}\| = min$. The influence on the curve is limited to 3 control points, i.e. to the interval $[t_{j-2}, t_{j+4}]$, see [29].

2.2 Knot removal for surfaces

Let us now describe the classical way to remove and reinsert knots in the bivariate case. To do so, the knot operations (section 2.1) have to be adapted to the tensor product surface description. Note that it's an inherent property of tensor products that a lot of algorithms on surfaces (e.g. de Boor algorithm, degree elevation, ...) can be reduced to the monodimensional curve algorithms with respect to one of the two parameters u and v.

Let k and l be two positive integers and let $\mathbf{u} = (u_i)_{i=0}^{n+k}$ and $\mathbf{v} = (v_j)_{j=0}^{m+l}$ be two knot vectors with $u_i < u_{i+k}$ and $v_j < v_{j+l}$. A parametric *tensor product B-spline surface* $\mathbf{X}$ in $\mathbb{R}^3$ of order (k, l) is then defined by

$$\mathbf{X}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{d}_{ij} N_{i,k,\mathbf{u}}(u) N_{j,l,\mathbf{v}}(v)\,, \quad (u,v) \in [u_{k-1}, u_{n+1}] \times [v_{l-1}, v_{m+1}] \tag{6}$$

where the coefficients $\mathbf{d}_{ij} \in \mathbb{R}^3$ form the control net, and where (u_i, v_j) $(i = k, \ldots, n;\ j = l, \ldots, m)$ are called *inner knots*. For more details about B-splines, see [4,30].
Knot removal for surfaces means *removing knot lines* from the underlying knot grid $(u_i, v_j)_{i,j=k,l}^{n,m}$. Knot removal and reinsertion for example in v-direction (e.g. removing the knot v_{s+1}) therefore works as follows: One has to apply a univariate knot removal algorithm (see (2) - (5)) successively to the following $(n - k + 1)$ univariate B-splines:

$$\mathbf{x}_i(t) = \sum_{j=0}^{m} \mathbf{d}_{ij} N_{j,l,\mathbf{v}}(t)\,, \quad t \in [v_{l-1}, \ldots, v_{m+1}]\,, \quad i = k, \ldots, n\,, \tag{7}$$

at the inner knot $t = v_{s+1}$. After re-numbering the rows of control points one gets finally the surface

$$\tilde{\mathbf{X}} = \sum_{i=0}^{n} \sum_{j=0}^{m-1} \tilde{\mathbf{d}}_{ij} N_{i,k,\mathbf{u}} N_{j,l,\tilde{\mathbf{v}}} = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{D}_{ij} N_{i,k,\mathbf{u}} N_{j,l,\mathbf{v}}\,,$$

where $\tilde{\mathbf{v}}$ is the subsequence of $\mathbf{v}$ with one knot (v_{s+1}) less. The control points $\mathbf{D}_{ij}$ are obtained after reinsertion of the knot line v_{s+1} into $\tilde{\mathbf{v}}$ by

$$\text{for } i = k, \ldots, n \qquad \text{do} \qquad D^i := A^i \tilde{\mathbf{d}}^i,$$

where $\mathbf{D}^i = \{\mathbf{D}_{i0}, \ldots, \mathbf{D}_{im}\}$ and $\tilde{\mathbf{d}}^i = \{\tilde{\mathbf{d}}_{i,0}, \ldots, \tilde{\mathbf{d}}_{i,m-1}\}$ and A^i corresponds to the univariate knot insertion matrix A (3), (4).
Note, that this knot removal step makes the surface cross derivatives of third order at the patch boundaries at v_{s+1} continuous, i.e.

$$\frac{\partial^3 \mathbf{X}}{\partial v^3}(u, v_{s+1}^-) = \frac{\partial^3 \mathbf{X}}{\partial v^3}(u, v_{s+1}^+) \, .$$

Bivariate knot removal and re-insertion in u-direction works analogously. With knot removal first in u-direction and then in v-direction, or vice-versa, it is generally not possible to obtain C^3-continuity at (u_i, v_j). Several variants and applications of knot insertion and removal for B-spline surfaces can be found in [10].

2.3 Farin/Sapidis' fairing algorithm

Farin/Sapidis' fairing algorithm for cubic splines is based on the fact that after a knot-removal-reinsertion step of the knot t_j the new curve $\tilde{\mathbf{x}}(t)$ is C^3 at t_j. Hence, the local fairness measure decreases to zero:

$$z_j = |\kappa'(t_j^+) - \kappa'(t_j^-)| \, . \tag{8}$$

By repeating this procedure several times, always at the most offending knot (i.e. t_k with $z_k = \max(z_i) : \{i \in (4, \ldots, n)\}$), they could expect, that the sum of the local measures

$$\xi = \sum_{i=4}^{n} z_i \tag{9}$$

decreases also. According to their fairness criterion (8), ξ is an appropriate global measure to control the number of monotone pieces of the entire curve. Therefore Farin/Sapidis' algorithm states as follows:

```
DO  1. compute $\xi = \sum z_i$
    2. determine $t_j$ with $z_j = \max(z_i)$
    3. knot-removal-reinsertion step at $t_j$
UNTIL (stop if a suitable criterion is fulfilled).
```

We want to close this section with the conclusion that a knot-removal-reinsertion step has a local fairing effect and can therefore be used in a fairing method for the entire curve, or tensor product surface resp. The main problem is to find a sequence of knots such that repeated knot-removal-reinsertion at these knots leads to a *global minimum of the fairness measure* ξ (9).

3. FAIRING B-SPLINE SURFACES

The curve fairing method can easily be extended to bi-cubic tensor product surfaces. Sure, one can simply apply the curve scheme to each row and each column of control points of the surface control net (see [8]). This procedure has two main disadvantages: the fairness measures are not adapted to the surface geometry and therefore do not justify the selection of a knot as the most offending one. And the "stop"-criterion for the algorithm does not guarantee finding the global minimum of the fairness measure ξ. The algorithm stops when the first local minimum of that function is reached. And this is probably not the lowest one. We present now two surface fairing algorithms which are based on knot-removal-reinsertion (which is a very fast fairing step) and which use two search strategies to *optimize the fairing process.*

Instead of performing the fairing step at the most offending knot while expecting the global fairness measure to decrease (see [29]), we intend to perform the fairing step uniquely at knots which ensure a decreasing fairness measure. The fairing methods are based on the following parameters:

(A) fairing step: knot-removal-reinsertion
(B) fairness measure (local and global)
(C) visualization
(D) the scheduler (selection method) which determines the sequence of knot pairs to be treated
(E) stop-condition of the iteration.

A so called *search tree* (see figure 1) describes all possible surfaces after a fixed number k of fairing steps. The surface, which minimizes the fairness criterion is taken as the faired surface.

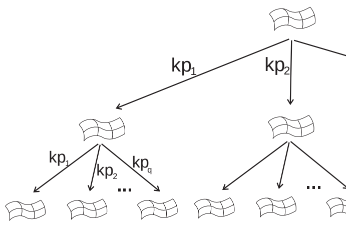


Figure 1: search tree of the iterative fairing method

The search tree advises on the root the starting surface to be faired. Each vertex of the tree represents a B-spline surface that results from one fairing step (knot removal and reinsertion) at one knot of the parent vertex. The edges of the tree are marked with the knot pair kp_i which is treated by the knot operations. The search tree lists systematically all possible steps of the fairing algorithm. Each path from the root to a leaf of the tree describes the history of one algorithm call. The labels kp_i in figure 1 stand for the different inner knot pairs. As for the knot vectors $\mathbf{u} = (0,0,0,0,1,2,3,4,4,4,4)$, $\mathbf{v} =$

$(0,0,0,0,1,5,7,7,7,7)$ we get $kp_0 = (u_4, v_4) = (1,1), kp_1 = (u_4, v_5) = (1,5), \ldots, kp_6 = (u_6, v_5) = (3,5)$.

At least one of the leaves represents a surface with a global minimum of the fairing measure. The global minimum (after k iterations) can only be found when performing a polynomial number $O((n \cdot m)^k)$ of fairing steps. The aim of the search strategies is to find this path without constructing the whole tree because of the polynomial number of vertices. The way the search tree is constructed and transversed, depends on the two search strategies, *best-first-search* and *simulated annealing.* The five mean characteristics (A) - (E) of our fairing methods are now presented:

(A) Fairing step

We use a knot-removal-reinsertion as a *local* fairing step for the surface and refer to it by KRR. The bivariate method presented in section 2.2 has the drawback to be *global.* If one knot line is removed, e.g. v_{s+1}, then the whole columns of control points $\mathbf{d}_{i,s-2}, \mathbf{d}_{i,s-1}, \mathbf{d}_{i,s}$, $(i = 0, \ldots, n)$ is be modified.
Our knot-removal-reinsertion step (KRR) is a localized version of the classical method. It locally performs a fairing step at the inner knot pairs (u_{r+1}, v_{s+1}) $(r, s = 3, \ldots, n, m)$ by modifying only a small set of control points at each step. Removing and reinsertion of the knot line v_{s+1} makes the third order partial derivative $\frac{\partial^3 \mathbf{X}}{\partial v^3}(u, v_{s+1})$ continuous for all $u \in [u_3, u_{n+1}]$. We only need the surface to have this third order partial derivative continuous at the knot (u_{r+1}, v_{s+1}). Therefore it is not necessary to remove and reinsert the whole knot line. The following Lemma shows that it is sufficient to apply the univariate (curve) KRR at v_{s+1} for only three curves which are defined by three columns of the surface control points.

Lemma. *Given a bi-cubic C^2 B-spline surface $\boldsymbol{X}(u,v)$ as defined in (6). If the three curves $\mathbf{x}_i(v) = \sum_{j=0}^{m} \mathbf{d}_{ij} N_{j,4,\mathbf{v}}(v)$ $(i = r-2, r-1, r)$ are C^3-continuous in $v = v_{s+1}$, then the third partial derivative $\frac{\partial^3 \boldsymbol{X}}{\partial v^3}(u,v)$ of the surface is continuous at (u_{r+1}, v_{s+1}). (analogous for u).*

Proof: Substituting the C^3-continuity condition of the three curves at a point (v_{s+1}) into that for surfaces at a point (u_{r+1}, v_{s+1}) gives the desired result. $\square$

In the following we distinguish KRR in u- and v-direction. KRR in v-direction means that we remove and reinsert a knot v_{s+1} from the knot vector $\mathbf{v}$. Similar to the univariate case we prefer the following two methods:

minimal region algorithm, called *mr-KRR*, in v-direction:
The univariate minimal region knot removal (sect. 2.1) is applied to remove and reinsert the knot $t = v_{s+1}$ from three curves $\mathbf{x}_i(t) = \sum_{j=0}^{m} \mathbf{d}_{ij} N_{j,4,\mathbf{v}}(t)$, $i = r-2, r-1, r$ The same can be done in u-direction. The distance of the old control net from the new one

can be relatively high, because only one control point per curve $d_{i,s-1}$ has been modified. The surface is then C^3-continuous with respect to v at the knot (u_{r+1}, v_{s+1}), see Lemma. (analogous for u).

least squares algorithm, called *ls-KRR*, in v-direction:
Here also the univariate least-squares knot removal and reinsertion step (sect. 2.1) is applied three times to $\mathbf{x}_i(t) = \sum_{j=0}^{m} \mathbf{d}_{ij} N_{j,4,\mathbf{v}}(t)$ for $i = r-2, r-1, r$ at the knot $t = v_{s+1}$. Analogous in u-direction. This method minimizes the squared distance between the old and new control net by moving nine control points: $\mathbf{d}_{ij}$, $i = r-2, r-1, r$ and $j = s-2, s-1, s$. Therefore the shape of the surface does not change very abruptly.

Notice, that there is no symmetry in the KRR-steps. Changing u and v leads to different results, because they depend on different control points. Applying KRR first in u-direction and then in v-direction leads to a surface which is only C^3-continuous with respect to v.

The general purpose of a fairing method is to *obtain maximal fairing effect within a predetermined 'tolerance'*. For small tolerances the ls-KRR seems to be the most appropriate method. Otherwise both methods lead to satisfying good results (see also section 4).

(B) Fairness measure

The *local measures* z_{ij}^u, z_{ij}^v of a knot pair (u_i, v_j) from a surface $\mathbf{X}(u,v)$ which evaluate the fairness of the knot pair are defined by the partial derivation of a curvature function $g(u,v) = g(\kappa_{min}(u,v), \kappa_{max}(u,v))$ along one of the two parameters of the surface, where $\kappa_{min}, \kappa_{max}$ are the principal curvatures. The Gaussian curvature $K = \kappa_{min} \cdot \kappa_{max}$ or $\kappa_{min}^2 + \kappa_{max}^2, \ldots$ are appropriate candidates for g.

$$z_{ij}^u = \frac{|\frac{\partial g}{\partial u}(u_i, v_j)|}{\|\frac{\partial X(u,v)}{\partial u}\|}, \qquad z_{ij}^v = \frac{|\frac{\partial g}{\partial v}(u_i, v_j)|}{\|\frac{\partial X(u,v)}{\partial v}\|}. \tag{10}$$

According to the selected curvature function the automatic fairing algorithm will evaluate all knot pairs.

Finally the sum of the curvature-slope discontinuities (local measures) from all knot pairs will define the *global fairing measure* ξ which will be minimized by the fairing algorithm:

$$\xi = \sum_{i,j} z_{ij}^u + z_{ij}^v. \tag{11}$$

Remark: If $z_{ij}^u > z_{ij}^v$, then the fairing step KRR will be applied in u-direction and vice-versa. The local measure (z_{ij}^u) or z_{ij}^v which contributes the most to the C^3-continuity at the knot (u_i, v_j) determines whether KRR is done in u- or v-direction.

(C) Visualization

An appropriate technique for emphasizing visually the fairing effect is a light reflection method. The fairing method reduces surface irregularities and provides a *more pleasing shape*. Those aesthetic aspects are well captured by isophotes, reflection line or highlight lines [26,21,1], because they are very sensitive to changes in the surface normals. Therefore we chose the isophote method.

(D) and (E) Scheduler and termination condition

The selection of the knot pair and the search for the global minimum of the fairing measure can be done in two ways.

Best-First-Search Fairing

The first method which is using the *best-first-search* strategy, tries to minimize the polynomial effort of a systematic search for the global minimum in the search tree. Here we begin with the initial surface on the root of the search tree an try to span only the interesting paths of the tree.

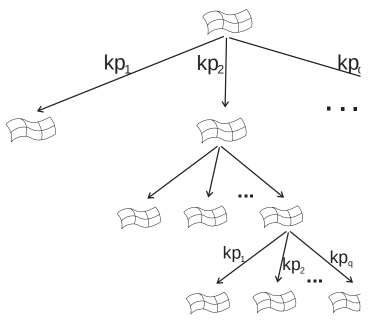


Figure 2: Possible search tree for the best-first-search algorithm

Starting with the initial surface the first level of the tree is spanned after performing KRR for each inner knot pair ($q = (n-3)(m-3)$ new surfaces are produced). All these surfaces are sorted in order to build a ranking list with respect to their increasing global fairness measure. The same operations are now repeated with only the first surface of the ranking list. The new q surfaces are also added into the ranking list. And so on ...

Under the assumption that the optimal surface (in the sense of the smallest global measure) is found when the KRR step is only performed on the first surface of the ranking list, the algorithm stops after a fixed number k (fixed before by the user) of fairing levels in the tree (see figure 2).

The resulting algorithm counts in the best case $O(k \cdot n \cdot m)$ fairing steps (i.e. is linear in the number of control points) and in the worst case $O((n \cdot m)^k)$ steps which is the same for a systematic search with k tree levels. For more details about the best-first-seach technique, see artificial intelligence literature, like [28].

Simulated Annealing Fairing

Simulated annealing is a heuristic method to solve large optimization problems. It ensures *"with high probability"* finding a global minimum of the function of interest, where iterative methods often get stuck at a local minimum. In 1983, IBM researchers [17,18] found an interesting analogy in material physics, where the heuristic process of an optimal annealing of metal is a quite difficult task.

Simulated annealing is a stepwise algorithm which is allowed to make *good steps* (i.e. steps which improve some measure of quality) and to make *bad steps* (those that do not). Those bad steps are necessary in order to leave a valley of a local minimum. The probability that bad steps are accepted decreases while the algorithm proceeds.

This algorithm can be compared with the *hill climbing model* : searching in this case the maximum of a function one can imagine that a hill climber is searching for the highest top. Depending of the departure point, he has to walk on tops (good steps) and valleys (bad steps) in order to reach the highest summit. Therefore he must not stop walking when he arrives at the first top. In this case he would only find a local maximum. Sometimes the climber ought to take a descending path. This helps to avoid a local maximum. In general the topography of the problem is not known in advance. Hence it is natural to use a probability for the acceptance of bad steps and in order to ensure that the algorithm stops.

In our fairing context, a function has to be *minimized* in contrast to the hill climbing model. The measure of quality is the global fairness measure ξ (11), which has to be minimized by a sequence of fairing steps, the knot-removal-reinsertion (KRR)-step. A topography of a simulated annealing optimization process is shown in figure 3.

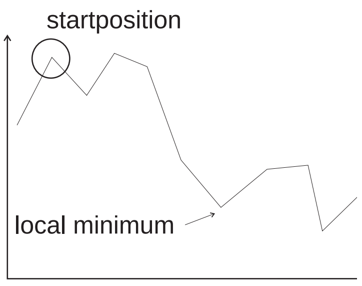
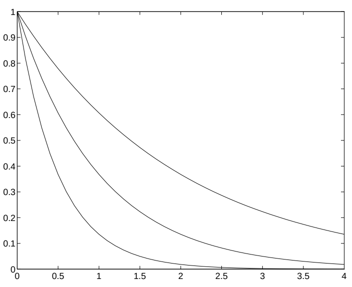
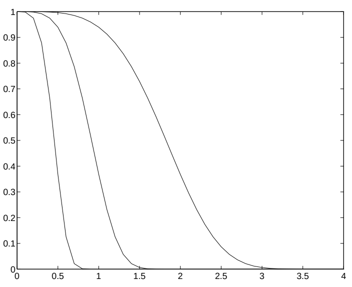


Figure 3: Topography of an optimization process

At each stage of the algorithm a knot pair (u_i, v_j) of the current surface is chosen either randomly or by a ranking list with respect to the local fairness measure $z^u_{ij} + z^v_{ij}$ at each knot pair. While the KRR-step increases ξ, the steps are accepted as good steps. Otherwise, a random number $\nu \in [0, 1]$ is chosen for the bad steps and is accepted with a certain probability. As the probability for bad steps gets smaller during the algorithm it gets less and less possible to leave the current minimum of the fairness measure and the algorithm stops there.



$$P(k) = e^{-\left(\frac{d\cdot k}{max\_kp}\right)}$$



$$P(k) = e^{-\left(\frac{d\cdot k}{max\_kp}\right)^4}$$

Figure 4: Probability of accepting bad steps (d = 0.5, 1, 2).

Let *max_kp* be the number of inner knot pairs. If the number of stages of the algorithm in controlled by the integer *max_step*, the probability function P at each stage t_k ($k = 1, \dots,$ *max_step*) should be a monotonously decreasing function. Some examples are shown in figure 4.

Algorithm: *(Simulated annealing for fairing surfaces)*

```
for $k = 1 \dots, end\_temp$ do
   do
      for $r = 1 \dots, max\_step$ do
          choose randomly a knot pair $kp$
          let $\boldsymbol{X}$ be the current surface, $\tilde{\boldsymbol{X}}$ the faired surface after KRR
          if $\xi_X > \xi_{\tilde{X}}$
          then make KRR; $\boldsymbol{X} := \tilde{\boldsymbol{X}}$; b_count = 1                /* good step
          else choose a random number $0 \leq \nu \leq 1$
               if $\nu < P(k)$
               then make KRR; $\boldsymbol{X} := \tilde{\boldsymbol{X}}$; b_count = 1           /* bad step
               else b_count += 1                                 /* bad step refused
      end for
   until (b_count $\geq$ blimit)                                 /* no more improvement
end for
```

The decreasing sequence of real numbers $t_1 > t_2 > \cdots > t_{max\_step}$ is called *annealing schedule* and controls the probability that bad steps will be done. The more they decrease, the less probable is a bad step. Once the annealing schedule is fixed, for example as a linear schedule $t_k = \frac{max\_kp}{k}$, an appropriate factor d in the probability function P has to be chosen. Two examples are shown in figure 4.

At each stage k the number of good steps is limited by *maxstep*, in order to avoid to make too many good steps at the beginning and to get stuck at a local minimum. A general suggestion we can make for the choice of the parameter *maxstep* is to take it half that of *max_kp*.

There are two possibilities for the algorithm to stop. In general the algorithm should stop when *blimit* times in succession the making of a bad step was refused (due to the decreasing probability of making bad steps). In this case there is no more way out of the current valley. The actual minimum is accepted as the optimal one. When the algorithm seems to be in an infinity loop, it stops after a maximal number of stages *end_temp* (final temperature). This should be an emergency stop, and a large enough *end_temp* has to be chosen to allow a regular stop.

The simulated annealing algorithm does not guarantee finding a global minimum, but it's well known that it's an effective way for searching one. This algorithm has f.ex. been used in CAGD by L. Schumaker [31] for computing optimal triangulations.

4. EXAMPLES

Before comparing the two fairing algorithms based on best-first-search and Simulated Annealing, we first want to illustrate the different fairing effects of the two KRR-strategies: least-squares KRR and minimal region KRR, see Figure 5.

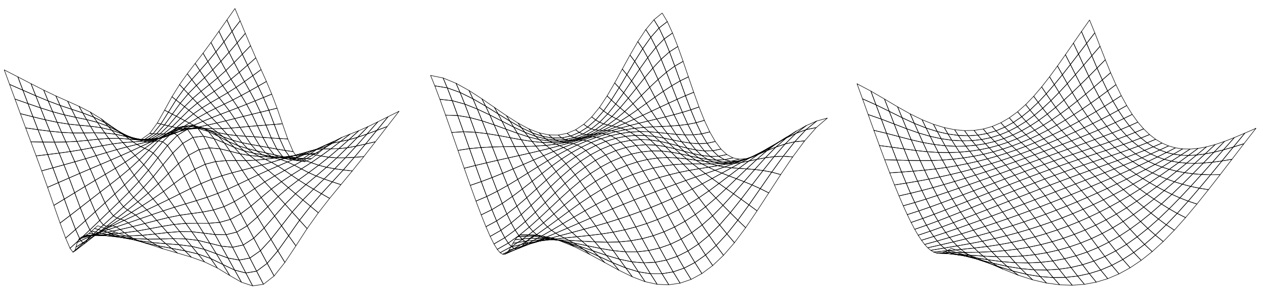


Figure 5: Fairing effect of the two different KRR-steps: (a) Initial surface. (b) Fairing with least-squares KRR. (c) Fairing with minimal-region KRR

Figure 5a shows an unfair surface which is **an academic example** with only 25 inner knots. One now expects from a fairing method that the high variations of curvature, which occur in the middle of the patch and on the boundary, become smaller or disappear. We then run the simulated annealing fairing (parameters: $g(u,v) := \kappa_{max}^2 + \kappa_{min}^2$ (10), $max\_step = blimit = 10$) once with the least-squares KRR-step (figure 5b) and once with the minimal-region KRR-step (figure 5c). The global fairness measure $\xi_{origin} = 1.3 \cdot 10^{-4}$ decreases to $\xi_{fair\_ls} = 2.1 \cdot 10^{-6}$ and $\xi_{fair\_mr} = 1.8 \cdot 10^{-7}$ in 32 and 71 steps. As it was assumed above, the ls-KRR is better adapted for shape preserving fairing, while a more flattening fairing effect is achieved with the mr-KRR.

The following two experiments are based on test surfaces close to the real CAD/CAM world because they represent noisy data sets (slightly (1% and 2%) perturbed data). From the definition of both search strategies it is clear, that they apply to different kinds of surfaces. The complexity of the best-first-search algorithm is linear (with respect to the number of inner knots) in the best case, otherwise it has polynomial complexity. If one claims from both algorithms to run in "reasonable" time (i.e. some cpu seconds) it is obvious that only the simulated-annealing-algorithm can run on surfaces with several (hundreds) of control points. Various experiments show that the Best-First-Search algorithm works better for surfaces with less than 50 control points, because it is nearer to a systematic search than the simulated-annealing algorithm. For both experiments $g = \kappa_{max} \cdot \kappa_{min}$ was used.

The **first example** is an application of the best-first-search (BFS) algorithm, shown in figure 6a,b. The unfair surface has 25 inner knots (i.e. 81 control points). The depth (maximum number of levels) of the BFS-tree is $k = 20$ in this experiment which results in a relatively high running time of 15s (IBM RS6000). k decides finally about the number of fairing steps and has to be chosen by the user. The total fairness measure decreased from $\xi_{origin} = 1.88 \cdot 10^{-6}$ to $\xi_{fair\_mr} = 1.72 \cdot 10^{-7}$ when applying the minimum-region KRR as fairing step (this example is shown in figure 6,7), and to $\xi_{fair\_ls} = 2.25 \cdot 10^{-7}$ when applying the least-squares KRR step.

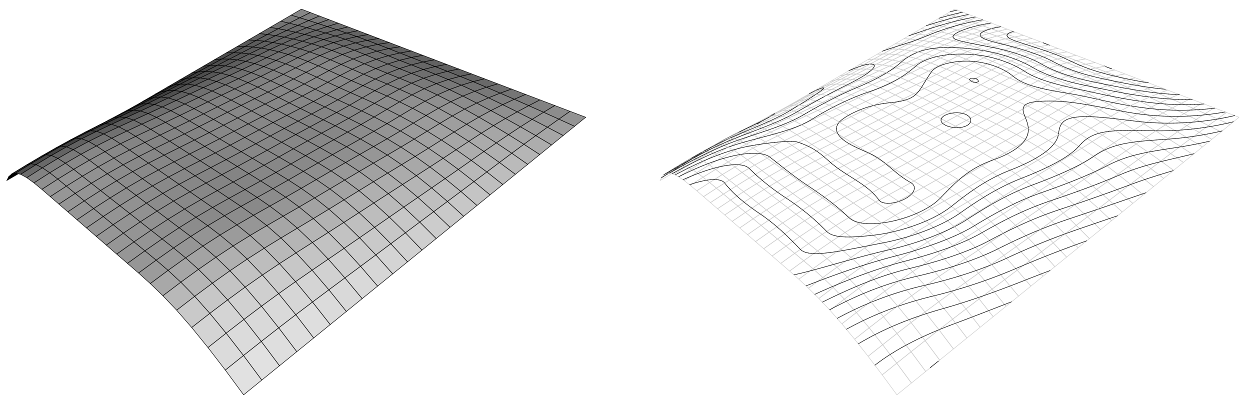


Figure 6: (a) Initial B-spline surface with 25 inner knot pairs. (b) Isophote analysis

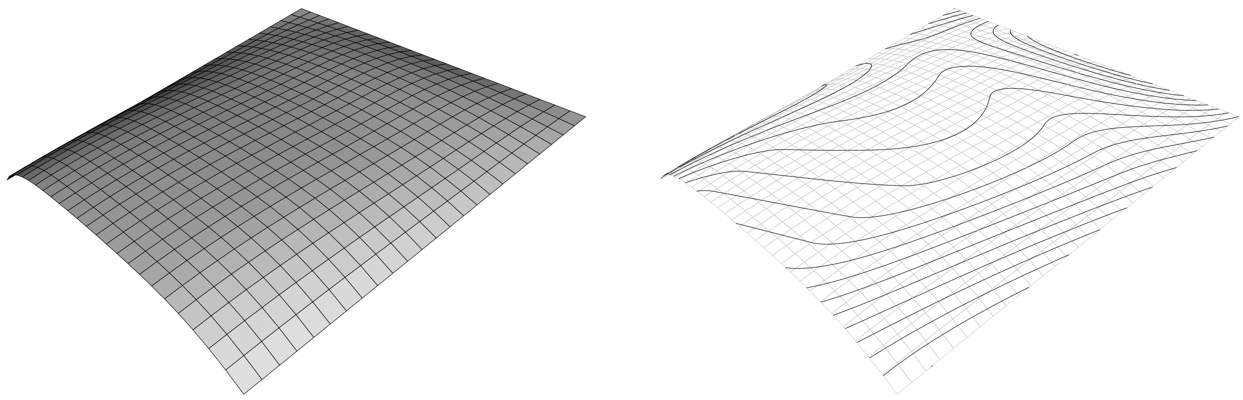


Figure 7:(a) The faired surface provided by the best-first-search algorithm. (b) Isophote analysis

We a posteriori measured the maximal distance

$$\tau_{max} = \frac{\max \|\boldsymbol{d}_{ij} - \boldsymbol{D}_{ij}\|}{\text{size of surface}}$$

(of corresponding control points) between the original and the faired surface of figure 7. After normalization we obtained $\tau_{max} = 0.013$. The lines mapped on the surface are isophotes (fig. 6b, 7b). They indicate the degree of fairness visually before and after fairing.

The **second numerical experiment** uses the simulated annealing algorithm. The initial (unfair) test surface has 256 inner knots, see figures 8a,b. This is an appropriate example for the simulated annealing algorithm, because it has more than 50 control points. The global fairness measure decreases from $\xi_{origin} = 3.92 \cdot 10^5$ to $\xi_{fair\_mr} = 1.59 \cdot 10^4$ for this surface using the mr-KRR step. The normalized maximal distance in this case gives $\tau_{max} = 0.037$. Simulated annealing is a non deterministic process, which means that several runs of the algorithms with always the same input have different output. This is usually a negative point of this algorithm. In the field of surface fairing it is an advantage, because fairness is basically a subjective criterion and depends on the designer's point of view. It can be observed that several runs of the simulated annealing algorithm always stop after ca. 400 steps with nearly the same result, i.e. a faired surface, like in figure 9a,b (running time 20s). Isophotes are used to visually control the fairing.

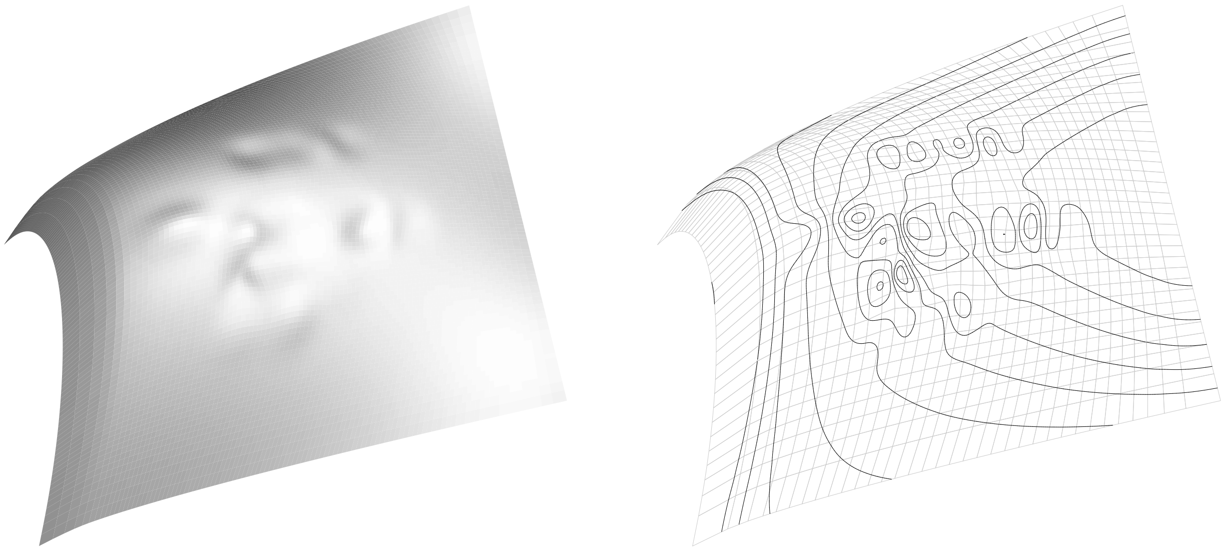


Figure 8: (a) Initial B-spline surface with 256 inner knot pairs. (b) Isophote analysis

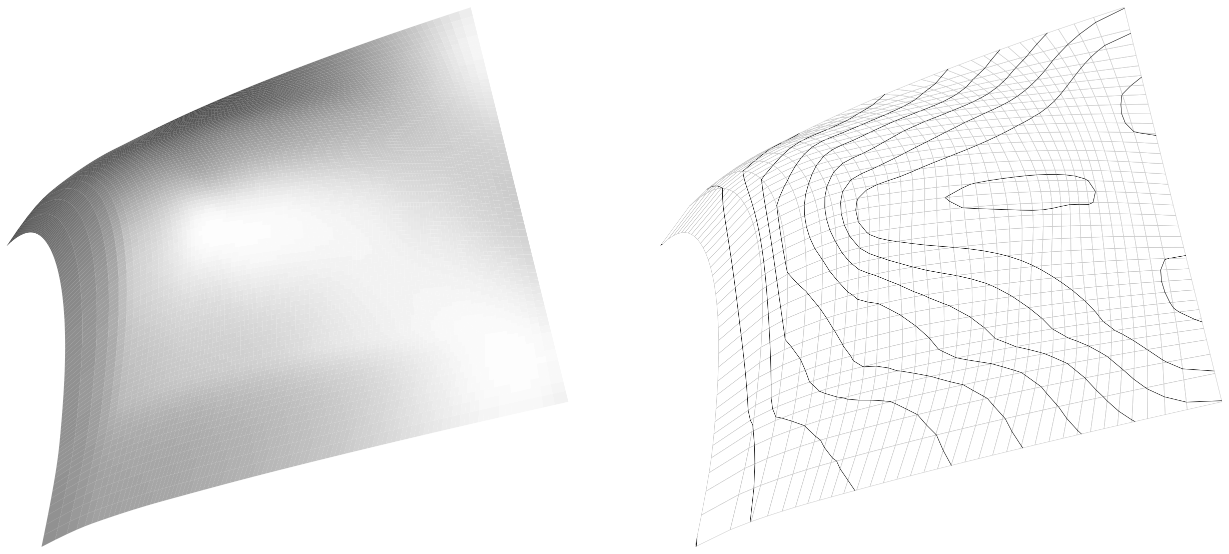


Figure 9: (a) The faired surface provided by the simulated annealing algorithm. (b) Isophote analysis

If the user is not satisfied by the result, the algorithm has to be run again. Perhaps he has to adjust the annealing parameters in order to increase or decrease the number of steps. In practice, the designer only needs a few runs to see which parameters fit well for the given surface. The running time of our prototype program varies between 2s and 20s, depending on the number of surface control points and the number of fairing steps. The example presented here in a certain sense a worst case with respect to running time.

In comparison to the BFS algorithm, 400 fairing steps for the simulated annealing algorithm are equivalent to only depth $k = 2$ in the BFS-tree for the second test surface. Remember that for one level of the BFS algorithm 256 fairing steps are evaluated and produce 256 different surfaces. Each of them was faired at one knot pair. Depth 2 means therefore that the final surface only differs by 2 KRR steps from the original one (only the first surface of the ranking list after 2 steps is retained). As a conclusion we can state that the heuristic simulated annealing fairing method is a very effective way to search for a minimum of the fairness measure when the initial surface has a lot of control points..
Finally notice that a systematic search for the global minimum of the fairness measure (after only 20 KRR-steps = 20 search tree levels) would have needed 256^{20}(!) KRR-steps.

CONCLUSION and REMARKS

Two fully automatic and locally working surface faring algorithms were developed in this paper. They are based on knot removal and knot insertion as fairing steps combined with two search strategies which search for the global minimum of the fairness function. Only the basic functionality of these algorithm was illustrated with the help of two examples.

It is important for fairing methods to preserve some shape properties and to keep in tolerances. With little programming effort we were able to improve the present algorithms in the following way:

Preservation of end constrains:
If all inner knots are involved in the fairing process, only the corner points of the surface are not modified. But the algorithm can be restricted to work only on a subset of knots. On one hand this limits the fairing to a local area on the surface. On the other hand, end constraints, like border curves and border derivative, can be preserved by excluding some rows and columns near the border from the fairing process.

Preservation of tolerances:
A given tolerance τ can be preserved *a posteriori* by allowing only such fairing steps which keep the new surface $\tilde{\boldsymbol{X}}(u,v)$ into the prescribed tolerance: $\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_{L^\infty} < \tau$. In practice the usual L^∞-norm can be approximated by the discrete $l^\infty, \boldsymbol{u}, \boldsymbol{v}$-norm on the control points, because the following estimation holds (for each component $c = 1, 2, 3$ of a parametric B-spline surface):

$$\|\boldsymbol{X}^c - \tilde{\boldsymbol{X}}^c\|_{L^\infty} \quad \leq \quad \|\boldsymbol{X}^c - \tilde{\boldsymbol{X}}^c\|_{l^\infty,\boldsymbol{u},\boldsymbol{v}} \quad = \quad \max_{i,j} |d_{ij}^c - \tilde{d}_{ij}^c| \,.$$

Further details about this estimation and its application to curve fairing with knot removal can be found in [23,25].

References

1. Beier K.-P., Highlight-line algorithm for real-time surface quality assessment, Computer Aided Design **26** (1994), 268–277.
2. Böhm W., Inserting new knots into B-spline curves, Computer Aided Design **12** (1980), 199–201.
3. Bonneau G.-P., and H. Hagen, Variational design of rational Bézier curves and surfaces, in *Curves and Surfaces in Geometric Design*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters, Wellesley MA, (1994), 51–58.
4. De Boor C., *A Practical Guide to Splines*, Springer, New York, 1978.
5. Brunet P., Increasing the smoothness of bicubic spline surfaces, Computer Aided Geometric Design **2** (1985), 157–164.
6. Eck M., and J. Hadenfeld, Knot removal for B-spline curves, Computer Aided Geometric Design **12** (1995), 259–282.
7. Farin G., G. Rein, N. Sapidis, and A. J. Worsey, Fairing cubic B-spline curves, Computer Aided Geometric Design **4** (1987), 91–103.
8. Farin G., and N. Sapidis, Curvature and the fairness of curves and surfaces, IEEE Computer Graphics & Applications **9** (1989), 53–57.

9. Ferguson D. R., P. D. Frank, and K. Jones, Surface shape control using constrained optimization on the B-spline representation, Computer Aided Geometric Design **5** (1988), 87–103.
10. Goldman R.N., and T. Lyche (eds.): *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*, SIAM (1993),
11. Greiner G., Variational design and fairing of spline surfaces, Proc. Eurographics 1994, 143–154.
12. Hadenfeld J., Local energy fairing of B-spline surfaces, in *Mathematical Methods for Curves and Surfaces*, Morten Dæhlen, Tom Lyche, Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville & London, 1995
13. Hagen H., and G. Schulze, Extremalprinzipien im Kurven- und Flächendesign, in *Verfahren der graphischen Datenverarbeitung*, (Encarnacao, J. Hoschek, J. Rix (eds.), Springer 1990, 46–60.
14. Hagen H., S. Hahmann, T. Schreiber, et al., Surface interrogation algorithms, IEEE Computer Graphics & Applications **12**(5) (1992), 53–60.
15. Hagen H., and P. Santarelli, Variational design of smooth B-spline surfaces, in *Topics in surface modeling*, . Hagen (ed.), SIAM (1992), 85–92.
16. Hahmann S., Visualization techniques for surface analysis, in *Data Visualization Techniques*, C. Bajaj (ed.), John Wiley, 1996.
17. Kirkpatrick S., C. D. Gelatt, M. P. Vecchi, Science **220** (1983), 671–.
18. Kirkpatrick S., Optimization by simulated annealing, J. Stat. Phys. **34** (5/6), (1984), 975-987.
19. Kjellander J. A., Smoothing of cubic parametric splines, Computer Aided Design **15** (1983), 175–179.
20. Kjellander J. A., Smoothing of bicubic parametric surfaces, Computer Aided Design **15** (1983), 289–293.
21. Klass R., Correction of local irregularities using reflection lines, Computer Aided Design **12** (1980), 73–77.
22. Lott N. J., and D. I. Pullin, Method for fairing B-spline surfaces, Computer Aided Design **20** (1988), 597–604.
23. Lyche T., and K. Morken, Knot removal for parametric B-spline curves and surfaces, Computer Aided Geometric Design **4** (1987), 217–230.
24. Moreton H. P., and C. H. Séquin, Functional optimization for fair surface design, Comp. Graph. **26** (1992), 167–176.
25. Pigounakis K. D., and P. D. Kaklis, Convexity-preserving fairing, Computer Aided Design **28** (1996), 981–994.
26. Poeschl T., Detecting surface irregularities using isophotes, Computer Aided Geometric Design **1** (1984), 163–168.
27. Rando T., and J. A. Roulier, Designing faired parametric surfaces, Computer Aided Design **23** (1991), 492–497.

28. Richter M. M., Prinzipien der künstlichen Intelligenz, B. G. Teubner, Stuttgart 1989.
29. Sapidis N., and G. Farin, Automatic fairing algorithm for B-spline curves, Computer Aided Design **22** (1990), 121–129.
30. Schumaker L. L., *Spline Functions: Basic Theory*, Wiley, New York, 1981.
31. Schumaker L. L., Computing optimal triangulations using simulated annealing, Computer Aided Geometric Design **10** (1993), 329–345.