Meshless Parameterization and B-spline Surface Approximation

Michael S. Floater[1]

SINTEF, P.O. Box 124, Blindern, 0314 Oslo, Norway

Summary. This paper proposes a method for approximating unorganized points in $\mathbb{R}^3$ with smooth B-spline surfaces. The method involves: meshless parameterization; triangulation; shape-preserving reparameterization; and least squares spline approximation.

1 Introduction

The goal of this paper is to describe a new method for approximating a set of distinct points

$$X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \qquad \boldsymbol{x}_i \in \mathbb{R}^3, \tag{1}$$

with a tensor-product B-spline surface

$$\boldsymbol{s}(u,v) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} B_i(u) C_j(v) \boldsymbol{c}_{ij}, \qquad \boldsymbol{c}_{ij} \in \mathbb{R}^3. \tag{2}$$

Here $B_1, \ldots, B_{m_1}$ and $C_1, \ldots, C_{m_2}$ are B-splines over non-uniform knot vectors with orders (degrees plus 1) K and L respectively. We assume that the points have been sampled from a (simply connected) patch of the surface of some object in $\mathbb{R}^3$. Many sets of measured data in practice are in the form of single patches, even though their geometry can be quite complex. The surface generation method we propose consists of several sequential steps: meshless parameterization; triangulation; reparameterization; and least squares approximation. The key ingredient is the meshless parameterization of [11] which we use to parameterize the points $\boldsymbol{x}_i$ without the need of a given mesh or topological structure. The overall surface approximation method has performed very well in numerical examples, at least when the underlying surface is not too far from being developable.

Surface generation from 'single patch' data sets can be viewed as a simple form of reverse engineering, which, in its widest sense, is usually understood to be the generation of a full B-rep (boundary representation) surface model from points measured from the whole surface of a physical object in $\mathbb{R}^3$. Since such surfaces are closed and can have arbitrary topology, building a B-rep model requires not only topology construction but also segmentation and the sewing together of surface patches. Thus reverse engineering in general is

a non-trivial operation and considerable research effort has been invested in developing automatic and semi-automatic methods; for a survey of techniques up to 1997, see [22].

Single-patch data is simpler than general point data in the sense that it offers the possibility of constructing a *parameterization*

$$U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}, \qquad \boldsymbol{u}_i \in \mathbb{R}^2, \tag{3}$$

a set of points in the plane corresponding to the data points $\boldsymbol{x}_i$. Any standard scattered method [17] can then be applied to find control points $\boldsymbol{c}_{ij}$ in (2) which in some sense minimize the error vectors

$$\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{x}_i. \tag{4}$$

We will minimize these vectors in a least squares sense, drawing on the work of several authors. A major part of this paper however, concerns the construction of a good parameterization. Certainly a simple parameterization can be generated if the data set X is sampled from a surface patch which can be projected 1-1 onto some *base surface*, a simple parametric surface such as a plane, sphere, or cylinder. One can then simply take the projections of the points $\boldsymbol{x}_i$ onto this base surface as parameter points $\boldsymbol{u}_i$. However, in the absence of a base surface, the only published parameterization method we know of is that of Hoschek and Dietz [16]. The main idea of their approach is to construct the parameterization and the surface simultaneously, and iteratively. The initial parameterization is formed by projecting the points $\boldsymbol{x}_i$ onto some least squares fitting plane: a mapping which may or may not be 1-1. Provided the iteration converges, the method generally leads to very well parameterized surfaces with small error. This is partly due to finding the parameterization through *parameter correction* [17] which aims to yield error vectors in (4) which are perpendicular to the surface. However, it has been pointed out in [16], [5], and [6] that when the geometry of the data set is complex, the method sometimes fails completely, due to foldover in the parameterization.

The method we propose here provides an alternative approach which generates a parameterization in which the parameter point of each data point is forced to lie in the convex hull of the parameter points of neighbouring data points. Though we are not at this stage able to offer any theoretical justification, all the parameterizations in our (numerous) test examples have been free of the foldover effect exhibited by the method of [17].

The basic structure of the paper is built around the three main steps of the surface approximation algorithm: meshless parameterization and triangulation in Section 2, shape-preserving reparameterization in Section 3, and least squares spline approximation in Section 4. Section 5 contains the results of applying the algorithm to a variety of test examples.

2 Meshless Parameterization and Triangulation

Our first step consists of *meshless parameterization*, recently introduced in [11]. Meshless parameterization determines a sequence of parameter points $\boldsymbol{u}_i$ from the points $\boldsymbol{x}_i$ *without* the need for any given topological structure. This almost immediately gives us a triangulation $\mathcal{T}$ of the points set X: we simply compute a Delaunay triangulation $\mathcal{S}$ of the (planar) parameter points $\boldsymbol{u}_i$ and define $\mathcal{T}$ to be the corresponding (surface) triangulation of the $\boldsymbol{x}_i$. In other words, we take $\mathcal{T}$ to be the set of triangles $[\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k]$ for which $[\boldsymbol{u}_i, \boldsymbol{u}_j, \boldsymbol{u}_k]$ is a triangle in $\mathcal{S}$.

Notice here that the emphasis is not on choosing the parameter points $\boldsymbol{u}_i$ to yield an optimal triangulation $\mathcal{T}$ for the $\boldsymbol{x}_i$ in some sense. This is because later, in Section 3, shape-preserving reparameterization will be used to make a better parameterization of X by using the topological structure of $\mathcal{T}$. For example, it is not critical if $\mathcal{T}$ contains unnecessarily long thin triangles in some regions, as long as the shape of $\mathcal{T}$ roughly mimics the shape of X. On the other hand the parameter points $\boldsymbol{u}_i$ should at least have the property that the resulting triangulation $\mathcal{T}$ is free of self-intersections. Currently we are not able to provide a condition which will ensure this, though we have never experienced this in any of our test cases.

Triangulation by meshless parameterization is relatively fast to both implement and compute compared with existing methods for triangulating unorganized points sets, though it is only applicable to single-patch data sets. Other triangulation methods include those of [3] and [18], which are based on the idea of successively removing tetrahedra from a Delaunay tetrahedrization of the points, and those of [21] and [1], which are based on properties of the Voronoi diagram. There are also the implicit methods of [14] and [2], which only approximate the data points.

How do we determine the set of parameter points U without needing a mesh? We first divide the set X into two disjoint subsets: X_I, the set of interior points, and X_B, the set of boundary points. Moreover the boundary points must be ordered. A method was outlined in [11] for first identifying boundary points and subsequently ordering them using a univariate analog of meshless parameterization. Without loss of generality we thus assume that $X_I = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_n\}$ for some n, and $X_B = \{\boldsymbol{x}_{n+1} \ldots, \boldsymbol{x}_N\}$, where the points $\boldsymbol{x}_{n+1} \ldots, \boldsymbol{x}_N$ are ordered consecutively along the boundary.

The method has two steps. In the first step we map the boundary points $\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_N$ into the boundary of some convex polygon D in the plane. Thus we choose the corresponding parameter points $\boldsymbol{u}_{n+1}, \ldots, \boldsymbol{u}_N$ to lie around ∂D in some anticlockwise order. In our numerical examples we take $\boldsymbol{u}_{n+1}, \ldots, \boldsymbol{u}_N$ to lie either on an ellipse or a rectangle and we place the points around these boundaries according to chord length.

In the second step, we choose for each interior point $\boldsymbol{x}_i \in X_I$, a *neighbourhood* N_i, a set of indexes of points $\boldsymbol{x}_j$ which are in some sense close by.

We also choose a set of (strictly) positive weights λ_{ij}, for $j \in N_i$, such that

$$\sum_{j \in N_i} \lambda_{ij} = 1.$$

Then, in order to find the n parameter points $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^2$ corresponding to the interior points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^3$, we solve the linear system of n equations

$$\boldsymbol{u}_i = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{u}_j. \qquad i = 1, \ldots, n. \tag{5}$$

These equations demand that each interior $\boldsymbol{u}_i$ be some convex combination of its neighbours $\{\boldsymbol{u}_j : j \in N_i\}$. Thus $\boldsymbol{u}_i$ will be contained in the convex hull of the neighbours.

It was established in [11] that the linear system (5) is nonsingular under a very mild condition, namely that every interior point $\boldsymbol{x}_i$ is *boundary connected*. By boundary connected we mean that $\boldsymbol{x}_i$ can be joined to the boundary X_B by a path of points in X where each point $\boldsymbol{x}_{j+1}$ in the path is a neighbour of the previous point $\boldsymbol{x}_j$, in the sense that $\boldsymbol{x}_{j+1} \in N_j$. This condition will be fulfilled if the neighbourhoods N_i are chosen large enough. In fact, in practice one can usually choose the neighbourhoods to be both large enough that every interior point is connected to *every* boundary point, and at the same time small enough that each point neighbourhood $\{\boldsymbol{x}_j : j \in N_i\}$ consists of points which are nearby with respect to the geometry of the underlying surface. Certainly we do not want to pick up points from extraneous branches of the surface. It was shown in [11] that due to the convex combinations and the fact that the domain D is convex, all parameter points will lie inside D.

Several choices of neighbourhood N_i were proposed in [11] but a simple and effective choice is to take the 'd nearest neighbours', in other words, we let N_i be the set of indexes of the d points $\boldsymbol{x}_j$ closest to $\boldsymbol{x}_i$. Going on several numerical tests, it appears that setting $d = 10$ or $d = 20$ is adequate for all but the most extreme data sets. As regards the choice of weights λ_{ij}, the naive choice of uniform weights $\lambda_{ij} = 1/d_i$, where $d_i = |N_i|$, can lead to the result that two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ end up being mapped to the same points: $\boldsymbol{u}_i = \boldsymbol{u}_j$, as shown in [11]. Our preferred choice in practice is to use the *reciprocal distance weights*

$$\lambda_{ij} = \frac{1}{||\boldsymbol{x}_j - \boldsymbol{x}_i||} \Big/ \sum_{k \in N_i} \frac{1}{||\boldsymbol{x}_k - \boldsymbol{x}_i||},$$

which depend on the distances between $\boldsymbol{x}_i$ and its neighbours. These have always resulted in distinct parameter points in all the numerical examples we have run. The theoretical best choice of weights still requires further research. Since the matrix A arising from the linear system (5) is sparse and in general non-symmetric, we have used the biconjugate gradient method to solve (5).

3 Shape-preserving Reparameterization

The goal of the meshless parameterization used in the previous section was to generate a (possibly rough) triangulation $\mathcal{T}$ of the data set X so that we have some topological structure. With the triangulation $\mathcal{T}$ in place, we can reparameterize the interior points $\boldsymbol{x}_i$, but this time use the neighbourhood information given by $\mathcal{T}$. Methods for doing this have been proposed in [7], [8], [12], [20] and we will apply the *shape-preserving parameterization* of [8]. This parameterization has the advantage that it has linear precision and so the triangles in the mapped triangulation tend to mimic the shape of the triangles in $\mathcal{T}$: hence the name. This parameterization also tends to lead to surface approximations $\boldsymbol{s}(u, v)$ in (2) with smooth isocurves. We note that it was shown in [9] that the shape-preserving parameterization of [8] is very similar, both empirically and theoretically, to the harmonic map of [7]. Indeed the harmonic map also has linear precision [9]. However, unlike the shape-preserving parameterization of [8], the harmonic map can fold over, due to negative weights; an example is given in [9].

To help in understanding the shape-preserving parameterization for triangulations in [8], let us first consider the analogous *chord length* parameterization for polygonal curves. If the vertices of the curve are $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^3$, then the sequence of real values $t_1, t_2, \ldots, t_N$ is called a *chord length* parameterization if

$$(t_{i+1} - t_i) = \rho ||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i||,$$

for some constant $\rho > 0$, where $||\cdot||$ is the Euclidean norm. It was observed in [8] that the sequence $t_1, t_2, \ldots, t_N$ is a chord length parameterization if and only if

$$t_i = \frac{||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|| t_{i-1} + ||\boldsymbol{x}_i - \boldsymbol{x}_{i-1}|| t_{i+1}}{||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|| + ||\boldsymbol{x}_i - \boldsymbol{x}_{i-1}||}.$$

This implies that t_i is a convex combination of its two neighbouring parameter values t_{i-1} and t_{i+1}, and the weighting depends on the lengths of the chords, in such a way that if the points $\boldsymbol{x}_i$ lie on a straight line (though not necessarily uniformly), then the whole sequence $t_1, \ldots, t_N$ will be some affine transformation, from $\mathbb{R}^3$ to $\mathbb{R}^1$, of the sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. In other words, chord length parameterization has linear precision and it is this property that is carried over to the shape-preserving parameterization for triangulations in [8].

In the shape-preserving parameterization for triangulations, we solve the system (5) where we take the neighbourhoods N_i to be the neighbourhoods of the triangulation $\mathcal{T}$. In addition we choose positive weights λ_{ij} which give linear precision.

For the sake of completeness, we will outline how the weights are determined. For each i, we refer to the set of triangles incident on $\boldsymbol{x}_i$ as the *cell* of $\boldsymbol{x}_i$. The vertices in the cell are $\boldsymbol{x}_i$ and its neighbours $\{\boldsymbol{x}_j : j \in N_i\}$. The

shape-preserving weights λ_{ij} depend only on $\boldsymbol{x}_i$ and its neighbours and are constructed in two steps.

The first step is to 'flatten out' the cell into the plane, yielding *local* (temporary) parameter points $\boldsymbol{u}_i$ and $\{\boldsymbol{u}_j : j \in N_i\}$ in $\mathbb{R}^2$; see Figure 1. We use an approximation of the geodesic polar map, adapted to triangulations, which, it seems, was first proposed in [23], in the context of computer graphics. We let $\boldsymbol{u}_i$ be arbitrary and choose the neighbours $\boldsymbol{u}_j$ such that for each $j \in N_i$,

$$||\boldsymbol{u}_j - \boldsymbol{u}_i|| = ||\boldsymbol{x}_j - \boldsymbol{x}_i||$$

and for each triangle $[\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k]$ in the cell of $\boldsymbol{x}_i$,

$$\text{ang}(\boldsymbol{u}_k, \boldsymbol{u}_i, \boldsymbol{u}_j) = \rho\, \text{ang}(\boldsymbol{x}_k, \boldsymbol{x}_i, \boldsymbol{x}_j),$$

where ρ is a constant. The scaling factor ρ is needed to ensure that the interior angles in the mapped cell sum to 2π.

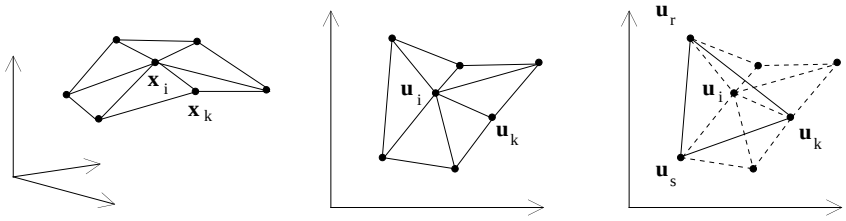

Fig. 1. Calculating the shape-preserving weights

The second step is to express $\boldsymbol{u}_i$ as a convex combination of the neighbouring mapped points $\{\boldsymbol{u}_j : j \in N_i\}$, in order to obtain linear precision. This is always possible since the mapped cell is star-shaped with $\boldsymbol{u}_i$ in its kernel. For each $k \in N_i$, we locate an edge $[\boldsymbol{u}_r, \boldsymbol{u}_s]$, $r, s \in N_i$, in the mapped cell for which

$$\boldsymbol{u}_i \in [\boldsymbol{u}_k, \boldsymbol{u}_r, \boldsymbol{u}_s],$$

and with τ_k^k, τ_r^k, τ_s^k the barycentric coordinates of $\boldsymbol{u}_i$ in this latter triangle, we have

$$\boldsymbol{u}_i = \tau_k^k \boldsymbol{u}_k + \tau_r^k \boldsymbol{u}_r + \tau_s^k \boldsymbol{u}_s.$$

Letting $\tau_j^k = 0$ for all $j \in N_i$, $j \neq k, r, s$ we then have

$$\boldsymbol{u}_i = \sum_{j \in N_i} \tau_j^k \boldsymbol{u}_j.$$

Finally, we take the shape-preserving weights to be averages of the local weights τ_j^k over all $k \in N_i$,

$$\lambda_{ij} = \frac{1}{d} \sum_{k \in N_i} \tau_j^k,$$

and we have

$$\boldsymbol{u}_i = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{u}_j, \qquad \text{and} \qquad \sum_{j \in N_i} \lambda_{ij} = 1,$$

and $\lambda_{ij} > 0$ for all $j \in N_i$.

4 Least Squares Approximation

Finally, we propose a least squares method for approximating the parameterized data by a tensor-product spline surface (2). Our discussion builds on the theoretical studies of [13] and [19] and follows closely the work of [16], [5], [6], [12], [10]. Least squares approximation of scattered data by tensor-product splines with a smoothing term seems to yield very well behaved, smooth surfaces even when the data set is very large and is certainly an attractive alternative to other scattered data methods such as piecewise polynomials over triangulations or radial basis functions.

Our goal is to determine the control points $\boldsymbol{c}_{ij}$ in (2) to minimize the sum of squared errors

$$\sum_{k=1}^{N} ||\boldsymbol{s}(\boldsymbol{u}_k) - \boldsymbol{x}_k||^2. \tag{6}$$

We assume here that we have already chosen the degrees K and L of the B-splines in (2). We must also choose the two knot vectors and we ensure that the rectangular domain $[a_1, b_1] \times [a_2, b_2]$ of $\boldsymbol{s}$ contains all the parameter points $\boldsymbol{u}_i$. If the parameter domain D of the parameter points $\boldsymbol{u}_i$ is chosen to be a rectangle then the knot vectors are chosen so that

$$D = [a_1, b_1] \times [a_2, b_2],$$

and the boundary of the resulting surface approximation $\boldsymbol{s}$ will closely follow the boundary points X_B. If on the other hand, the domain D has a different shape, for example a circle or ellipse, we choose the knot vectors so that

$$D \subset [a_1, b_1] \times [a_2, b_2].$$

The resulting surface approximation $\boldsymbol{s}$ will then need to be trimmed so that the final surface approximation will be a trimmed B-spline surface. One should also choose non-uniform knot vectors which reflect the distributions of the points $\boldsymbol{u}_k$ along each of their two coordinate axes.

It remains to find the control points $\boldsymbol{c}_{ij}$. One could try to minimize the function (6) directly but since the $\boldsymbol{u}_i$ are unstructured, the minimum will rarely be unique and the usual procedure is to add a smoothing term. There are several smoothing terms one can add such as ones defined in terms of curvature, torsion, etc. Several approaches, linear and nonlinear, are mentioned in [5] and [12]. We have found in practice that provided the shape-preserving

parameterization is used to compute the $\boldsymbol{u}_i$, good surface approximations result from simply adding the thin-plate spline energy term. Specifically we propose minimizing the function

$$F = \sum_{k=1}^{N} ||\boldsymbol{s}(\boldsymbol{u}_k) - \boldsymbol{x}_k||^2 + \lambda J, \tag{7}$$

where J is the thin-plate spline integral

$$J = \int_{a_1}^{b_1} \int_{a_2}^{b_2} ||\boldsymbol{s}_{uu}||^2 + 2||\boldsymbol{s}_{uv}||^2 + ||\boldsymbol{s}_{vv}||^2 \, dv \, du, \tag{8}$$

and $\lambda > 0$ is a constant.

We will discuss some computational details of how one minimizes F and the first thing to notice is that the minimization 'decouples' in the sense that since

$$F = \sum_{\alpha=1}^{3} \left(\sum_{k=1}^{N} (s^\alpha(u_k) - x_k^\alpha)^2 + \lambda J^\alpha \right),$$

it is minimized by the independent minimization of its three component functions. Thus for the sake of simplicity, we next replace the points $\boldsymbol{x}_i$ by real values x_i and the control points $\boldsymbol{c}_{ij}$ by real coefficients c_{ij}. The vector valued spline surface $\boldsymbol{s}$ is now replaced by the real valued spline function s.

Our simplified goal is to determine a coefficient vector

$$c = (c_{1,1}, \ldots, c_{m_1,1}, c_{1,2}, \ldots, c_{m_1,m_2})^T$$

of length $m = m_1 m_2$, which minimizes the function

$$F(c) = \sum_{k=1}^{N} (s(\boldsymbol{u}_k) - x_k)^2 + \lambda J(c), \tag{9}$$

where J is the thin plate spline integral

$$J(c) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} s_{uu}^2 + 2s_{uv}^2 + s_{vv}^2 \, dv \, du.$$

This integral can be re-expressed as

$$\sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \sum_{r=1}^{m_1} \sum_{s=1}^{m_2} E_{ijrs} c_{ij} c_{rs},$$

where

$$E_{ijrs} = A_{ijrs} + 2B_{ijrs} + C_{ijrs}$$

and

$$A_{ijrs} = \int_{a_1}^{b_1} B_i''(u) B_r''(u) du \int_{a_2}^{b_2} C_j(v) C_s(v) dv,$$

$$B_{ijrs} = \int_{a_1}^{b_1} B_i'(u)B_r'(u)du \int_{a_2}^{b_2} C_j'(v)C_s'(v)dv,$$

$$C_{ijrs} = \int_{a_1}^{b_1} B_i(u)B_r(u)du \int_{a_2}^{b_2} C_j''(v)C_s''(v)dv.$$

A minimum of $F(c)$ must occur at a point c where all partial derivatives are zero, a *critical point*. The equations $\partial F/\partial c_i = 0$ are called the *normal equations* of the least squares problem. It will help to write

$$J(c) = c^T Ec,$$

where E is the $m \times m$ matrix whose elements are

$$E_{(j-1)m_1+i,\,(s-1)m_1+r} = E_{ijrs},$$

for $i, r = 1, \ldots, m_1$ and $j, s = 1, \ldots, m_2$. By differentiating $F(c)$ in (9) explicitly and rearranging the subsequent expression, the normal equations can thus be rewritten as the single matrix equation

$$(B^T B + \lambda E)c = B^T x, \tag{10}$$

where $x = (x_1, \ldots, x_N)^T$ and B is the $N \times m$ matrix

$$B = \begin{pmatrix} B_1(u_1)C_1(v_1) & B_2(u_1)C_1(v_1) & \ldots & B_{m_1}(u_1)C_{m_2}(v_1) \\ \vdots & \vdots & & \vdots \\ B_1(u_N)C_1(v_N) & B_2(u_N)C_1(v_N) & \ldots & B_{m_1}(u_N)C_{m_2}(v_N) \end{pmatrix},$$

where $\boldsymbol{u}_k = (u_k, v_k)$. Then the solution to minimizing (9) is the solution c to (10). As observed in [13], the $m \times m$ matrix

$$G = B^T B,$$

whose elements are

$$G_{(j-1)m_1+i,\,(s-1)m_1+r} = \sum_{k=1}^{N} B_i(u_k)C_j(v_k)B_r(u_k)C_s(v_k) \tag{11}$$

for $i, r = 1, \ldots, m_1$ and $j, s = 1, \ldots, m_2$, is symmetric and positive semidefinite. So is E and thus the matrix sum

$$A = G + \lambda E \tag{12}$$

is also symmetric and positive semidefinite. We will derive a sufficient condition for when A is *strictly* positive definite and thereby nonsingular. The matrix A is strictly positive definite if the only solution to $c^T Ac = 0$ is $c = 0$. First observe that

$$c^T Ec = J(c) = 0$$

implies that s must be a linear polynomial $a + bu + cv$. Second, observe that

$$c^T G c = c^T B^T B c = ||Bc||^2 = 0$$

implies that $s(\boldsymbol{u}_k) = 0$ for all $k = 1, \ldots, N$. Thus we have that $c^T A c = 0$ implies that s is a linear polynomial which is zero at every parameter point $\boldsymbol{u}_k$. Clearly then, if there are at least three points $\boldsymbol{u}_k$ which do not lie on a straight line, s would have to be zero and therefore all the coefficients c_{ij} would also have to be zero. Since, due to our parameterization method, the parameter points $\boldsymbol{u}_k$ can never all be collinear, we deduce that A is indeed nonsingular and the minimizer c of (10) is unique.

In order to compute the elements of the matrix A in (12), we have used the Cox de Boor algorithm for the evaluation of B-splines and their derivatives, and the algorithm in [4] for the exact evaluation of inner products of B-splines. When building A, it is very important not to compute each element of G independently, for this can be terribly costly when N is large. The simple trick is to instead process each parameter point $\boldsymbol{u}_k$ in turn and compute *all* the tensor-product B-splines whose supports contain it, applying the Cox de Boor algorithm just once. For each new pair (u_k, v_k), all non-zero products

$$B_i(u_k) C_j(v_k) B_r(u_k) C_s(v_k)$$

are added to the current value of $G_{(j-1)m_1+i,\,(s-1)m_1+r}$.

The matrices G and E and A are clearly sparse. If the B-splines B_i and C_j have orders K and L respectively, then

$$A_{(j-1)m_1+i,\,(s-1)m_1+r} = 0$$

if either $|i - r| \geq K$ or $|j - s| \geq L$. The non-zero elements of A are shown in Figure 2 for the case when $m_1 = m_2 = 10$ and the spline $s(u, v)$ is bicubic ($K = L = 4$). As illustrated by the figure, the matrix A has $m_2 \times m_2$ blocks, each of which has size $m_1 \times m_1$, and is banded with bandwidth $2K - 1$. If A is viewed as an $m_2 \times m_2$ block matrix then it has bandwidth $2L - 1$. The real bandwidth of A however is $(2L - 1)m_1$.

We have chosen to use the conjugate gradient method to solve each of the two components of (10), which capitalizes on the sparse form of A. The number of non-zero elements of A in any row is at most $(2K - 1)(2L - 1)$ and is independent of m_1 and m_2. For example, in Figure 2, the maximum number of non-zeros per row is 49.

The final point about the least squares is the choice of the smoothing parameter λ in (7). This can be used to vary the emphasis of the approximation between error minimization and smoothing. However, it is useful in practice to a have a good default value. In the absence of any better estimation, we suggest the simple and pragmatic choice of

$$\lambda = ||G||/||E||,$$

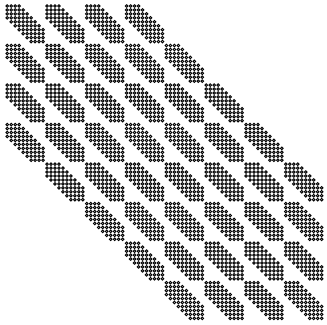

Fig. 2. Matrix structure when $K = L = 4$, $m_1 = 10$, and $m_2 = 8$

for some matrix norm $||.||$, such as the l_2 norm $||M|| = \sqrt{(\sum_{ij} m_{ij}^2)}$. The effect of this choice of λ is roughly speaking to ensure that the two contributions G and λE to the matrix A in (12) have equal weight. This choice has performed very well in our test examples.

After the surface $\boldsymbol{s}$ has been computed, the error of the approximation can be computed numerically at each point and if the error is unacceptable with respect to some error measure, the knot vectors of the surface can be refined and a new surface computed, keeping the parameterization fixed. Eventually the error will be within a chosen tolerance.

5 Numerical Examples

The first numerical example illustrates all the steps of the whole algorithm from unorganized points to B-spline surface. Figure 3a shows 192 unorganized points for which a boundary is self-evident. Figure 3b shows its meshless parameterization where the points are mapped into the rectangular domain $D = [0, 2] \times [0, 1]$. The shape of the domain roughly mimics the shape of the point set X. Figure 3c shows the Delaunay triangulation of the parameter points in Figure 3b and Figure 3d shows the corresponding triangulation $\mathcal{T}$ of the original points. Since the points in Figure 3b provide only a rough parameterization, lacking for example linear precision, we then make a shape-preserving reparameterization of the points x_i, which uses the neighbourhood information of the triangulation in Figure 3d. The new parameterization, with the corresponding triangulation, is shown in Figure 3e and is used to compute the least squares surface approximation of Figure 3f. The spline surface is bicubic ($K = L = 4$) and has dimensions $m_1 = m_2 = 10$.

For the sake of comparison, we also show in Figure 3g the Delaunay retriangulation of the final parameter points of Figure 3e and one sees how the corresponding retriangulation $\mathcal{T}'$ of X in Figure 3h has better proportioned triangles than the earlier one, $\mathcal{T}$, in Figure 3d. This would be useful if one simply wanted to triangulate the original data set.

The second example illustrates how well the method performs equally well on data for which there is no obvious base surface to project onto (in which case other methods could be applied). Following the shape of the boundary, the 1800 points of the 'S-shape in Figure 4a were also mapped into a rectangle and following the same steps as for the previous example, the triangulation $\mathcal{T}'$ of the points is shown in Figure 4b (corresponding to Figure 3h). The spline approximation is shown in Figure 4c, together with the corners of the polynomial patches. The surface is again bicubic.

Figure 5a shows a real data set of 24712 points from a sofa. Rather than try to mimic the complex boundary shape with a rectangular parameter domain, we chose here to map the boundary into the unit circle. After making a shape-preserving reparameterization, the resulting triangulation $\mathcal{T}'$ is shown in Figure 5b. Figure 5c shows a bicubic spline surface approximation of dimensions $m_1 = m_2 = 100$ which captures the crease in the back of the sofa. This surface would need to be trimmed by the circle in its parameter domain for it to hug the original data set properly. Note here that in a real reverse engineering application one would probably want to segment the data set according to the 'folds' in the sofa.

The last example illustrates the limitation of our surface approximation technique when the data set is far from being developable. In contrast to the S-shape of Figure 2a, the Spock data set of Figure 6a requires considerable deformation to map into the plane. After mapping it into the unit square and reparameterizing, the distance between the closest pair of parameter points was found to be just 0.0004. This poses no obvious problem for the corresponding triangulation $\mathcal{T}'$ of the original points in Figure 6b, but the 100×100 bicubic least squares approximation in Figure 6c exhibits undesirable oscillations, for example, around the mouth. Here the two knot vectors were constructed so that there are many densely placed knot lines, in both directions, in the middle of the domain, and fewer elsewhere. This was necessary in order to have sufficient degrees of freedom to reproduce the detail at the top of the head and the face. On the other hand these dense knot lines create the unwanted oscillations lower down the head. Despite this, the smooth rows of patch corners in Figure 6d clearly indicate that the parameterization is good. As pointed out in [15], the inevitably high deformation when parameterizing a data set such as the head of Spock suggests that approximating with some kind of multiresolution surface, as in [12], would give better results.

Fig. 3a. Point set.

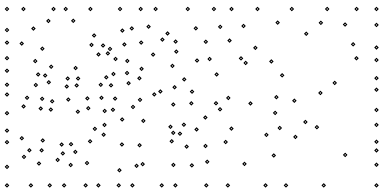

Fig. 3b. Meshless parameterization.

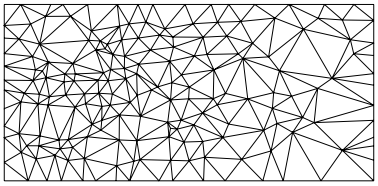

Fig. 3c. Delaunay triangulation.

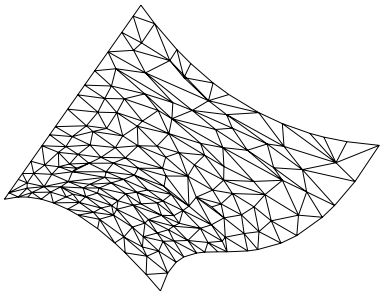

Fig. 3d. Surface triangulation.

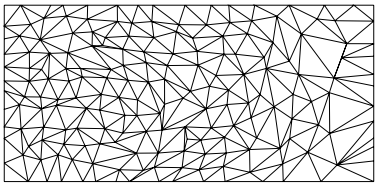

Fig. 3e. Shape-preserving parameterization.

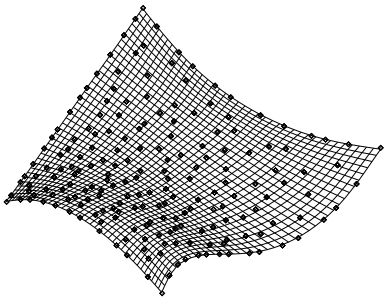

Fig. 3f. Spline surface.

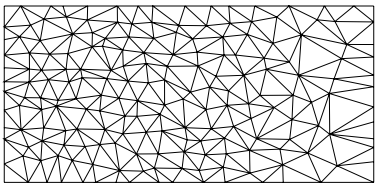

Fig. 3g. Delaunay retriangulation.

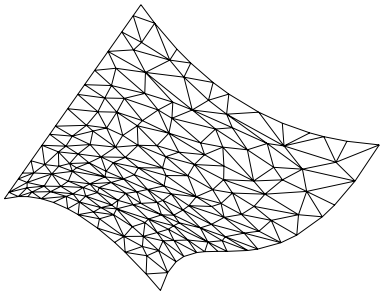

Fig. 3h. Surface retriangulation.

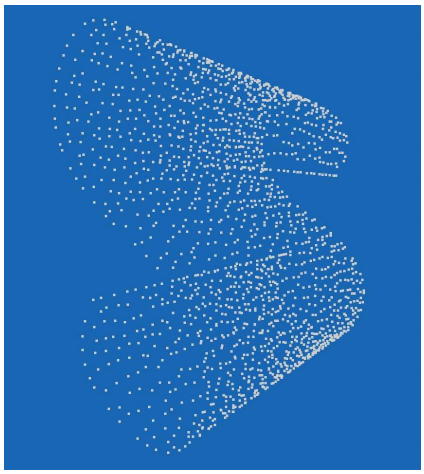

Fig. 4a. Point set.

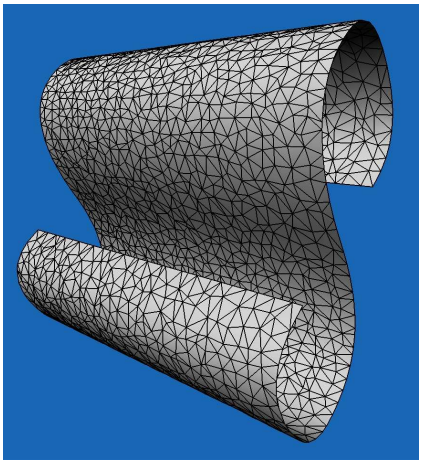

Fig. 4b. Triangulation.

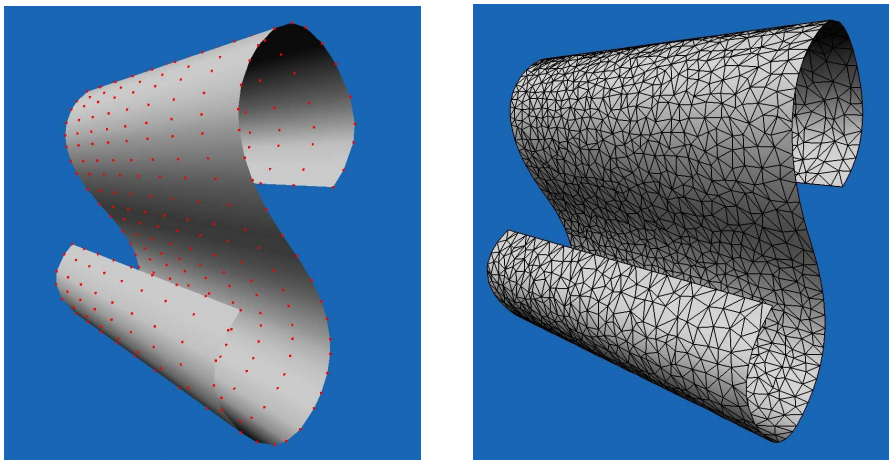

Fig. 4c. Spline surface with patch corners.

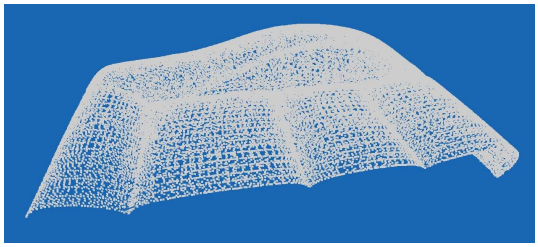

Fig. 5a. Point set.

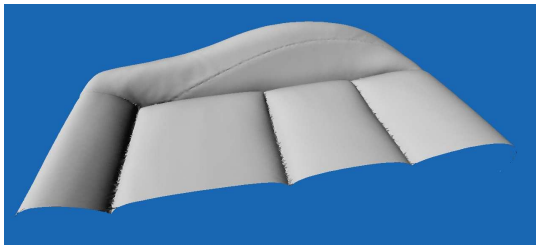

Fig. 5b. Triangulation.

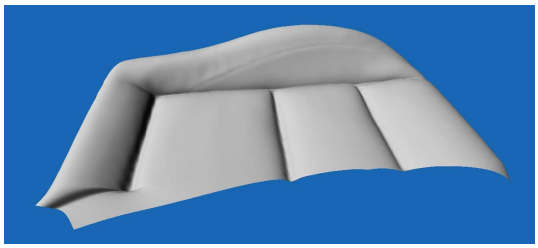

Fig. 5c. Spline surface.

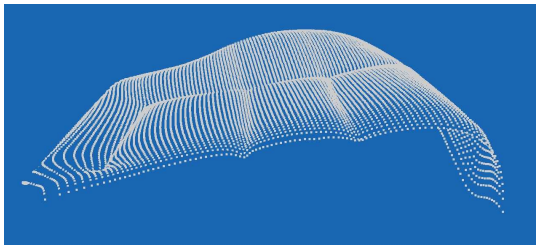

Fig. 5c. Patch corners.

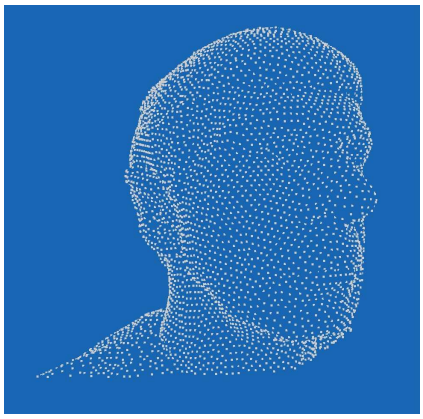

Fig. 6a. Point set.

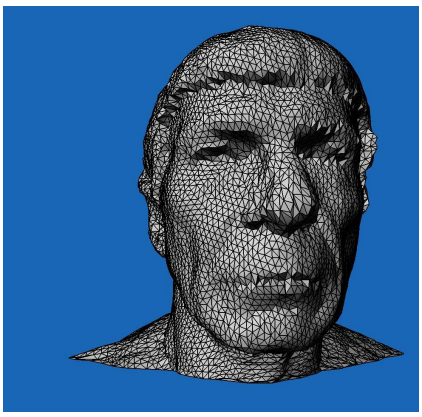

Fig. 6b. Triangulation.

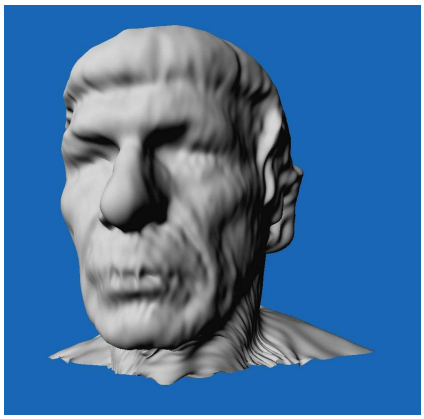

Fig. 6c. Spline surface.

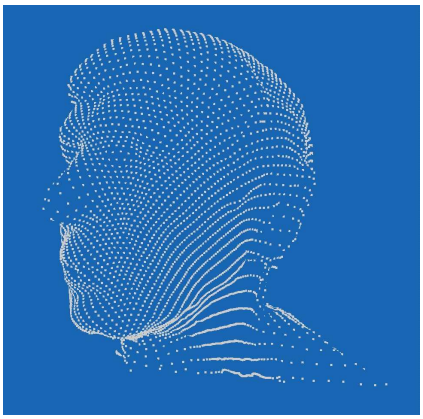

Fig. 6d. Patch corners.

Acknowledgement. I wish to thank Martin Reimers for help with some of the data sets, Ewald Quak for helpful discussions, and Steve Hull of Metrocad GmbH for use of the sofa data set.

References

1. N. Amenta, M. Bern, and M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, Computer Graphics, Proceedings ACM Siggraph 98 (1998), 415–421.
2. C. L. Bajaj, F. Bernardini, and G. Xu, Automatic reconstruction of surfaces and scalar fields from 3d scans, Computer Graphics Proceedings, SIGGRAPH '95, Annual Conference Series (1995), 109–118.
3. J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, ACM Transactions on Graphics **3(4)** (1984), 266–286.
4. C. de Boor, T. Lyche, and L. Schumaker, On calculating with B-splines, II. Integration, in Proc. Oberwolfach, 1975.
5. U. Dietz, B-spline approximation with energy constraints, in *Advanced Course on Fairshape*, J. Hoschek and P. Kaklis (eds.), Teubner, Stuttgart, (1996), 229–240.
6. U. Dietz, Fair surface reconstruction from point clouds, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche & L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, (1998), 79–86.
7. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, SIGGRAPH Comp. Graph. (SIGGRAPH '95 Proceedings) **26**(2) (1995), 173–182.
8. M. S. Floater, Parametrization and smooth approximation of surface triangulations, Comp. Aided Geom. Design **14** (1997), 231–250.
9. M. S. Floater, Parametric tilings and scattered data approximation, International Journal of Shape Modeling **4** (1998), 165–182.
10. M. S. Floater, How to Approximate Scattered Data by Least Squares, SINTEF Report No. STF42 A98013, Oslo, (1998).

11. M. S. Floater and M. Reimers, Meshless parameterization and surface reconstruction, preprint.
12. G. Greiner and K. Hormann, Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville 1997, 163–172.
13. M. von Golitschek and L. L. Schumaker, Data fitting by penalized least squares, in *Algorithms for Approximation II*, J. C. Mason and M. G. Cox (eds.), Chapman & Hall, 1990, 210-227.
14. H. Hoppe, T. DeRose, T. DuChamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Computer Graphics, Vol. 26, No. 2 (1992), 71–78.
15. K. Hormann and G. Greiner, MIPS: An efficient global parametrization method, preprint.
16. J. Hoschek and U. Dietz, Smooth B-spline surface approximation to scattered data, in *Reverse Engineering*, J. Hoschek and W. dankwort (eds.), B.G. Teubner, 1996, 143–152.
17. J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, AKPeters, Wellesley, 1994.
18. F. Isselhard, G. Brunnett, and T. Schreiber, Polyhedral reconstruction of 3d objects by tetrahedral removal, Technical Report No. 288/97, Fachbereich Informatik, University of Kaiserslautern, Germany, 1997.
19. C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, SIAM, Philadelphia, 1995.
20. B. Lévy and J. L. Mallet, Non-distorted texture mapping for sheared triangulated meshes, Proceedings of SIGGRAPH 98 (1998), 343–352.
21. T. Schreiber and G. Brunnett, Approximating 3d objects from measured points, in *Proceedings of 30th ISATA*, Florence, Italy, 1997.
22. T. Varady, R. R. Martin, and J. Cox, Reverse engineering of geometric models — an introduction, CAD **29** (1997), 255–268.
23. W. Welch and A. Witkin, Free-form shape design using triangulated surfaces, Computer Graphics, SIGGRAPH 1994, 247–256.