Least–squares fitting of algebraic spline surfaces *

Bert Jüttler [a] Alf Felis [b]

[a] *Johannes Kepler University, Altenberger Str. 69, 4040 Linz, Austria*
E-mail: Bert.Juettler@jk.uni-linz.ac.at
[b] *ProSTEP GmbH, Dolivostr. 11, 64293 Darmstadt, Germany*

We present an algorithm for fitting implicitly defined algebraic spline surfaces to given scattered data. By simultaneously approximating points and associated normal vectors, we obtain a method which is computationally simple, as the result is obtained by solving a system of linear equations. In addition, the result is geometrically invariant, as no artificial normalization is introduced. The potential applications of the algorithm include the reconstruction of free form surfaces in reverse engineering. The paper also addresses the generation of exact error bounds, directly from the coefficients of the implicit representation.

Keywords: Algebraic surface, spline surface, surface fitting, least squares fitting, reverse engineering.

AMS Subject classification: 65D17, 68U07, 53A05

1. Introduction

We describe computational techniques for reconstructing surfaces from scattered data in 3D. This surface fitting problem arises in the context of the process of reverse engineering. Due to the recent progress in the technology of 3D scanners, even complex objects can now be digitized with impressive accuracy. Consequently, reverse engineering of free–form shapes becomes an increasingly valuable alternative to the standard top–down engineering process in Computer Aided Design.

In the sequel we use implicitly defined algebraic spline surfaces for generating free–form surfaces. Traditionally, most CAD systems rely on parametric representations, such as NURBS surfaces, see [7,12]. When compared to these representations, the use of implicitly defined surfaces offers a number of advantages.

* The research described in this paper was partly done within a project leading to the Master thesis of the second author [8]. The financial support by Holometric Technologies GmbH (Aalen, Germany) is gratefully acknowledged.

First, in order to use the algorithms for surface fitting with parametric representations, one always has to assign auxiliary parameter values to the data. This is often done by projecting the data into some auxiliary surface [18]. A more sophisticated method has recently been described in [9]. The estimated parameters have a strong influence to the resulting shape, and the generation of suitable values can sometimes be difficult, in particular for more complex shapes. Clearly, the parameterization will have problems if the shape of the data is not compatible with that of a segment of the plane. As a major advantage of algebraic (spline) surfaces, these parameters are not needed in the implicit case.

Second, the problem of implicit surface fitting can be solved without generating a triangulation of the 3D data. Although algorithms for generating triangular meshes from scattered data have made some progress (in particular due to recent advances in Computer Graphics), 'meshless' techniques may help to circumvent the various computational issues associated with triangular meshes. (It should be noted, that the parameterization can be generated even without an auxiliary triangulation, using a meshless technique [9]. However, many techniques for parameterization assume the existence of a triangulation.)

Third, shape constraints (e.g., convexity conditions) for implicitly defined spline surfaces are much easier to deal with than in the parametric case. For instance, the convexity of an algebraic spline surface can be guaranteed by the convexity of the underlying spline function, see [13] for suitable linear criteria. Convexity conditions for truly parametric representations, by contrast, are highly non–linear, see [16]. Consequently, it is relatively easy to introduce shape constraints to the framework of implicit spline surface fitting. Typically, this produces optimization problems with linear constraints, which are relatively easy to deal with.

Other advantages of implicitly defined shapes (not directly related to surface fitting) include the possibility to define solids, simply be evaluating the sign of the generating real function, and availability of simple algorithms for computing the intersection(s), e.g., with straight lines.

On the other hand, the use of algebraic spline surfaces leads to some additional issues which have to be addressed. For instance, special algorithms are needed for visualizing and evaluating these surfaces, such as marching cubes, cf. [6,12]. Also, in order to exchange data with industrial CAD systems, the implicitly defined surfaces have to be converted into parametric (NURBS–) format.

Various methods for implicit surface fitting have been described in the vast literature on the subject, see [2,19,22] and others. Most of these methods use the so–called algebraic distance and combine it with a suitable normalization of the coefficients.

For instance, Pratt ('simple fit' method, [19]) introduces a linear normalization, by keeping the value of one of the coefficients. Clearly, this normalization has no geometric meaning, as it depends on the choice of the coordinates.

In order to obtain a geometrically invariant normalization, Taubin [22] constrains the sum of the squared gradients at the data sites. Minimizing the algebraic distance then produces a constrained quadratic programming problem. The solutions are found by (numerically) solving a generalized eigenvalue problem.

Recently, the methods of Pratt and Taubin have been compared by Umasuthan and Wallace [23].

Bajaj and various co–authors [1,4] have developed implicit algebraic surfaces into a powerful tool for surface design. Their approach is mainly based on low–degree patches, where the coefficients are made to satisfy certain sign conditions, in order to guarantee the desired topological structure.

Werghi et al. [25] have developed an incremental framework, incorporating geometric constraints (such as orthogonality), for fitting implicitly defined geometric primitives, such as planes and quadrics.

In this article we describe a new technique for surface fitting with algebraic spline surfaces. We consider a set of points

$$\mathbf{p}_i = (p_{i,1}, p_{i,2}, p_{i,3}) \in \mathbb{R}^3; \quad i = 1, \ldots, N. \tag{1}$$

A typical data set, consisting of 16,500 points, is shown in Figure 1.

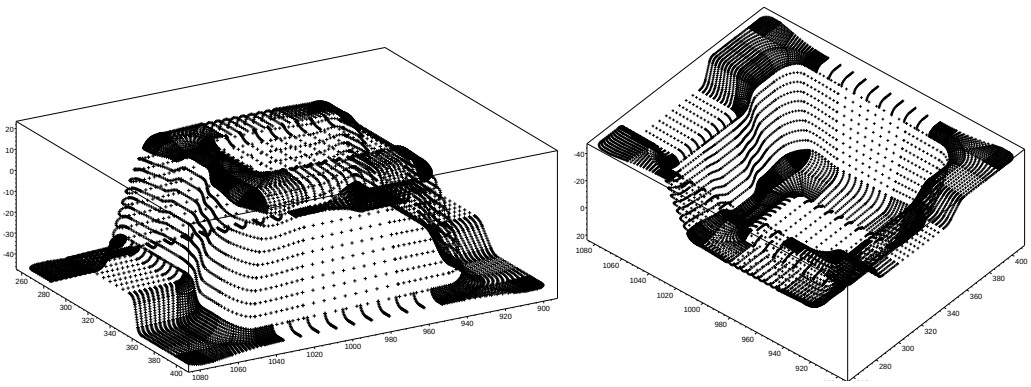


Figure 1. The data set, consisting of 16,500 points (Courtesy of Holometric Technologies GmbH).

As a preprocessing step we estimate normal vectors $\vec{\mathbf{n}}_i$ which can be associated with the given data. Then, an algebraic spline surface

$$\{\, (x_1, x_2, x_3) \mid f(x_1, x_2, x_3) = 0, \quad (x_1, x_2, x_3) \in \Omega \,\} \tag{2}$$

(where $f(.)$ is a trivariate real spline function with domain $\Omega \subset \mathbb{R}^3$) is generated by simultaneously approximating both the data and the associated normals. This approach avoids the use of an artificial normalization. Consequently, the method is both computationally simple (as the result is obtained by solving a system of linear equations) and geometrically invariant.

A similar algorithm for scattered data fitting with planar algebraic spline curves has been described in the conference article [15]. In the present paper we generalize it to the surface case.

In the final section of the paper, we address the generation of exact error bounds, directly from the coefficients of the implicit representation.

2. Preprocessing: Estimation of associated normal vectors

The estimation of normal vectors $\vec{\mathbf{n}}_i = (n_{i,1}, n_{i,2}, n_{i,3}) \in \mathbb{S}^2$, $i = 1, \ldots, N$, where $\mathbb{S}^2$ is the 3D unit sphere, from the given scattered data (1) is a well–known problem. In many cases, the normal vectors can be obtained from additional information. For instance, if the points $\mathbf{p}_i$ are generated from a certain 3D scalar field, such as in Computer Tomography, then the normal vectors could be chosen as the (normalized) gradients of the scalar function.

Alternatively, the normal vectors can be estimated from the data. As a simple approach, one may use – for each point $\mathbf{p}_i$ – the normal vector $\vec{\mathbf{n}}_i$ of a local plane of regression P_i. Let $(\mathbf{p}_j)_{j \in \mathcal{N}_i}$. be the set of the neighbouring points, possibly with associated non–negative weights. One may choose a fixed number of closest neighbours to $\mathbf{p}_i$, or all points within a certain neighbourhood of $\mathbf{p}_i$. Clearly, one has to use efficient techniques for structuring the data, in order to find the neighbouring points in reasonable computing time, cf. [20]. The normal vector of the plane of regression is the eigenvector associated with the smallest eigenvalue of the real 3×3 matrix

$$\sum_{j \in \mathcal{N}_i} (\mathbf{p}_j - \mathbf{p}_i^*)(\mathbf{p}_j - \mathbf{p}_i^*)^\top, \tag{3}$$

where $\mathbf{p}_i^*$ is the center of gravity of the (possibly weighted) point set $(\mathbf{p}_j)_{j \in \mathcal{N}_i}$, and coordinate vectors are chosen to be column vectors. Its computation involves the (numerical) solution of a cubic equation.

This simple estimate can be improved by using a more sophisticated local surface of regression, providing more degrees of freedom. For instance, one may use the graph of a quadratic polynomial, defined over the local plane of regression (that is, a paraboloid with axis $\vec{\mathbf{n}}_i$). See [15] for the analogous technique in the planar situation.

In order to get useful results, we need to make sure that neighbouring normal vectors point approximately into the same direction. More precisely, the inner products $\vec{\mathbf{n}}_i \cdot \vec{\mathbf{n}}_j$ of normal vectors associated with neighbouring points should be non–negative. As the estimated normals obtained from the plane of regression will be randomly oriented, some of them have to be swapped. In our current implementation of the method, we use a simple region–growing technique in order to guarantee the desired orientation.

With the help of the local planes of regression, we have estimated normal vectors for the 16,500 data from Figure 1. The result is shown in Figure 2. The local planes of regression have been computed using the 25 closest neighbours of each point.

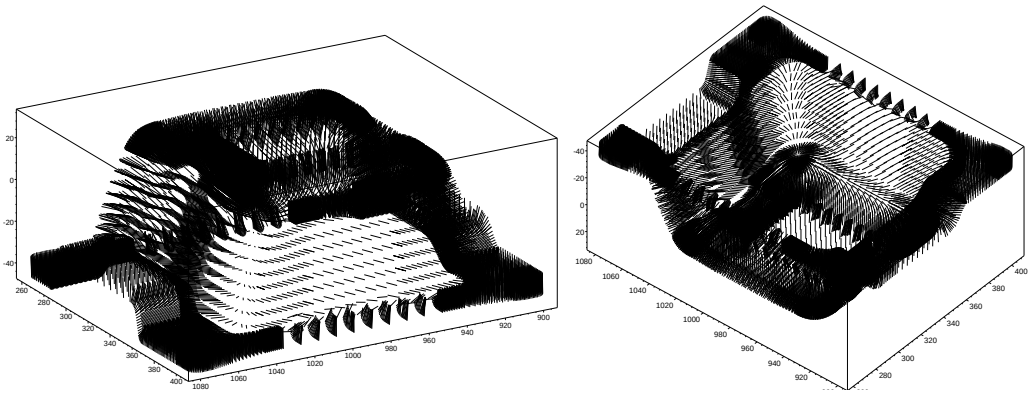


Figure 2. The associated normal vectors.

The number of neighbours which should be used for estimating the normal depends on the local distribution of the data. In our example, most data are organized in scanlines. Hence, in order to get a stable estimate, the number of neighbours should be large enough such that data from more than one scanline is used. The process of choosing a suitable number of neighbours can partly be automatized by taking the ratio of the first two smallest eigenvalues of the matrix (a ratio which is close to 1 corresponds to an instable estimate, hence the number should be increased) and the distances of the points from the local plane of regression (if the distance is too large, then the number should be decreased) into account.

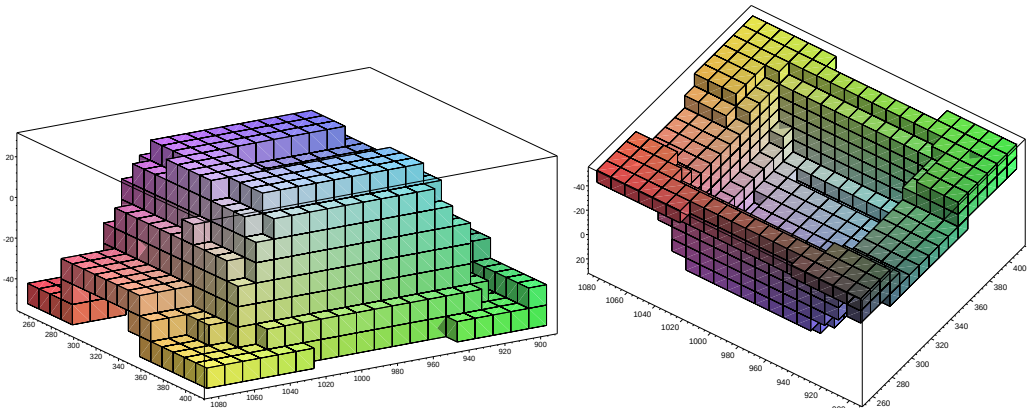


Figure 3. The figures show all boxes which contain data. The domain Ω of the tensor–product spline function (4) consists of all theses boxes, and all the neighbouring ones.

3. Algebraic tensor–product spline surfaces

The algebraic spline surface is defined as the zero level set (2) of a tensor–product spline function of (tri–) degree d ($d \geq 2$),

$$f(x,y,z) = \sum_{(i,j,k)\in\mathcal{I}} M_i(x)\; N_j(y)\; O_k(z)\; c_{i,j,k} \tag{4}$$

with the real coefficients (control points) $c_{i,j,k}$, where $\mathcal{I} \subset \mathbb{Z}^3$ is a certain index set, see below. The basis functions $(M_i(x))_{i=1..m}$, $(N_j(y)_{j=1..n})$, and $(O_k(z)_{k=1..o})$ are B-splines of degree d with respect to the knot sequences $\mathcal{X} = (\xi_i)_{i=1..m+1}$, $\mathcal{Y} = (\eta_j)_{j=1..n+1}$, and $\mathcal{Z} = (\zeta_k)_{k=1..o+1}$, see [5,12]. We choose the first and last $d+1$ knots of the knot sequences $\mathcal{X}$, $\mathcal{Y}$ resp. $\mathcal{Z}$ as the vertex coordinates of the slightly enlarged (blown up) bounding box of the data $(\mathbf{p}_l)_{l=1,..,N}$. The remaining inner knots are equidistant.

The knot sequences define a partition of the (slightly enlarged) bounding box $[\xi_1, \xi_{m+d+1}] \times [\eta_1, \eta_{n+d+1}] \times [\zeta_1, \zeta_{o+d+1}]$ into rectangular boxes. Generally, most of these boxes do not contain any data. The domain Ω of the tensor–product spline function (4) is the union of all cells that contain data, plus all neighbouring boxes, cf. Figure 3. If the data have been sampled from an existing surface, and if their distribution is not too irregular, then this choice should be sufficient to construct a reasonable approximating spline surface. Otherwise, further boxes have to be added. The summation in (4) includes all products $M_i(x)\; N_j(y)\; O_k(z)$ that do not vanish on the domain Ω, i.e.,

$$\begin{aligned}\mathcal{I} = \{\; (i,j,k) \mid \exists\, l \in \{1,..,N\}, r \in \{1,..,m\}, s \in \{1,..,n\}, t \in \{1,..,o\} :\\ M_r(p_{l,1})\; N_s(p_{l,2})\; O_t(p_{l,3}) \neq 0 \;\wedge\; \max\{|i-r|, |j-s|, |k-t|\} \leq 1 \;\}.\end{aligned} \tag{5}$$

The number of degrees of freedom (and hence the numbers m, n, o of B-splines) depends on the knot spacing. Clearly, there is a tradeoff between efficiency and accuracy, as smaller boxes may lead both to more accurate results and to increasing computing times. Although an initial estimate can be generated automatically, the optimal choice still requires some user interaction (similar to the situation for parametric surface fitting).

Clearly, the partition of the space into rectangular boxes, and hence the domain Ω of the tensor–product spline function (4), depend on the choice of coordinates. If a geometrically invariant choice is needed, it can easily be obtained with the help of the eigenvectors of the matrix of inertia of the data.

The use of tensor–product spline offers several advantages, including simple implementation, simple conditions for global smoothness and differentiability, simple evaluation, sufficient flexibility and refinability (e.g. using hierarchical B-spline representations, see [10]). However, as a certain disadvantage, the segments of the resulting spline (2) are algebraic surfaces of the relatively high order $3d$.

In order to keep the algebraic order as low as possible (if needed), it would be more appropriate to choose a trivariate spline function of total degree d, defined with respect to a partition of space into tetrahedra. For instance one may choose 3D Powell–Sabin elements or simplex splines, see e.g. [11,12]. We preferred to use the tensor–product representations, as the resulting data structures are far simpler than in the tetrahedral case.

Within the tensor–product setting, a lower algebraic degree can be guaranteed by introducing additional side–conditions. For instance, the zero level set (2) of a triquadratic tensor–product spline which satisfies the linear constraints

$$f_{xxy} = f_{xxz} = f_{xyy} = f_{yyz} = f_{yzz} = f_{yzz} = f_{xyz} = 0 \tag{6}$$

is C^1 spline surfaces of algebraic order 2, i.e., C^1 quadric spline surface. These constraints, however, lead to a highly redundant representation.

Alternatively, one may easily convert tensor–product spline surfaces approximately into lower order algebraic spline surfaces, with any desired accuracy. For instance, this can be achieved with the help of Powell–Sabin–type or Clough–Tocher–type 3D Hermite interpolants which are defined with respect to a suitable tetrahedralization of the 3D domain.

The approximation method which is described in the next section can be applied to any implicitly defined algebraic spline surface, not only to surfaces in tensor–product representation. Due to the use of tensor–products, the implementation of the method is relatively simple. The results, however, depend on the choice of the system of coordinates (unless we choose the coordinates with the help of the matrix of inertia, as outlined before). Note that this dependency

is caused only by the choice of the space of functions, not by the approximation method. For instance, applying the approximation method to trivariate polynomials of total degree d (instead of spline surfaces) leads to geometrically invariant results.

4. Surface fitting

We compute the coefficients $(c_{i,j,k})_{(i,j,k)\in\mathcal{I}}$ of the tensor–product spline function f by minimizing a quadratic objective function. In order to simplify notations, the spline coefficients (in a suitable ordering) are gathered in a (column) vector $\mathbf{c}$.

4.1. Approximating the data and the normals

First we consider the data $(\mathbf{p}_i)_{i=1,\dots,N}$ and the associated normal vectors $(\vec{\mathbf{n}}_i)_{i=1,\dots,N}$. In order to approximate the data we will minimize the sum of the squared 'algebraic distances' (see [19,22]),

$$L(\mathbf{c}) = \sum_{i=1}^{N} [\ f(p_{i,1}, p_{i,2}, p_{i,3})\ ]^2. \tag{7}$$

This gives a quadratic form, $L(\mathbf{c}) = \mathbf{c}^\top Q\mathbf{c}$. Its minimizer is the null vector $(c_{i,j,k} = 0)_{i\in\mathcal{I}}$, corresponding to the trivial spline function $f(x, y, z) \equiv 0$.

In order to get useful results, one has to introduce a suitable normalization. Several techniques have been described in the literature [2,19,22], most of them based on a suitable norm in the coefficient space. We will use a different approach. In addition to the quadratic form L, we minimize the sum

$$\begin{aligned} M(\mathbf{c}) &= \sum_{i=1}^{N} \parallel \nabla f(p_{i,1}, p_{i,2}, p_{i,3}) - \vec{\mathbf{n}}_i \parallel^2 \\ &= \sum_{i=1}^{N} (f_x(p_{i,1}, p_{i,2}, p_{i,3}) - n_{i,1})^2 + (f_y(p_{i,1}, p_{i,2}, p_{i,3}) - n_{i,2})^2 \\ &\qquad + (f_z(p_{i,1}, p_{i,2}, p_{i,3}) - n_{i,3})^2. \end{aligned} \tag{8}$$

Consequently, the gradients $\nabla f = (f_x,\ f_y,\ f_z)$ of the tensor–product spline function $f(x, y, z)$ at the given data $\mathbf{p}_i$ will match the estimated normal vectors $\vec{\mathbf{n}}_i$. Note that the given (estimated) normals $\vec{\mathbf{n}}_i$ are unit vectors. Consequently, the algebraic distances in (7) can be expected to approximate the real distances. See Section 5 for more information on distance bounds for implicit representations.

4.2. Adding Tension

Minimizing a weighted linear combination of L and M may lead to a singular system of equations, as it may happen that some of the products $M_i(x)\ N_j(y)\ O_k(z)$, $(i,j,k) \in \mathcal{I}$, of basis functions vanish on all data, cf. (5). Even if the system has full rank, the approximating spline surfaces may split into several disconnected components. In order to resolve these problems, we add a simple 'tension term' that pulls the approximating surface towards a simpler shape. If the influence of this tension term (governed by a non–negative weight) is strong enough, then the approximating surface has the desired topology and the linear system is non–singular.

A possible quadratic tension term is

$$G(\mathbf{c}) = \iiint_{\Omega} f_{xx}^2 + 2\, f_{xy}^2 + f_{yy}^2 + 2\, f_{xz}^2 + 2\, f_{yz}^2 + f_{zz}^2 \, dx\, dy\, dz, \tag{9}$$

measuring the deviation of $f(x,y,z)$ from a linear function. Hence, by increasing the influence of this tension term, the resulting spline becomes more similar to a plane.

In addition to the global tension term one may use data–dependent tension terms, similar to the ones used in [15], or higher–order terms, involving derivatives of higher than second order. As another straightforward generalization, the tension can be localized by inserting a non–negative weight function $w(x,y,z)$ in the integral (9).

4.3. Solving the surface fitting problem

The coefficients $(c_{i,j,k})_{(i,j,k)\in\mathcal{I}}$ are computed by minimizing the weighted linear combination

$$F(\mathbf{c}) = L(\mathbf{c}) + w_1\ M(\mathbf{c}) + w_2\ G(\mathbf{c}) \quad \rightarrow \quad \text{Min}, \tag{10}$$

see (7), (8) and (9), with certain non–negative weights w_1 and w_2. This leads to a quadratic objective function of the unknown control points $\mathbf{c} = (c_{i,j,k})_{(i,j,k)\in\mathcal{I}}$. Consequently, the solution can be found by solving the sparse linear system of equations

$$\frac{\partial}{\partial c_{i,j,k}} F(\mathbf{c}) = 0, \quad (i,j,k) \in \mathcal{I}, \tag{11}$$

with the help of appropriate methods from numerical linear algebra. Alternatively, one may also compute the solution of (10) by generating an overconstrained system of linear equations, and computing a least–squares solution to it using QU

factorization, see e.g. [5]. As another possibility, an approximate solution can be found with the help of quadratic programming tools such as Vanderbei's LOQO package [24].

As a first advantage of our approach, the simultaneous approximation of points and normal vectors helps to keep unwanted branches away from the approximating surface. This is demonstrated by the curve example shown in Fig. 4, where we computed various approximations of a planar data set (points with associated normals, shown in grey) by the zero contour of a bicubic polynomial. The curves in the first three figures have been generated by minimizing the objective function (10) for three values of w_1, leading to systems of linear equations. The weight w_2 of the tension is either 0 (low tension, marked by 'l'), 0.01 ('m'edium tension), or 0.1 ('h'igh tension). The curves in the fourth figure are obtained by minimizing the objective function (10) with $w_1 = 0$, but subject to the quadratic normalization $\sum_{i=1}^{N} ||\nabla f(p_{i,1}, p_{i,2}, p_{i,3})||^2 = 1$, which leads to a generalized eigenvalue problem (Taubin's method [22]). Without tension ('l'), the result of Taubin's scheme splits into a curve and an additional closed loop.

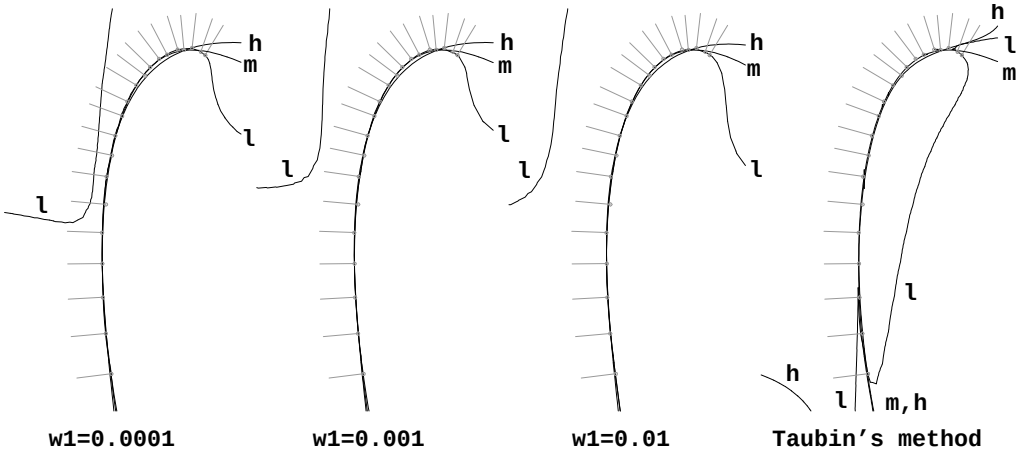


Figure 4. Influence of the weights w_1, w_2 and comparison with Taubin's method. The weight w_2 of the tension term takes the values 0 ('l'ow), 0.01 ('m'edium), and 0.1 ('h'igh).

The figures illustrate the effects which can be achieved by various weights w_1 and w_2. Increasing either weight pushes unwanted the branches of the curve away. The weight w_1 controls the influence of the estimated normals. The tension term G (associated with w_2) tends to flatten the curve. With the help of the first weight, we may obtain useful results without having to resort to flattened shapes.

Clearly, the automatic choice of the weights is a challenging problem, as the optimal values depend on the shape and the level of noise present in the

data. This is similar to the choice of tension parameters for parametric surface fitting with "fainess functionals". Currently, the optimal choice is determined by numerical experiments involving user interaction. A semi–automatic heuristical choice should take the variation of the normal vectors and the presence of unwanted branches (which can be checked using positivity criteria for the directional derivatives) into account. A high variation should correspond to small values of w_1, and unwanted branches should lead to an increase of both weights.

As a second advantage of our approach, the approximating spline surface can be found by solving a system of linear equations. This is to be compared with the normalization–based methods [19,22], where the solution is computed by solving a generalized eigenvalue problem. Clearly, solving a linear system seems to be the easier task. By taking the sparsity into account, and using suitable tools from numerical linear algebra, such as a PCG method (see Chapter 6 of [17]), it should be possible to achieve a computational complexity of $\mathcal{O}(h^2)$ for solving the sparse linear system, where $h = |\mathcal{I}|$ is the number of coefficients. According to Taubin [22], solving the generalized eigenvalue problem needs algorithms with complexity $\mathcal{O}(h^3)$. However, taking again the sparsity into account, and using more sophisticated tools from numerical analysis, the computational complexity should be similar to the case of linear systems.

4.4. Existence and uniqueness

Under certain mild assumptions, the surface fitting problem (10) can be shown to lead to a unique solution.

Proposition 1. If the weights w_1 and w_2 are positive, then the quadratic optimization problem (10) has a unique solution.

Proof. First we observe that the three quadratic functionals (7), (8) and (9) are convex functions $\mathbb{R}^{|\mathcal{I}|} \to \mathbb{R}$ on the coefficient vectors $\mathbf{c}$. The intersection of the three null spaces of the three quadratic functionals is either empty, or it consists of a single linear function. (E.g., for $N = 1$, the intersection of the three null spaces is the unique linear function f with $f(p_{1,1}, p_{1,2}, p_{1,3}) = 0$ and $\nabla f(p_{1,1}, p_{1,2}, p_{1,3}) = \vec{\mathbf{n}}_1$.) Consequently, the weighted linear combination with positive coefficients w_1, w_2 is a strictly convex function. □

Consequently, the rank of the square coefficient matrix of the linear system (11) equals the number of coefficients $h = |\mathcal{I}|$.

The tension term G is the only part of the objective function (10) that acts on *all* cells of the domain Ω. The other parts act only on cells which contain data.

Thus, choosing $w_2 = 0$ will produce a singular system (11), if at least one of the products $M_i(x)\, N_j(y)\, O_k(z)$ vanishes at all data $(x, y, z) = (p_{l,1}, p_{l,2}, p_{l,3})$. This may happen quite easily, as the domain of the spline function $f(x, y)$ consists of all cells containing data, and the neigbouring cells.

4.5. Examples

We have applied the surface fitting procedure to the 16,500 data shown in Figure 1 with the associated normal vectors shown in Figure 2. The function f is a tri–quadratic tensor–product spline function; its domain consist of all cells shown in Figure 3 and all neigbouring cells (the gridsize is 10). The function f is described by $|\mathcal{I}| = 3015$ coefficients. The resulting data volume corresponds to a parametric tensor product surface patch with 32×32 control points.

The implicitly defined surface (2) is shown in Figure 5. Its coefficients have

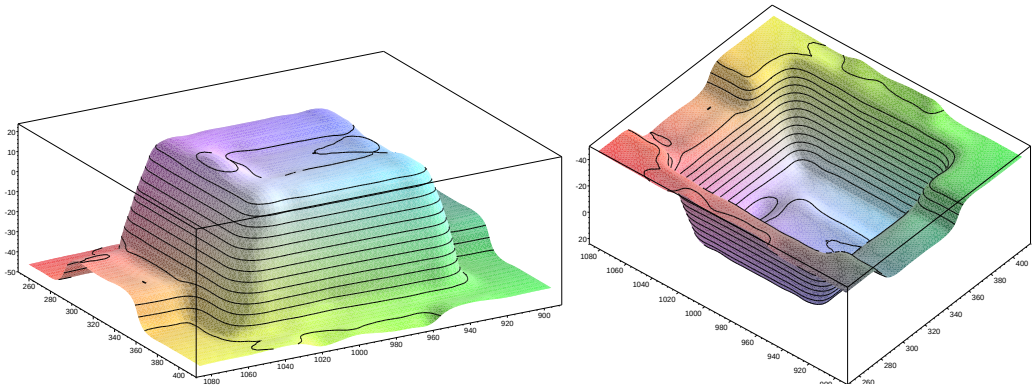


Figure 5. Approximation of the data from Figure 1 with the estimated normals (see Figure 2) by an algebraic tensor–product spline surfaces of degree 2.

been found by minimizing the objective function (10) with the weights $w_1 = 1$ and $w_2 = 0.0001$.

The maximum distance error is equal to 2.3 (dimensions of the bounding box are approx. $200 \times 150 \times 70$). For 80% of the data, the error does not exceed 0.36.

Figure 6 visualizes the distribution of the error. Although the error distribution is fairly uniform, some regions larger error exist, e.g., along the boundaries of the trough on top of the object. Adding degrees of freedom locally (e.g., using a hierarchical B-spline representation, such as spline wavelets) would improve the result. This will be a possible subject of future research.

In order to illustrate the influence of the tension term (9), we have applied the method to a set of 2,645 points sampled from the surface of a sphere. The

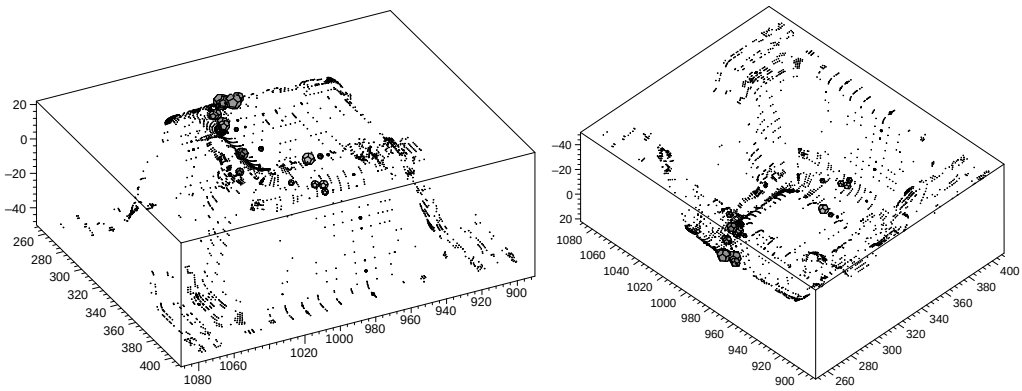


Figure 6. Error distribution. The 20 % of the data with an error exceeding 0.36 are shown. For 105 of them, the error is even greater than 1.00; these points are marked by dodecahedrons, with the radius equal to $3 \times (\text{error} - 1)$.

estimated normal vectors are almost identical to the normals of the sphere. We computed two different solutions, choosing $w_2 = 0.0001$ and $w_2 = 0.25$, see Figure 7.

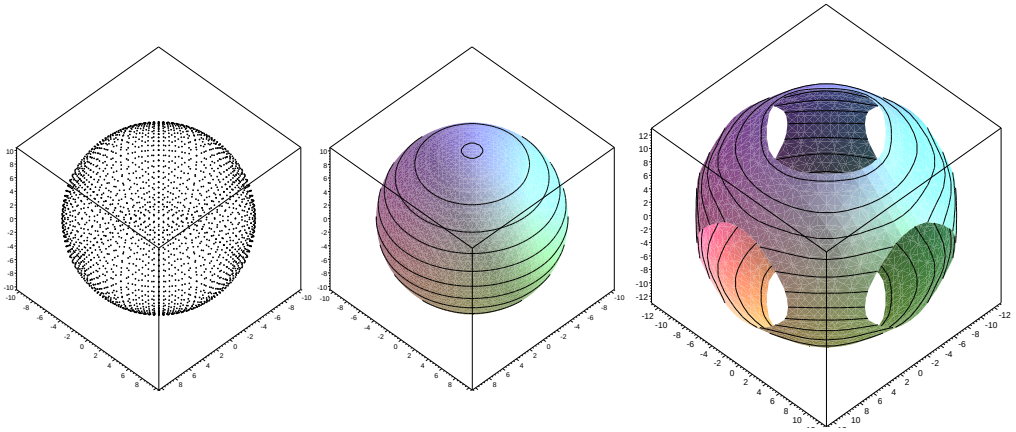


Figure 7. Sphere example: influence of the tension term. The data (left), the resulting surface for $w_2 = 0.0001$ (center) and $w_2 = 0.25$ (right).

Here, increasing the weight w_2 tension leads to a "blowing up" effect of the resulting surface. After increasing the weight w_2, the algorithm produces a surface which is flatter than the previous result. This is due to the fact that the level surfaces of functions with $G(\mathbf{c}) = 0$ are planes.

Finally, we have applied the method to a set of 90,695 points; 10 % of these data are shown in Figure 8. Due to the curved shape and the noise which

is present in the data, the estimation of normals is more difficult than in the previous examples. The algebraic spline surface is the zero contour of a spline function with 2,184 coefficients; the domain consists of 1,111 boxes. The data volume is that of a tensor–product spline surface with 26×27 control points. The maximum distance error equals 6.7, where the bounding box has the dimensions $250 \times 40 \times 120$. For 80% of the data, the error is smaller than 1.0.

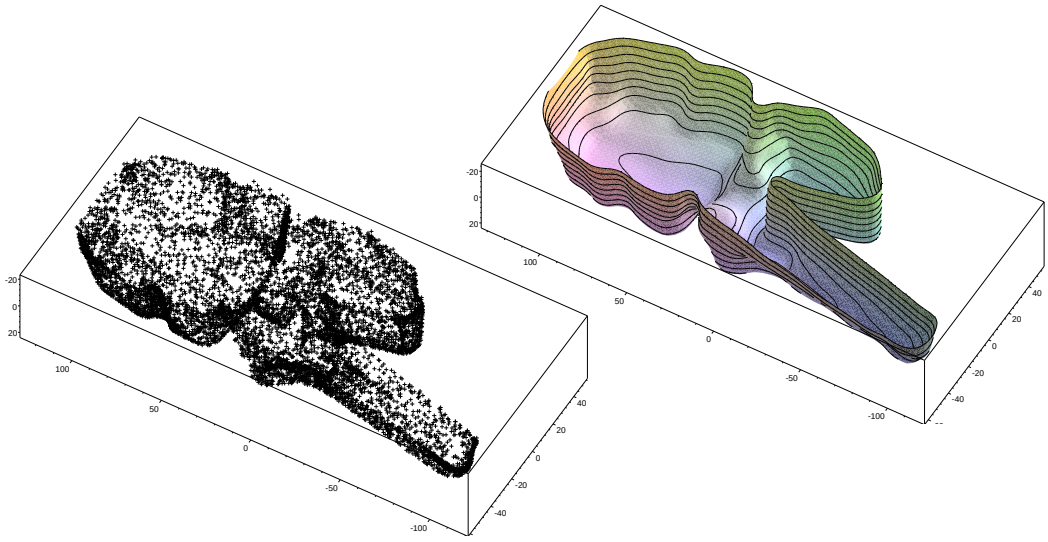


Figure 8. Algebraic spline surface approximating another data set (Data courtesy of Holometric Technologies GmbH)

4.6. Adapting the objective function

After an initial solution has been found, the objective function can be subject to an iterative adaptation, in order to obtain a better result. Then, a new solution is found by minimizing the new objective function. This approach is similar in spirit to the well-known method of parameter correction, see [12]. We give an outline of two possible adaptations. Both ideas have been implemented and tested in the planar situation, see [15] for further details.

Generally, the algebraic distances (7) are not the true distances between the data and the approximating spline surface. One may use a weighted least–squares sum instead, where each term has a positive weight ω_i. Ideally, the weights would be chosen such that weighted least–squares sum approximates the sum of the squared Euclidean distances. This can be achieved with the help of a suitable 'reweight procedure', see [22].

A similar procedure can be applied to the estimated normal vectors $(\vec{\mathbf{n}}_i)_{i=1,\ldots,N}$. In order to adjust these vectors $\vec{\mathbf{n}}_i$, one may use the normalized gradients of the first approximation, evaluated at the data $\mathbf{p}_i$.

5. Bounding the error

In order to check the quality of the surface fit, one may generate the footpoints to all data $\mathbf{p}_i$, using Newton–Raphson iterations. Alternatively, with the help of the implicit representation of the surface, an exact upper bound can be generated. This bound is of the form

$$\inf_{f(\mathbf{x})=0,\, \mathbf{x}\in\Omega} \|\mathbf{p}_i - \mathbf{x}\| \le K\, |f(\mathbf{p}_i)|, \tag{12}$$

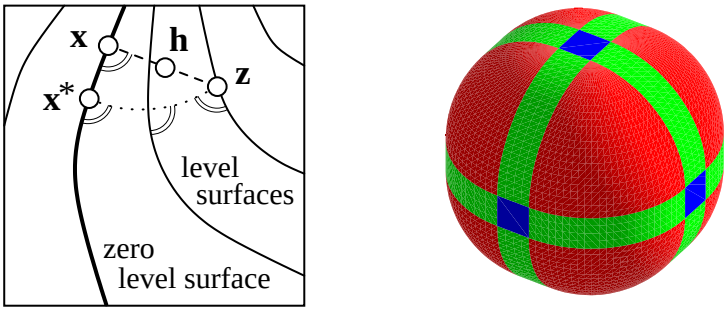
where K is a certain constant.

5.1. Footpoint distance

Consider a point $\mathbf{z} \in \mathbb{R}^3$. Any point $\mathbf{x} = (x_1, x_2, x_3) \in \Omega$ which satisfies both

$$f(x_1, x_2, x_3) = 0 \quad \text{and} \quad (\mathbf{z} - \mathbf{x}) \times \nabla f(x_1, x_2, x_3) = \vec{\mathbf{0}} \tag{13}$$

is called a *footpoint* of $\mathbf{z}$, see Figure 9a for a schematic illustration.



(a) (b)

Figure 9. Footpoint distance (a), offset surface of a cube (b).

Lemma 2. Consider a point $\mathbf{z} \in \Omega$, and let $\mathbf{x}$ be a footpoint of it. If the line segment $\overline{\mathbf{xz}}$ is contained in the domain Ω, then the footpoint distance can be expressed as

$$\|\mathbf{x} - \mathbf{z}\| = \frac{\|\nabla f(\mathbf{x})\|}{|\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{h})|}\, |f(\mathbf{z})|, \tag{14}$$

where $\mathbf{h}$ is a certain point on the line segment connecting the points $\mathbf{x}$ and $\mathbf{z}$.

Proof. We restrict the function f to the normal of the surface (2) at $\mathbf{x}$,

$$g(t) = f(\mathbf{x} + t \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}). \tag{15}$$

Due to the footpoint property, it satisfies one of the equations

$$g(\|\mathbf{x} - \mathbf{z}\|) = f(\mathbf{z}) \quad \text{or} \quad g(-\|\mathbf{x} - \mathbf{z}\|) = f(\mathbf{z}), \tag{16}$$

depending on the orientation of the gradients. In addition, $g(0) = 0$ holds. With the help of the mean value theorem, one gets in the first case

$$\frac{f(\mathbf{z}) - 0}{\|\mathbf{x} - \mathbf{z}\|} = g'(\lambda) = \nabla f(\mathbf{h}) \cdot \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \tag{17}$$

for some $\lambda \in [0, \|\mathbf{x} - \mathbf{z}\|]$, where

$$\mathbf{h} = \mathbf{x} + \lambda \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \tag{18}$$

By solving this equation for the footpoint distance, we obtain the desired result. The second case follows similarly. □

5.2. Two auxiliary constants

We generate two bounds C and D_h from the coefficients $c_{i,j,k}$ of the spline function. C is an upper bound C on the length of the gradients,

$$\|\nabla f(\mathbf{x})\| \leq C \quad \text{for} \quad \mathbf{x} \in \Omega. \tag{19}$$

D_h is a lower bound on the inner product of the gradients of any two neighbouring points whose distance does not exceed a certain constant h,

$$|\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{y})| \geq D_h \text{ holds for all } \mathbf{x}, \mathbf{y} \in \Omega \text{ with } \|\mathbf{x} - \mathbf{y}\| \leq h. \tag{20}$$

In order to generate these constants we subdivide the spline function f into polynomial segments with the subdomains $\Omega^{(l)}$,

$$f(\mathbf{x}) = f^{(l)}(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \Omega^{(l)}, \quad l = 1, \ldots, L. \tag{21}$$

The subdomains $\Omega^{(l)}$ of the polynomial pieces are either the cells of the original spline function, or they are obtained by splitting them further into even smaller boxes. The additional splitting can be used in order to obtain tighter bounds C and D_h.

With the help of knot insertion we generate for each subdomain a tensor–product Bézier representation

$$f^{(l)}(x,y,z) = \sum_{i,j,k=0}^{d} B_i^d(x)\, B_j^d(y)\, B_k^d(z)\, b_{i,j,k}^{(l)}, \quad (x,y,z) \in \Omega^{(l)}, \tag{22}$$

with certain coefficients $b_{i,j,k}^{(l)}$. In addition, we generate a tensor–product Bézier representation of the associated gradients,

$$\nabla f^{(l)}(x,y,c) = \sum_{i,j,k=0}^{d} B_i^d(x)\, B_j^d(y)\, B_k^d(z)\, \mathbf{c}_{i,j,k}^{(l)} \quad \mathbf{x} \in \Omega^{(l)}. \tag{23}$$

Here, the control points $\mathbf{c}_{i,j,k}^{(l)}$ are obtained from the formulas for differentiation and degree elevation of tensor–product polynomials in Bernstein–Bézier form. Note that the components of the gradient have different degrees. Thus, degree elevation is needed in order to obtain the representation of the form (23).

Inequality (20) deals with gradients at two points with a certain maximum distance h. For each subdomain $\Omega^{(l)}$, we denote with $\mathcal{L}_h^{(l)}$ the indices of all subdomains that are within distance h of it,

$$i \in \mathcal{L}_h^{(l)} \iff \exists\, \mathbf{x} \in \Omega^{(l)}\ \exists\, \mathbf{y} \in \Omega^{(i)} :\ \|\mathbf{x}-\mathbf{y}\| \le h. \tag{24}$$

Geometrically, the set $\mathcal{L}_h^{(l)}$ contains the indices of all subdomains which have points within the offset surface (which consists of rectangular faces, segments of cylinders, and octants of spheres) at distance h of the boundary of $\Omega^{(l)}$. For instance, if $\Omega^{(l)}$ is a cube, the resulting offset surface is shown in Figure 9b.

Lemma 3. The inequalities (19) and (20) are valid with the following constants:

$$C = \max_{\substack{l=1,\dots,L \\ i,j,k=0,\dots,d}} \|\mathbf{c}_{i,j,k}^{(l)}\| \quad \text{and} \quad D_h = \min_{\substack{l_1=1,\dots,L;\ l_2 \in \mathcal{L}_h^{(l_1)} \\ i_1,i_2,j_1,j_2,k_1,k_2=0,\dots,d}} \mathbf{c}_{i_1,j_1,k_1}^{(l_1)} \cdot \mathbf{c}_{i_2,j_2,k_2}^{(l_2)}. \tag{25}$$

The proof results from the convex hull property of polynomials in Bernstein–Bézier form.

5.3. Bounding the footpoint distance

As a corollary, we bound the footpoint distance of a point without computing the associated footpoint.

Corollary 4. Consider again the situation of Lemma 2, and let (19) and (20) be satisfied. Choose the parameter h such that $h \geq C/D_h \, f(\mathbf{z})$. The distance of the point $\mathbf{z}$ from its footpoint $\mathbf{x}$ is bounded by

$$\|\mathbf{x} - \mathbf{z}\| \leq \frac{C}{D_h} \, |f(\mathbf{z})|. \tag{26}$$

The bound has the form (12) with the global constant $K = C/D_h$ which does not depend on the point p_i. Clearly, this result can be used only if the parameter h is not too small, as otherwise K will be too big. On the other hand, the smaller the parameter h, the bigger the lower bound D_h. This parameter acts as an initial guess of the maximum error.

If the point $\mathbf{z}$ approaches its footpoint $\mathbf{x}$, the bound (26) converges to zero, as $\mathbf{z} \to \mathbf{x}$ implies $f(\mathbf{z}) \to 0$.

An analogous bound can be formulated for algebraic spline curves, see [14]. The curve version of the bound improves the erroneous inequality (6.2) of [21]. Following that inequality, the distance can be bounded by $C \, f(\mathbf{z})$. This formula, however, is valid only if additional assumptions about the domain Ω are satisfied: the domain Ω has to contain both the point $\mathbf{z}$ and the point $\mathbf{x}^*$ on the curve (or surface) which is obtained by following the path of steepest descent, starting at $\mathbf{z}$, see Figure 9a. Obviously, the latter point is not guaranteed to be the footpoint $\mathbf{z}$.

In many cases, the error distribution is non–uniform, leading to isolated regions with larger errors. The bound obtained from the corollary can be used in order to guarantee an upper bound on the approximation error for the majority of the points, in regions with a relatively small error. Here, it can be used in order to avoid the footpoint computation via Newton-Raphson, which otherwise would be necessary. In the regions with a relatively large error, it will still be more appropriate to use footpoint computations, as this gives more accurate results.

Concluding remarks

We have described a novel technique for fitting implicitly defined algebraic spline surfaces to scattered data in 3D. By simultaneously approximating points and associated normal vectors, which are estimated from the data, one obtains a method which is both computationally simple, as the result is obtained by solving a sparse system of linear equations, and geometrically invariant, as no artificial normalization of the spline coefficients is needed. Weighted least–squares and an iterative adjustment of the normal vectors can be used in order to improve the initial result.

In principle, the method can be applied to data taken from arbitrary orientable two–dimensional manifolds in 3–space. Unlike the case of parametric surfaces, there is no limitation of the genus of the surface. However, as a crucial step in the surface fitting process, the estimation of surface normals has to be possible. Thus, there should be no creases in the data, the distribution of the data should be not too irregular, and the level of numerical noise should be relatively small.

Future research will address the issue of adding degrees of freedom locally, where they are needed. As another future project, we intend to integrate fair– and shape–preserving techniques into the framework of implicitly defined spline surfaces.

As part of ongoing research, we plan to combine the techniques described in this paper with the existing methods of parametric surface fitting, as follows. As a first step, we plan to fit an algebraic spline surface (with relatively few degrees of freedom) to the data. This surface will be used as an initial guess of a parametric spline surface consisting of parametric surface patches, pieced together with (approximate) geometrical continuity. It will be used to choose the non–linear parameters involved in the geometric continuity conditions, the segment boundaries, and in order to assign parameter values to the data. (This is related to the problem of parameterizing the algebraic spline surfaces, in order to generate representations which are compatible with existing CAD standards cf. [3]). Then, as a second step, the quality of the fit can be improved by locally adding further degrees of freedom to the parametric surface patches. The details and the implementation of this method are currently under way.

References

[1] C.L. Bajaj, Publications, `www.ticam.utexas.edu/CCV/papers/fhighlights.html`.

[2] C.L. Bajaj, I. Ihm and J. Warren, Higher–Order Interpolation and Least–Squares approximation Using Implicit Algebraic Surfaces, ACM Transactions on Graphics 12 (1993), 327–347.

[3] C.L. Bajaj and G. Xu, Spline approximations of real algebraic surfaces, J. Symb. Comput. 23 (1997), 315-333

[4] F. Bernardini, C.L. Bajaj, J. Chen and D.R. Schikore, Automatic Reconstruction of 3D CAD Models from Digital Scans, Int. J. Comp. Geom. Appl. 9 (1999), 327–370.

[5] W. Boehm and H. Prautzsch, *Numerical methods*, AK Peters, Wellesley (Mass.), and Vieweg, Braunschweig, 1993.

[6] M. Chen, A.E. Kaufman and R. Yagel (eds.), *Volume Graphics*, London, Springer 2000.

[7] G. Farin, *NURB curves and surfaces*, AK Peters, Wellesley (Mass.), 1995.

[8] A. Felis, Approximation von Meßpunkten mittels algebraischer Splineflächen, Diplomarbeit (Master thesis), Darmstadt University of Technology, Dept. of Mathematics, 2000.

[9] M.S. Floater and M. Reimers, Meshless parameterization and surface reconstruction, Computer Aided Geom. Des. 18 (2001), 77-92.

[10] D. Forsey and R. Bartels, Surface Fitting with Hierarchical Splines, ACM Transactions on Graphics 14 (1995), 134–161.

[11] M. Froumentin and C. Chaillou, Quadric surfaces: a survey with new results, in: *The Mathematics of Surfaces VII*, eds. T. Goodman and R. Martin, Information Geometers, Winchester, 363–382 (1997).

[12] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, AK Peters, Wellesley (Mass.), 1993.

[13] B. Jüttler, Surface fitting using convex tensor–product splines, J. Comp. Appl. Math. 84 (1997), 23–44.

[14] B. Jüttler, Bounding the Hausdorff Distance of Implicitly Defined and/or Parametric Curves, in: *Mathematical Methods in CAGD: Oslo 2000*, eds. T. Lyche and L.L. Schumaker, Vanderbilt University Press, Nashville TN, to appear.

[15] B. Jüttler, Least–squares fitting of algebraic spline curves via normal vector estimation, in: *The Mathematics of Surfaces IX*, eds. R. Cipolla and R.R.Martin, Springer, London 2000, 263–280.

[16] G.D. Koras and P.D. Kaklis, P.D., Convexity conditions for parametric tensor–product B-spline surfaces, Adv. Comput. Math. 10 (1999), 291–309.

[17] G. Meurant, *Computer solution of large linear systems*, Elsevier, Amsterdam 1999.

[18] L.A. Piegl and W. Tiller, Parametrization for surface fitting in reverse engineering, Computer-Aided Des. 33 (2001), 593–603.

[19] V. Pratt, Direct Least–Squares Fitting of Algebraic Surfaces, ACM Computer Graphics 21 (Siggraph 1987), 145–152.

[20] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer, New York, 1985.

[21] T.W. Sederberg, Planar piecewise algebraic curves, Computer Aided Geom. Des. 1 (1984), 241–255.

[22] R. Taubin, Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation, IEEE Trans. Pattern Analysis and Machine Intelligence 13 (1991), 1115–1138.

[23] M. Umasuthan, and A.M. Wallace, A comparative analysis of algorithms for fitting planar curves and surfaces defined by implicit polynomials, in: *Design and applications of Curves and Surfaces (Mathematics of Surfaces V)*, ed. R.B. Fisher, Clarendon Press, Oxford 1994, 495–514.

[24] R.J. Vanderbei, LOQO, `http://www.orfe.princeton.edu/~loqo`.

[25] N. Werghi et al., Object reconstruction by incorporating geometric constraints in reverse engineering, Comp. Aided Des. 31 (1999), 363–399.