Scattered Data Interpolation Using Data Dependant Optimization Techniques

Günther Greiner[1] Andreas Kolb[2] Angela Riepl[1]

[1] IMMD IX, Graphische Datenverarbeitung, Universität Erlangen, Germany
Email:{greiner,riepl}@informatik.uni-erlangen.de

[2] debis Systemhaus Engineering GmbH, CAD/CAM Department, Leinfelden, Germany
Email: askolb@str.daimler-benz.com

Abstract

Interpolation of scattered data has many applications in different areas. Recently, this problem has gained a lot of interest for CAD applications, in combination with the process of "reverse engineering", i.e. the construction of CAD models for existing objects.

Up until now, no method for Scattered Data Interpolation with a bivariate function produces surface formats that can be directly integrated into a CAD system. Additionally many of the existing interpolation schemes exhibit undesirable curvature distribution of the reconstructed surface.

In this paper we present a method for Scattered Data Interpolation producing tensor-product B-splines with high quality curvature distribution. This method first determines the knot vectors in a way that guarantees the *existence* of an interpolating B-spline. In a second step the degrees of freedom not specified by the interpolation constraints are automatically set using a *data dependent optimization technique.*

Examples demonstrate the quality of the resulting interpolants w.r.t. curvature distribution and approximation of known surfaces.

1 Introduction

Interpolation of discrete data by a curve (univariate problem) or a surface (bivariate problem), often called *Scattered Data Interpolation*, has many applications in different areas. This interpolation problem has been investigated for years and many methods have been developed dealing with this question. Recently, the problem of "reverse engineering" gained a lot of interest. The problem can briefly be stated as follows: A physical object (e.g. tool part, clay model) is digitized by some mechanical measuring device or a 3D laser scanner. Based on this discrete information a CAD model of the object has to be constructed. Scattered Data Interpolation (SDI) methods, i.e. the interpolation of scalar data specified over randomly scattered data points in the parameter plane, have the capacity to handle this reconstruction problem since they generate a continuous surface having a set of discrete points as input.

In the univariate case of the data interpolation problem a well developed theory and several very efficient methods are available. For example, cubic B-spline interpolation is very efficient and the approximation error is well-known (see [18]). In the bivariate case the situation is quite different. There exist many methods for SDI, (Shepard, Radial Basis Functions (both described in [10]), Nielson's Minimum Norm Network [15], the Oslo Mask Method[1] [1]). But in contrast to the univariate case, none of these methods has proven to be superior to the others. Moreover, none of these approaches constructs the interpolating function in a format that allows direct integration in a standard CAD/CAM systems. Other drawbacks (of some) of these methods are poor quality of the surface even for smooth data, bad approximation behavior and the restriction to relatively small data sets.

In this paper we introduce a SDI method which uses tensor product B–splines (TP B-splines) for data interpolation. The resulting surfaces show a good curvature distribution, for smooth data the quality of approximation is at least as good as for the classical methods. In the procedure, the most time consuming step is the solution to a sparse linear system whose size is proportional to the number of data points. This indicates, that the time complexity depends quadratically on the number of data points.

For functional surface reconstruction based upon SDI, the following problem is addressed:

Given: A finite set of *data points* $\mathbf{u}_i$, $i = 1, \dots, N$ in $\mathbf{R}^2$. Furthermore, for each data point $\mathbf{u}_i$ we have a *data value* $f_i \in \mathbf{R}$.

Goal: A fair TP B–spline function F that interpolates the given data values at all data points, i.e.:

$$F(\mathbf{u}_i) = f_i.$$

In case of univariate interpolation with B-spline curves there exists a unique characterization of the solvability of the interpolation problem based on the location of the data points w.r.t. the knots (the *Schoenberg-Whitney Theorem*; see [4]).

In the bivariate case no such unique characterization exists. There are sufficient characterizations which severely restrict the positions of the data points and thus are not comprehensive in the general situation. Trying to find appropriate knot vectors for the parameter space in order to fulfill uniquely the interpolation constraints is therefore a very difficult task and is most likely to fail.

We choose the knot vectors in a way, that only the *existence* of a interpolating TP-B-splines is guaranteed. This can always be achieved by using a sufficiently fine knot spacing. The degrees of freedom (DOFs) not set by the interpolation constraints are determined using an *optimization technique*:

> Given a cost (or *fairness*) functional J measuring certain geometric properties of a surface (e.g. total curvature), we are looking for a surface F_0 that interpolates the data and which has optimal behavior w.r.t. this properties:
>
> $$J(F_0) \leq J(F) \text{ for all } F \text{ satisfying } : F(\mathbf{u}_i) = f_i,\ i = 1, \dots, N. \qquad (1.1)$$

With help of this variational approach the remaining DOFs of the TP-B-spline surface are determined automatically.

The usage of functionals for surface fairing and design is very popular (see [2, 3, 11, 19]). In [17] a similar approach is used to handle a SDI problem based upon piecewise triangular Bézier patches using the so-called *simplified Thin Plate Energy* functional as fairness functional (see Section 2).

Our approach has quite a few advantages compared with the other interpolation techniques:

[1]The scheme presented by Arge, Dæhlen und Tveito does only construct an exact interpolant unless all data points coincide with points of the regular grid of the discrete function space used in the method.

- The resulting surface is given in a CAD-compatible format (TP B-spline).
- There are no restriction on the size or the structure of the data set.
- Carefully choosing the cost functional J will guarantee a good curvature distribution of the interpolating surfaces.
- The interpolation scheme can be extended to the case of interpolation of 3D point clouds.

The choice of the fairness functional has a great impact on the method. The quality of the resulting surfaces (harmonic curvature distribution), as well as the efficiency of the procedure will heavily depend on J. A trade-off between these two extremes is necessary in each specific application context.

This paper is organized as follows: In the following section the design of fairness functionals is discussed. The algorithm to solve the SDI problem based on the optimization technique is presented in Section 3. In Section 4 some examples are given demonstrating the quality of the presented scheme. Conclusions and outlines to future work are presented in Section 5.

2 Design of fairness functionals

In this section we describe fairness functionals that have been widely used in the context of surface fairing. Furthermore we discuss the concept of *data dependent* functionals. In particular, we introduce a new type of functional: the *data dependent Thin-Plate-Energy.*

As mentioned above, the variational approach to surface design problems based on fairness functionals has become very popular. This is due to the fact that it offers the opportunity to handle automatically DOFs not set by the user, i.e. not uniquely determined by the constraints that have been specified in order to obtain a certain geometric configuration.

Not surprisingly, there is a tradeoff between quality and time: Using the fairness functional which measures the total curvature exactly[2]:

$$J_{exact}(\mathbf{F}) = \int_{\Omega} \kappa_1^2 + \kappa_2^2 \, d\omega_{\mathbf{F}}, \tag{2.2}$$

the process of finding an optimal solution to (1.1) involves very time-consuming numerical algorithms, even for small data sets. This is due to the fact that J_{exact} is a highly nonlinear functional. However, concerning the quality of the resulting surface this functional performs very well (see [14]).

Quite opposite is the situation when simple quadratic functionals are used. In fact, minimizing quadratic functionals is a relatively easy task, one (only) has to set up the *normal equations* and then solve this linear system. Thus surface design methods based on the widely used simplified Thin-Plate-Energy

$$J_{simple}(\mathbf{F}) = \int_{\Omega} (\mathbf{F}_{uu})^2 + 2\,(\mathbf{F}_{uv})^2 + (\mathbf{F}_{vv})^2 \; dudv. \tag{2.3}$$

are very efficient. For small data sets results can be obtained in real time. The disadvantage of this functional is that it does not represent a geometric quantity. Only for nearly flat surfaces it is a good approximation to $J_{exact}(\mathbf{F})$. As a consequence, for flat surfaces it produces good results, but for highly curved shapes optimizing this functional may not lead to fair surfaces.

[2] $\kappa_{1,2}$ denote the *principal curvatures* of the surface defined by $\mathbf{F}$; $d\omega_{\mathbf{F}}$ denotes the area element of $\mathbf{F}$.

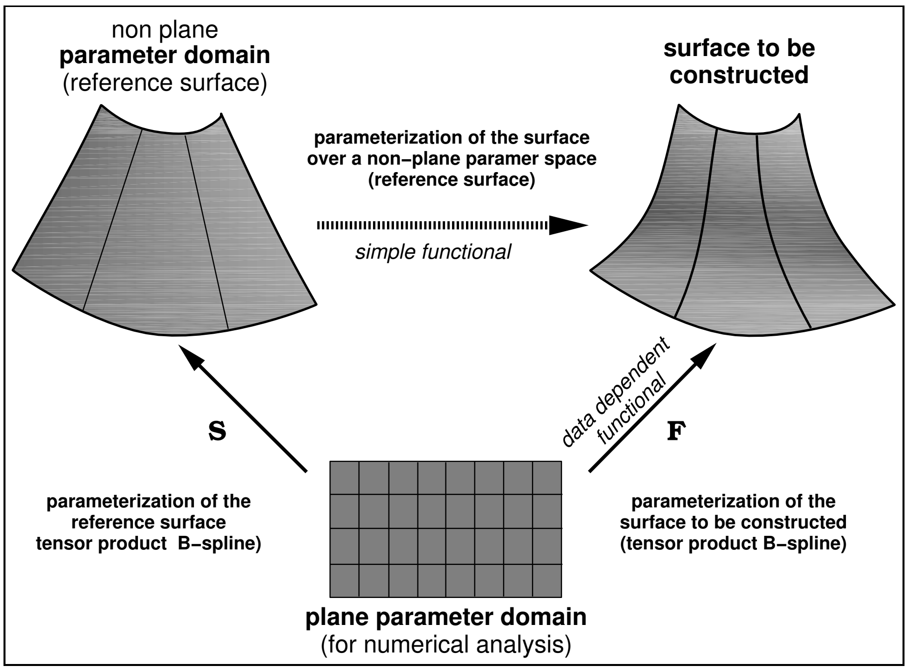
non plane
parameter domain
(reference surface)
surface to be constructed
parameterization of the surface over a non-plane paramer space (reference surface)
simple functional
data dependent functional
S
F
parameterization of the reference surface tensor product B-spline)
parameterization of the surface to be constructed (tensor product B-spline)
plane parameter domain
(for numerical analysis)

Figure 2.1: A simple functional defined over a reference surface leads to a data dependent functional on the plane parameter space of the reference surface

A very successful combination of good approximation and performance are the so-called *data dependent* functionals. In [8, 9] data dependent approximations to the functional measuring the square of the mean curvature have been used for the construction of blend surfaces and for surface fairing. The basic idea of this concept is the following (see Figure 2.1): Use a *reference surface* **S** as non-plane parameter space. Consider functions/surfaces to be parameterized over **S** and define the simple quadratic functionals on the non-plane parameter space. In order to be able to solve the minimization problem numerically, the problem is transformed by standard methods (change of variables) to the parameter domain of the reference surface. Thus leading to a quadratic functional, whose coefficients depend on the geometry of the reference surface **S**. If the reference surface is carefully chosen, the resulting functional is a good approximation to an exact curvature functional.

In the following, we develop a data dependent *Thin-Plate-Energy* functional (briefly TPE functional) which we will use as cost functional in our algorithm to determine an interpolant.

Consider the simple TPE functional (2.3). This functional can be expressed with help of the *Hessian* Hess of **F** or alternatively using the *gradient* grad (**F**) and the differential $D(\vec{\mathbf{t}})$ of a vector field $\vec{\mathbf{t}}(u,v) = (t_1(u,v), t_2(u,v))$:

$$(\mathbf{F}_{uu})^2 + 2\,(\mathbf{F}_{uv})^2 + (\mathbf{F}_{vv})^2 = \mathsf{tr}\ (\mathsf{Hess}\,\mathbf{F})^2 = \mathsf{tr}\ (D\,(\mathsf{grad}\,\mathbf{F}))^2\,. \tag{2.4}$$

Here $\mathsf{tr}\,(M)$ denotes the *trace* of a square matrix $M = (m_{ij})$, $\mathsf{tr}\,(M) = \sum_i m_{ii}$.

The following definition gives the generalizations for the gradient and the derivative of a vector field for functions resp. vector fields defined on the reference surface **S** (see [12]).

Definition 2.1 *Let* $\mathbf{S} : \Omega \to \mathbf{R}^3$, $(\Omega \subseteq \mathbf{R}^2)$ *denote a surface with first fundamental form* $\mathbf{I_S} = \big(g_{ij}\big)$. *We assume that* **S** *is regular, i.e.* $\det(\mathbf{I_S}) = g_{11}g_{22} - g_{12}^2 \neq 0$, *hence the inverse* $\mathbf{I_S}^{-1} = \big(g^{ij}\big)$ *exists.*

(i) The gradient $\mathsf{grad}_{\mathbf{S}}(h)$ *of a scalar-valued function* $h : \Omega \to \mathbf{R}$ *on the surface* **S** *is defined*

as[3]

$$\mathsf{grad}\,_{\mathbf{S}}(h) = (h_u, h_v)\mathbf{I_S}^{-1}.$$

(ii) *The (covariant) derivative of the tangential vector field* $\vec{\mathbf{t}}(u,v) = (t_1(u,v), t_2(u,v))$ *is defined as*[4]*:*

$$\nabla_{\mathbf{S}}(\vec{\mathbf{t}}) = \begin{pmatrix} (t_1)_u & (t_1)_v \\ (t_2)_u & (t_2)_v \end{pmatrix} + t_1 \begin{pmatrix} \Gamma^1_{11} & \Gamma^1_{12} \\ \Gamma^1_{21} & \Gamma^1_{22} \end{pmatrix} + t_2 \begin{pmatrix} \Gamma^2_{11} & \Gamma^2_{12} \\ \Gamma^2_{21} & \Gamma^2_{22} \end{pmatrix}.$$

Here Γ^k_{ij}, $i,j,k = 1,2$ *denote the* Christoffel symbols. *They represent the coordinates of the normal-projections of the second derivatives of* $\mathbf{S}$ *onto the tangent plane in the* $\{\mathbf{S}_u, \mathbf{S}_v\}$*-basis, e.g.*

$$\mathbf{S}_{uu} = \Gamma^1_{11}\mathbf{S}_u + \Gamma^2_{11}\mathbf{S}_v + \langle\, \mathbf{S}_{uu} \mid \vec{\mathbf{N}}\,\rangle \vec{\mathbf{N}}. \tag{2.5}$$

In the following remark we give some more details for the Christoffel symbols (see [12, 13]).

Remark 2.2 *(i) The Christoffel symbols for a surface* $\mathbf{S}$ *are given by*

$$\begin{pmatrix} \Gamma^1_{ij} \\ \Gamma^2_{ij} \end{pmatrix} = \mathbf{I_S}^{-1} \begin{pmatrix} \langle\, \mathbf{S}_{u_i u_j} \mid \mathbf{S}_{u_1}\,\rangle \\ \langle\, \mathbf{S}_{u_i u_j} \mid \mathbf{S}_{u_2}\,\rangle \end{pmatrix} \qquad (\textit{here}\ \ u_1 = u,\ u_2 = v).$$

(ii) *The Christoffel symbols can be calculated in terms of the first fundamental form and its first order derivatives and thus they depend only on the* inner geometry *of the surface* $\mathbf{S}$ *(see [12, Proposition 3.8.1]).*

(iii) *The Christoffel symbols are invariant under isometric transformations.*

Combining these generalizations for the gradient and for the derivative of a vector field, we can define the Hessian $\mathsf{Hess}\,_{\mathbf{S}}$ for a function h on the surface $\mathbf{S}$ via the relation[5]

$$\mathsf{Hess}\,_{\mathbf{S}}(h)(\vec{\mathbf{a}}, \vec{\mathbf{b}}) = \langle\, \nabla_{\mathbf{S}}\,(\mathsf{grad}\,_{\mathbf{S}}(h))\,\vec{\mathbf{a}} \mid \vec{\mathbf{b}}\,\rangle_{\mathbf{I_S}} \quad \forall\, \vec{\mathbf{a}}, \vec{\mathbf{b}} \in \mathbf{R}^2. \tag{2.6}$$

Considering $\mathsf{Hess}\,_{\mathbf{S}}(h)$ as linear mapping in the tangent space we obtain the following representation as 2×2-matrix

$$\mathsf{Hess}\,_{\mathbf{S}}(h) = \mathbf{I_S}^{-1} \left[\begin{pmatrix} h_{uu} & h_{uv} \\ h_{uv} & h_{vv} \end{pmatrix} - h_u \begin{pmatrix} \Gamma^1_{11} & \Gamma^1_{12} \\ \Gamma^1_{21} & \Gamma^1_{22} \end{pmatrix} - h_v \begin{pmatrix} \Gamma^2_{11} & \Gamma^2_{12} \\ \Gamma^2_{21} & \Gamma^2_{22} \end{pmatrix} \right]. \tag{2.7}$$

Analogously as for the simple thin plate energy functional, the data dependent thin plate energy functional for the surface $\mathbf{F} : \Omega \to \mathbf{R}^3$ is defined using the generalized Hessian w.r.t. the reference surface $\mathbf{S}$:

$$J_{\mathbf{S}}(\mathbf{F}) = \int_\Omega \sum_{i=1}^{3} \mathsf{tr}\ (\mathsf{Hess}\,_{\mathbf{S}}(\mathbf{F}_i))^2 \ d\omega_{\mathbf{S}}. \tag{2.8}$$

In our situation, the surface $\mathbf{F}$ is functional, i.e. $\mathbf{F}(u,v) = (u, v, F(u,v))$. Thus $J_{\mathbf{S}}(F) = J_{\mathbf{S}}(\mathbf{F}) = \int_\Omega \mathsf{tr}\ (\mathsf{Hess}\,_{\mathbf{S}}(F))^2 \ d\omega_{\mathbf{S}} + \textit{const}$ where the constant only depends on $\mathbf{S}$.

In case this approach might look somehow arbitrary, the following theorem gives a justification for the above choice of $J_{\mathbf{S}}$.

[3]The value $h(u,v)$ is identified with the value of h at $\mathbf{S}(u,v)$.

[4]To the surface point $\mathbf{S}(u,v)$ the tangent vector $t_1\mathbf{S}_u + t_2\mathbf{S}_v$ is assigned.

[5]$\langle \cdot \mid \cdot \rangle_{\mathbf{I_S}}$ denotes the inner product induced by the Riemannian metric.

Theorem 2.3 *For a two times differentiable parameterized surface* $\mathbf{S}$*, the functional* $J_{\mathbf{S}}$ *defined in* (2.8) *has the following properties.*

a) $J_{\mathbf{S}}$ *is a quadratic functional.*

b) $J_{\mathbf{S}}$ *is a positive semi-definite functional with a null space having dimension* ≤ 9*. For a non-developable reference surface* $\mathbf{S}$*, the null space has dimension* 3.

c) $J_{\mathbf{S}}(\mathbf{S}) = J_{exact}(\mathbf{S})$.

d) $J_{\mathbf{S}}$ *does not depend on the specific parameterization of the surface* $\mathbf{S}$.

Proof: a) From (2.7) it follows that for fixed $\mathbf{S}$ $\mathsf{Hess}_{\mathbf{S}}(\mathbf{F}_i)$ linearly depends on the first and second order partial derivatives of $\mathbf{F}_i$. Thus $\left(\mathsf{Hess}_{\mathbf{S}}(\mathbf{F}_i)\right)^2$ quadratically depends on these derivatives and so does its trace.

b) Semi-definiteness is clear from the definition of $J_{\mathbf{S}}$ (note that the trace of the square of a symmetric matrix A is always ≥ 0.) Moreover, assuming that $J_{\mathbf{S}}(\mathbf{F}) = 0$, then $\mathsf{Hess}_{\mathbf{S}}(\mathbf{F}_i) = 0$. This means that the covariant derivative of the vector field $\mathsf{grad}_{\mathbf{S}}(\mathbf{F}_i)$ vanishes. Therefore the vector field $\mathsf{grad}_{\mathbf{S}}(\mathbf{F}_i)$ is composed by parallel displacements (see [12, 13]) of a unique tangent vector on $\mathbf{S}$. Such a vector can be specified by two coordinates. Finally, when integrating the gradient $\mathsf{grad}_{\mathbf{S}}(\mathbf{F}_i)$ to obtain $\mathbf{F}_i$, there is an additional constant, the value of $\mathbf{F}_i$ at a single point. Thus for each component $\mathbf{F}_i$ we have three degrees of freedom. The additional statement follows from the fact that non-zero parallel vector fields can exist only on developable surfaces (see [13]).

c) We make use of the *Einstein summation convention* which says that in an expression we sum over all indices which appear twice.

By (2.7) we have $\mathsf{Hess}_{\mathbf{S}}(\mathbf{S}_i) = \left(g^{kl}((\mathbf{S}_i)_{u_j u_l} - (\mathbf{S}_i)_{u_p}\Gamma^p_{jl})\right)_{kj}$[6]. Using (2.5) we can replace $(\mathbf{S}_i)_{u_j u_l}$ and obtain

$$\mathsf{Hess}_{\mathbf{S}}(\mathbf{S}_i) = \left(g^{kl}(\Gamma^p_{jl}\,(\mathbf{S}_i)_{u_p} + \langle\, \mathbf{S}_{u_j u_l} \mid \vec{\mathbf{N}}\,\rangle\, \vec{\mathbf{N}}_i - (\mathbf{S}_i)_{u_p}\,\Gamma^p_{jl})\right)_{kj} = \left(g^{kl}\, h_{jl} \vec{\mathbf{N}}_i\right)_{kj} = \left(g^{kl}\, h_{jl}\right)_{jk} \cdot \vec{\mathbf{N}}_i\,.$$

where $(h_{jl}) = \langle\, \mathbf{S}_{u_j u_l} \mid \vec{\mathbf{N}}\,\rangle$ is the second fundamental form $\mathbf{II}_{\mathbf{S}}$ of $\mathbf{S}$.

Now $g^{kl}\, h_{jl} = g^{kl}\, h_{lj}$ is the (k,j)-th entry of the product of the matrices ${\mathbf{I}_{\mathbf{S}}}^{-1} = (g^{ij})_{ij}$ and $\mathbf{II}_{\mathbf{S}} = (h_{ij})_{ij}$. Therefore we have

$$\mathsf{Hess}_{\mathbf{S}}(\mathbf{S}_i) = {\mathbf{I}_{\mathbf{S}}}^{-1}\mathbf{II}_{\mathbf{S}} \cdot \vec{\mathbf{N}}_i\;. \tag{2.9}$$

$\mathbf{W} = {\mathbf{I}_{\mathbf{S}}}^{-1}\mathbf{II}_{\mathbf{S}}$ is the Weingarten map, whose eigenvalues are the principal curvatures κ_1, κ_2 (see [12, Proposition 3.5.2]. $\mathbf{W}^2$ has eigenvalues κ_1^2 and κ_2^2, hence $\mathsf{tr}\,(\mathbf{W}^2) = \kappa_1^2 + \kappa_2^2$.

Since $\sum_i (\vec{\mathbf{N}}_i)^2 = 1$, we conclude $\kappa_1^2 + \kappa_2^2 = \mathsf{tr}\,(\mathbf{W}^2) = \sum_i (\mathsf{tr}\,\left((\mathbf{W} \cdot \vec{\mathbf{N}}_i)^2\right)$ and the assertion follows from (2.2) and (2.8).

d) Both, the gradient and the covariant derivative are *geometric invariants* (see [12, Lemma 4.1.5]), that is, they do not depend on a specific parameterization of $\mathbf{S}$. Since our construction only uses these notions, $J_{\mathbf{S}}$ is independent of the parameterization of $\mathbf{S}$ as well. ■

[6]By Einstein's summation convention $\left(g^{kl}((\mathbf{S}_i)_{u_j u_l} - (\mathbf{S}_i)_{u_p}\Gamma^p_{jl})\right) = \sum_l \left(g^{kl}((\mathbf{S}_i)_{u_j u_l} - \sum_p (\mathbf{S}_i)_{u_p}\Gamma^p_{jl})\right)$

Discussion and consequences of the theorem.

- At a first glance $J_{\mathbf{S}}$ looks quite complicated. However, since it is quadratic (by assertion a)) it can be minimized very easily, much faster than J_{exact}. In fact, the minimum of such a functional can be obtained as the solution of a linear system.
- A consequence to b) is that only few constraints are necessary in order to guarantee a unique solution to a constraint optimization problem for $J_{\mathbf{S}}$. In general, specifying 3 (vector) constraints, will lead to a positive definite problem. For example, it is enough, to specify for one parameter value the position and two directional derivatives. Alternatively, specifying the position at at least three different parameter values will also lead to a unique minimum. When a non-developable surface is used as reference surface, only one interpolation condition is enough to ensure a unique solution.
- Assertion c) implies that $J_{\mathbf{S}}$ is (locally) a good approximation to J_{exact}, thus guaranteeing that the minimization process really leads to surfaces of fair shape. The argument is as follows. The functional $J_{\mathbf{S}}$ depends in a continuous way on the reference surface. This means that when $\mathbf{F}$ is close to $\mathbf{S}$ (in the appropriate Sobolev space), then $J_{\mathbf{S}} \approx J_{\mathbf{F}}$, hence we have $J_{exact}(\mathbf{F}) = J_{\mathbf{F}}(\mathbf{F}) \approx J_{\mathbf{S}}(\mathbf{F})$.
- Finally assertion d) is important as well. When designing a surface, usually the designer a priori knows how the surface will approximately look like. Thus he can specify (the geometry of) a reference surface $\mathbf{S}$. However, it is not clear, how one has to parameterize this reference surface. In view of assertion d) this is not a problem. No matter how $\mathbf{S}$ is parameterized, $J_{\mathbf{S}}$ will not depend on the parameterization, but solely on its geometry.

3 The Algorithm

In this section we describe the algorithm that is used to determine an interpolating bicubic TP-B-spline surface F_0 having minimal value w.r.t the given fairness functional $J_{\mathbf{S}}$, i.e.:

$$J_{\mathbf{S}}(F_0) \leq J_{\mathbf{S}}(F) \text{ for all } F : F(\mathbf{u}_i) = f_i,\ i = 1, \dots, N.$$

$J_{\mathbf{S}}$ is the data dependent TPE functional introduced in the previous section (see 2.8). For $\mathbf{S}$ being the identity, i.e. $\mathbf{S}(u,v) = (u, v, 0)$ we obtain the simplified TPE (2.3) as a special case.

We introduce the following notation for bicubic TP B-spline basis functions defined over the knot vectors $U = \{u_{-2}, \dots, u_{n+2}\}$ and $V = \{v_{-2}, \dots, v_{m+2}\}$:

> $B_{i,j}$ denotes the bicubic B-spline basis function corresponding to the knot indices i, j; $B_{i,j}$ is supposed to have it's support symmetric to (u_i, v_j), i.e. $\text{supp}(B_{i,j}) = [u_{i-2}, u_{i+2}] \times [v_{j-2}, v_{j+2}]$.

The algorithm consists of the following four steps (see Figure 3.2):

Step 1: Define the parameter area of the TP-B-spline surface as the bounding box $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$ of all data points $\mathbf{u}_i$, $i = 1, \dots, N$.

Step 2: Determine uniform knot vectors U and V with knot spacing $h_u = u_{i+1} - u_i$ and $h_v = v_{j+1} - v_j$ in u- and v-direction respectively.

The knot vectors have to be chosen in a way such that there exists an interpolating TP-B-spline surface over the corresponding rectangular grid (see Section 3.1).

Step 3: If a data dependent functional is used, an appropriate reference surface has to be determined (see Section 3.2).

Step 4: Find among all interpolating TP-B-spline defined over the knot grid $U \times V$ the one which minimizes the fairness functional $J_{\mathbf{S}}$ (see Section 3.3).

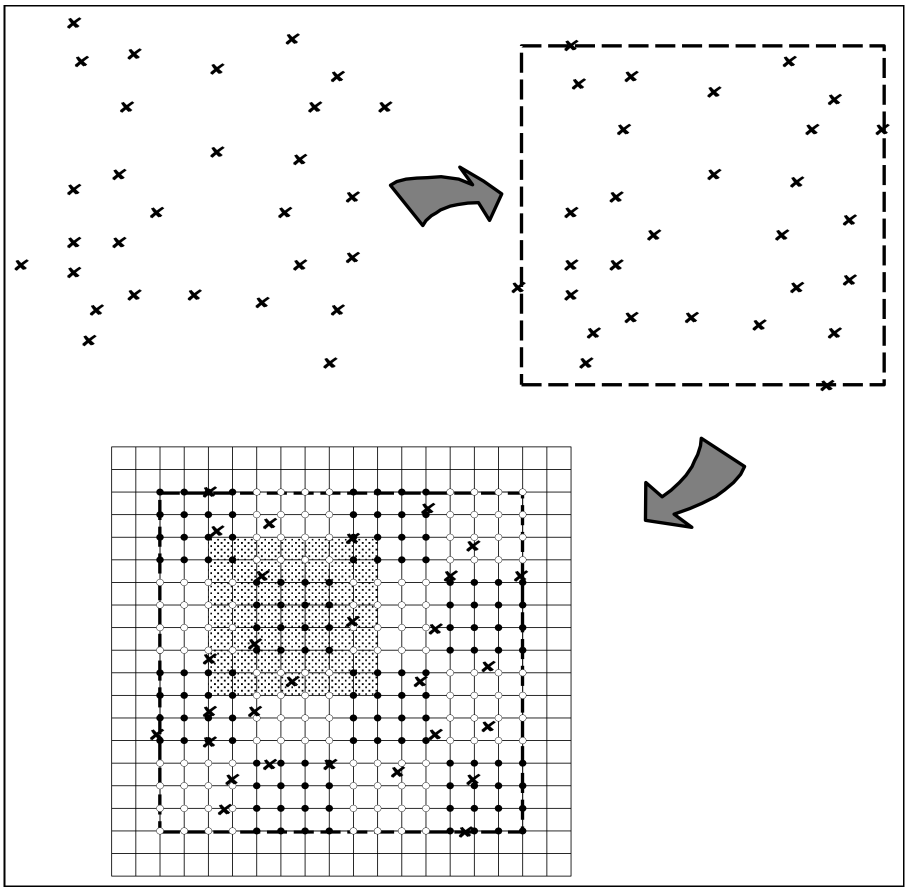

Figure 3.2: Overview of the algorithm: The data points are boxed and an adequate knot grid is generated. Uniform knot vectors for $n = m = 15$ have been chosen and the index sets have been set to $\{0, 3, 7, 11, 15\}$ for both directions (see Equation (3.10)). The resulting local interpolation problems $IP^{loc}_{k,l}$ are solvable (see Equation (3.12).

Step 1 needs no further explanation. In the following sections we describe the details of Steps 2, 3 and 4.

3.1 Knot Vectors (Step 2)

In this section we describe a way to construct appropriate knot vectors U and V, i.e. knot vectors for which a TP-B-spline surface exists that interpolates the given data.

First of all we define the grid such that all data points $\mathbf{u}_i$ lie in the *inner region* $[u_1, u_{n-1}] \times [v_1, v_{m-1}]$ of the parameter region, i.e.:

$$u_{min} = u_1, \quad u_{max} = u_{n-1}, \quad v_{min} = v_1, \quad v_{max} = v_{m-1}.$$

The $B_{i,j}$'s ($0 \leq i \leq n$, $0 \leq j \leq m$) defined over the knot grid $U \times V$ form a basis of the space of TP-surfaces generated by C^2 piecewise polynomials over the rectangle $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$.

We state that there always exist knot vectors for which the corresponding space of TP-B-spline surfaces contains a surface F satisfying $F(\mathbf{u}_i) = f_i$, $i = 1, \dots, N$. This result is obvious since we always can define a knot grid such that each support $\mathrm{supp}(B_{i,j})$ contains at most one data point $\mathbf{u}_i$. However, this approach yields a number of knots that is far to large for practical use.

In the following we give a generalization of this simple result which is much more appropriate in our situation. This generalization is based on *local interpolation problems*. Therefore we consider sets of increasing indices[7]:

$$\{i_0, \dots, i_p\} \text{ where } 0 = i_0 < i_1 < \dots < i_{p-1} < i_p = n \tag{3.10}$$
$$\{j_0, \dots, j_q\} \text{ where } 0 = j_0 < j_1 < \dots < j_{q-1} < j_q = m \tag{3.11}$$

.

The local interpolation problem $IP^{loc}_{k,l}$ is now formulated as follows:

Find the B-spline surface:

$$F^{loc}_{k,l}(u,v) = \sum_{\substack{i_k \le i < i_{k+1} \\ j_l \le j < j_{l+1}}} b_{i,j} B_{i,j}, \tag{3.12}$$

satisfying $F^{loc}_{k,l}(\mathbf{u}_i) = f_i, \forall i \in I_{k,l}$, where $I_{k,l}$ contains the indices of all data points lying within the support of $F^{loc}_{k,l}$, i.e.:

$$I_{k,l} = \{\mathbf{u}_1, \dots, \mathbf{u}_N\} \cap [u_{i_k-2}, u_{i_{k+1}+1}] \times [v_{j_l-2}, v_{j_{l+1}+1}].$$

The following theorem motivates the introduction of the local interpolation problems $IP^{loc}_{k,l}$.

Theorem 3.1 *If all local interpolation problems $IP^{loc}_{k,l}$ are solvable, then there exists a TP-B-spline surface F satisfying $F(\mathbf{u}_i) = f_i, i = 1, \dots, N$.*

Proof: But assumption $IP^{loc}_{0,0}$ is solvable. For $IP^{loc}_{1,0}$ we consider the data values:

$$f_i^{1,0} = \begin{cases} 0 & \text{if } i \in I_{0,0} \\ f_i & \text{otherwise} \end{cases}.$$

Thus $F^{loc}_{1,0}$ and $F^{loc}_{0,0} + F^{loc}_{1,0}$ satisfies the interpolation constraints for the data points $\mathbf{u}_i \in I^{loc}_{0,0}$ and $\mathbf{u}_i \in I^{loc}_{0,0} \cup I^{loc}_{1,0}$ respectively.

Proceeding this way, the resulting control points $b_{i,j}$ after solving the local interpolation problem $IP^{loc}_{i_{k-1},j_{l-1}}$ define an interpolating TP-B-spline surface F. ■

To check the solvability of the local interpolation problems $IP^{loc}_{k,l}$, we investigate the system of linear equations:

$$\underbrace{\left(B_{i_k,j_l}(\mathbf{u}_i) \cdots B_{i_{k+1}-1,j_{l+1}-1}(\mathbf{u}_i)\right)_{i \in I_{k,l}}}_{=L_{k,l}} \begin{pmatrix} b_{i_k,j_l} \\ \vdots \\ b_{i_{k+1}-1,j_{l+1}-1} \end{pmatrix} = (f_i)_{i \in I_{k,l}}. \tag{3.13}$$

[7]This partition of the knot vectors is arbitrary but fix.

If $L_{k,l}$ has full rank the local interpolation problem $IP_{k,l}^{loc}$ has a solution for arbitrary data values f_i.

In our implementation we always use a fixed block-size of 4×4 neighboring control points. Thus we have to assure that the number of control points in u- and v-direction is a multiple of 4.

In order to define a knot grid we start with initial values for n and m satisfying: $(n + 1)/4$, $(m + 1)/4 \in \mathbf{N}$. This initial knot grid is refined until all local interpolation problems $IP_{k,l}^{loc}$ are solvable. The step of refinement simply sets

$$h_u \leftarrow \frac{h_u}{2} \quad \text{and} \quad h_v \leftarrow \frac{h_v}{2}.$$

We found that $|I_{k,l}| \leq 7$ is a good precondition in order to guarantee the solvability of $IP_{k,l}^{loc}$. Thus we first refine the knot grid until $|I_{k,l}| \leq 7$ is met for all local interpolation problems. After that we check the rank of matrix $L_{k,l}$. Only in few cases we have to further refine the knot grid.

3.2 Reference surface (Step 3)

If we use a data dependent functional, we have to provide a reference surface $\mathbf{S}$ whose first fundamental form is used to define the functional. In our situation it is very natural to define $\mathbf{S}$ as functional surface, i.e., $\mathbf{S}(u, v) = (u, v, S(u, v))$ using the *functional* reference surface S.

The reference surface has to be chosen under the following considerations:

- S should have roughly the same shape as the resulting interpolating surface,
- S should be fair,
- efficient algorithms for the generation as well as for the evaluation the first fundamental form of S are required.

We decide to use a TP-B-spline surface applying a *least-squares-fit* to the given interpolation data. The knot grid $U' \times V'$ with $U' = \{u'_{-2}, \ldots, u'_{n'+2}\}$ and $V' = \{v'_{-2}, \ldots, v'_{m'+2}\}$ of S is chosen such that the inner region coincides with the bounding box of the data points $\mathbf{u}_i$:[8]

$$S = \sum_{\substack{0 \leq i \leq n' \\ 0 \leq j \leq m'}} c_{i,j} B'_{i,j}.$$

Least-squares fitted TP-B-spline surfaces tend to oscillate near the boundary if the data implies a rather non-uniform distribution of curvature. Therefore we use a *weighted* least-squares-fit instead of an ordinary one where more weight is put on the data points $\mathbf{u}_i$ that are close to the boundary:

$$\sum_{k=1}^{N} \omega_k \left\| \sum_{\substack{0 \leq i \leq n' \\ 0 \leq j \leq m'}} c_{i,j} B'_{i,j}(\mathbf{u}_k) - f_k \right\| \longrightarrow \min.$$

In our implementation we set ω_k to be the inverse distance of $\mathbf{u}_k$ to the boundary of the parameter space of the B-spline surface.

[8]Note that the B-spline basis functions $B'_{i,j}$ are defined over a different grid for F and S.

We found that $n' = m' \approx \sqrt{N/2}$ results in least-squares fitted surfaces that are well balanced considering smoothness and "shape-approximation".

3.3 Variational Statement (Step 4)

For a given knot grid $U \times V$ and functional $J_{\mathbf{S}}$, the problem of finding the optimal TP-B-spline surface w.r.t. $J_{\mathbf{S}}$ is handled with the method of Lagrangian multipliers:

Minimize $J_{\mathbf{S}}\left(\sum_{\substack{0\le i\le n\\0\le j\le m}} b_{i,j}B_{i,j}\right)$ w.r.t. $\sum_{\substack{0\le i\le n\\0\le j\le m}} b_{i,j}B_{i,j}(\mathbf{u}_k) = f_k, k = 1,\dots,N,$ is equivalent to solving

$$\begin{pmatrix} A & P^t \\ P & 0 \end{pmatrix}\begin{pmatrix} \mathbf{b} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{f} \end{pmatrix}, \tag{3.14}$$

where for $F = \sum_{\substack{0\le i\le n\\0\le j\le m}} b_{i,j}\mathsf{B}_{n,m}$

$$A\mathbf{b} = \begin{pmatrix} \frac{\partial J_{\mathbf{S}}(F)}{\partial b_{0,0}} \\ \vdots \\ \frac{\partial J_{\mathbf{S}}(F)}{\partial b_{i,j}} \end{pmatrix} \quad \text{and} \quad P\mathbf{b} = \begin{pmatrix} F(\mathbf{u}_1) \\ \vdots \\ F(\mathbf{u}_N) \end{pmatrix} = \mathbf{f}.$$

$\mathbf{b}, \mathbf{f}$ denote the vector-notation for all $b_{i,j}$ and f_k. The entries of $\mathbf{d}$ are the Lagrangian multipliers.

The number of unknown control points $b_{i,j}$ and multipliers d_i can be relatively large. Thus using direct methods to solve equation (3.14) is very time-consuming. The usage of standard iterative methods like Gauss-Seidel or SOR is not possible due to the 0's in the lower-left block of the matrix.

We handle this problem by rearranging the equations and the unknowns such that a block-Gauss-Seidel or block-SOR can be used to solve (3.14). We use again the 4×4-blocks of neighboring B-spline basis functions of Step 2 and associate each data point $\mathbf{u}_i$ with exactly one of those blocks by decomposing the parameter space in regions

$$\mathcal{R}_{k,l} = [u_{i_k} - (\tfrac{h_u}{2}), u_{i_{k+1}} - \tfrac{h_u}{2}[\times[v_{j_l} - \tfrac{h_v}{2}, v_{j_{l+1}} - \tfrac{h_v}{2}[.$$

Thus each data point $\mathbf{u}_i$ lies within exactly one region $\mathcal{R}_{k,l}$.

Now we consider the following local optimization problem OP $OP_{k,l}^{loc}$ that relates to the local interpolation problems $IP_{k,l}^{loc}$:

Find among all B-spline surfaces

$$F_{k,l}^{loc}(u,v) = \sum_{\substack{i_k\le i< i_{k+1}\\ j_l\le j< j_{l+1}}} b_{i,j}B_{i,j}(u,v),$$

satisfying $F_{k,l}(\mathbf{u}_i) = f_i, \forall \mathbf{u}_i \in \mathcal{R}_{k,l}$ the one with minimal value w.r.t. the fairness functional $J_{\mathbf{S}}$.

The local optimization problems $OP_{k,l}^{loc}$ can also be handled with the method of Lagrangian multipliers by solving

$$\underbrace{\begin{pmatrix} A_{k,l} & P_{k,l}^t \\ P_{k,l} & 0 \end{pmatrix}}_{=C_{k,l}} \begin{pmatrix} \mathbf{b}_{k,l} \\ \mathbf{d}_{k,l} \end{pmatrix} = \begin{pmatrix} * \\ \mathbf{f}_{k,l} \end{pmatrix}. \tag{3.15}$$

When calculating the derivatives of $J_{\mathbf{S}}(F)$, the unconsidered $B_{i,j}$ with $B_{i,j}(\mathbf{u}_r) \neq 0$ for any $\mathbf{u}_r \in \mathcal{R}_{k,l}$ appear on the right hand side.

The quantities $A_{k,l}, P_{k,l}, \mathbf{b}_{k,l}, d_{k,l}$ and $\mathbf{f}_{k,l}$ correspond to the same matrices and vectors as for the global Lagrange system (3.14). Furthermore we realize that proper rearrangement gets an equivalent expression of the global system (3.14) of the form:

$$\begin{pmatrix} C_{0,0} & * & \cdots & * \\ * & C_{4,0} & \cdots & * \\ \vdots & \ddots & & \vdots \\ * & \cdots & & C_{\frac{n+1}{4}-1,\frac{m+1}{4}-1} \end{pmatrix} \mathbf{x} = \mathbf{r}. \tag{3.16}$$

The advantage of this rearrangement is that the local optimization problem $OP_{k,l}^{loc}$ can easily be solved since the number of unknowns is relatively small and thus direct methods to solve such a local optimization problem can be used. Additionally, solving a local optimization problem is equivalent to one step of a block-Gauss-Seidel or block SOR method applied to system (3.16).

In order for the block Gauss-Seidel or block SOR method to converge to the solution of the linear system (3.16), we have to guarantee that

(i) the functional is positive definite on the space of all interpolating TP-B-spline surfaces,

(ii) each local optimization problem $OP_{k,l}^{loc}$ has a solution and that

(iii) there exists an interpolating TP-B-spline surface.

The existence of an interpolating TP-B-spline surface is guaranteed by Step 2. The solvability of $OP_{k,l}^{loc}$ results from the following fact: $I_{k,l} \supseteq (\mathcal{R}_{k,l} \cap \{\mathbf{u}_i : i = 1, \ldots, N\})$, i.e., the matrix $P_{k.l}$ representing the interpolation constraints for $OP_{k,l}^{loc}$ consists of a subset of the equations from matrix $L_{k,l}$ representing $IP_{k,l}^{loc}$ and has therefore full rank. Finally, by Thm. 2.3(b) the functional is positive definite (if more than two data points have to be interpolated).

4 Results

In this section we give some examples and present the results of comparing our algorithm with other known methods for solving the SDI problem.

In order to compare known methods to the algorithm described in this paper we use the following procedure, which is widely used in the area of SDI:

(i) choose a test function F_{test} and a set of data points in the domain of F_{test}.

(ii) sample F_{test} at the data points and compute the interpolating function to this data.

(iii) compare the interpolating function which the test function on a finr regular grid: usually the discrete l_1-, l_2- or l_∞-norm is used to evaluate the approximation error of the interpolating function.

Since our method is mostly designed to construct interpolants with well-distributed curvature, we additionally use color-coded curvature plots to demonstrate the quality of the interpolant constructed with our algorithm. These curvature plots display mean curvature. Areas of high positive curvature are blue whereas red indicates regions of high negative curvature. The color shifts from red to yellow, green and cyan corresponding to curvature changes from positive to negative.

As test functions we use the functions presented by Franke and Ritchie (see [7])

$$F_{test}(u,v) = \tfrac{1}{2}\exp\left(-\tfrac{(9u-2)^2+(9v-2)^2}{4}\right) + \tfrac{3}{4}\exp\left(-\tfrac{(9u+1)^2}{49} + \tfrac{9v+1}{10}\right)$$
$$+ \tfrac{1}{2}\exp\left(-\tfrac{(9u-7)^2+(9v-3)^2}{4}\right) + \tfrac{1}{5}\exp\left(-(9u-4)^2 - (9v-7)^2\right),$$

and Ritchie (see [16])

$$F_{test}(u,v) = \begin{cases} 1 & \text{if } v - 2.1u + 0.1 \geq \frac{1}{2} \\ 2(v - 2.1u + 0.1) & \text{if } 0 \leq v - 2.1u + 0.1 < \frac{1}{2} \\ \frac{\cos(4\pi r(u,v))+1}{2} & \text{if } r(u,v) \leq \frac{1}{4} \\ 0 & \text{otherwise,} \end{cases}$$
$$\text{where } r(u,v) = \sqrt{(2.1u-1.6)^2 + (v-0.5)^2}.$$

Two sets of data points defined over the unit-square $[0,1]^2$ consisting of 100 and 200 data points are used.

We compare our algorithm with the RBF method based on Duchon's Thin-Plate-Splines (TPS) [5] and with the Oslo Mask-Method (OMM) presented by Arge-Dæhlen-Tveito [1].

Table 4.1 shows the number of 4×4 blocks of control points used to construct the interpolating TP B-spline surface and the number of control points for the reference surface.

	4×4 blocks of control points	control points for reference surface
100 data points	10/10	7/7
200 data points	12/12	11/11

Table 4.1: Number of 4×4 blocks of control points for interpolating TP B-spline surface and number of control points for reference surface.

In Tables 4.2–4.5 we present the different results obtained by the TPS-, the OMM-method and our algorithm for the four interpolation setups. The discrete l_1-, l_2- and l_∞-norms are obtained using a regular 75×75 grid over the bounding-box of the data points. Additionally to the approximation quality we give the computing time for the interpolant.

In case of Franke's function, which is curvature continuous having regions with strongly varying curvature, we find that our method has better approximation properties compared to the TPS- and the OMM-method. Ritchie's function, on the other hand, is not curvature continuous. The shaded image of Ritchie's test function are shown in Figure 4. For this function we find, that our algorithm based on 200 data points constructs an interpolating TP

B-spline surface having slightly higher approximation error than both, the TPS method (see Table 4.5). But observing Figures 4.3 and 4.4 one can clearly see that the interpolant based on the data dependent functional exhibits much smoother distribution of curvature.

Comparing the run-time behavior we find that the data dependent optimization needs more computing time than the data independent optimization and the TPS- and OMM-method[9]. This is not a surprising result since for the data dependent functional the value of the related inner product of the basis functions $\langle B_{i,j}|B_{k,l}\rangle$ depends on the reference surface and must be determined using numerical integration. However, depending on the specific application, the better quality of the resulting surface will justify the higher costs.

	TPS	OMM	simple TPE	data depend. TPE
l_∞	0.17186	0.173079	0.153844	0.116442
l_1	0.0101117	0.0184403	0.010426	0.00715282
l_2	0.0223111	0.0349568	0.0217872	0.015922
time	21.47	115.26	21.34	226.17

Table 4.2: Interpolation of 100 data values sampled from Franke's test function.

	TPS	OMM	simple TPE	data depend. TPE
l_∞	0.030172	0.131664	0.0313066	0.0325615
l_1	0.00236577	0.00583201	0.00249948	0.00202109
l_2	0.00464141	0.0143146	0.00499386	0.00484371
time	192.47	101.26	27.34	341.17

Table 4.3: Interpolation of 200 data values sampled from Franke's test function.

	TPS	OMM	simple TPE	data depend. TPE
l_∞	0.464249	0.496428	0.474206	0.509238
l_1	0.0277653	0.0327752	0.0293525	0.0248366
l_2	0.059029	0.0673014	0.0609666	0.0636782
time	21.67	108.96	18.78	262.62

Table 4.4: Interpolation of 100 data values sampled from Ritchie's test function.

	TPS	OMM	simple TPE	data depend. TPE
l_∞	0.226727	0.27666	0.215077	0.324951
l_1	0.0136	0.0174471	0.0138894	0.0120837
l_2	0.0289624	0.038039	0.0287011	0.0306399
time	192.67	80.96	22.78	403.62

Table 4.5: Interpolation of 200 data values sampled from Ritchie's test function.

[9]For the OMM scheme the computation time decreases slightly with the number of data points since each data point determines one function values of the grid function.

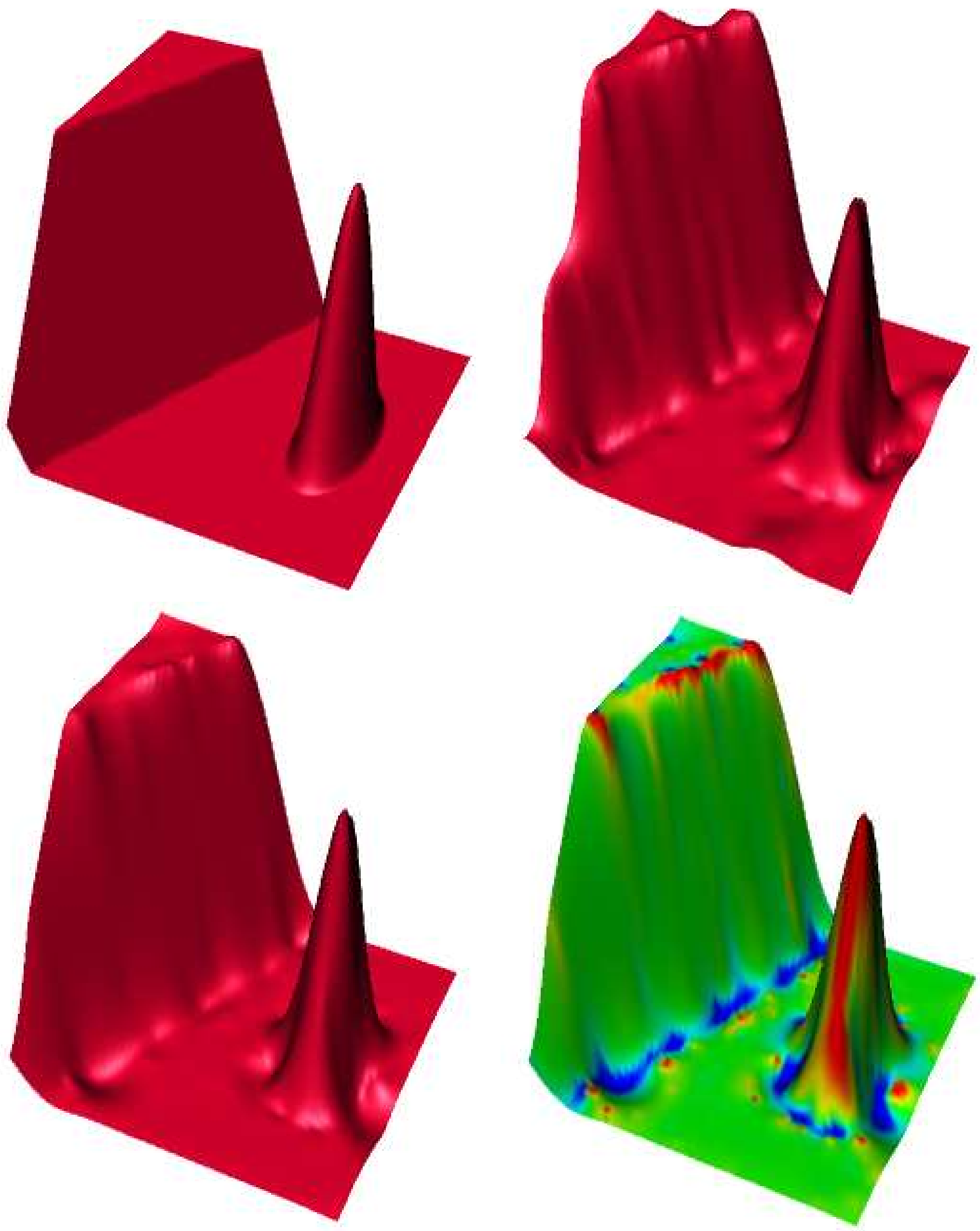

Figure 4.3: Shaded images (upper left) of Ritchie's test functions, the Oslo Mask Method (upper right) and the RBF-function based on Duchon's Thin-Plate-Splines as shaded image and curvature plot (lower left and lower right resp.)

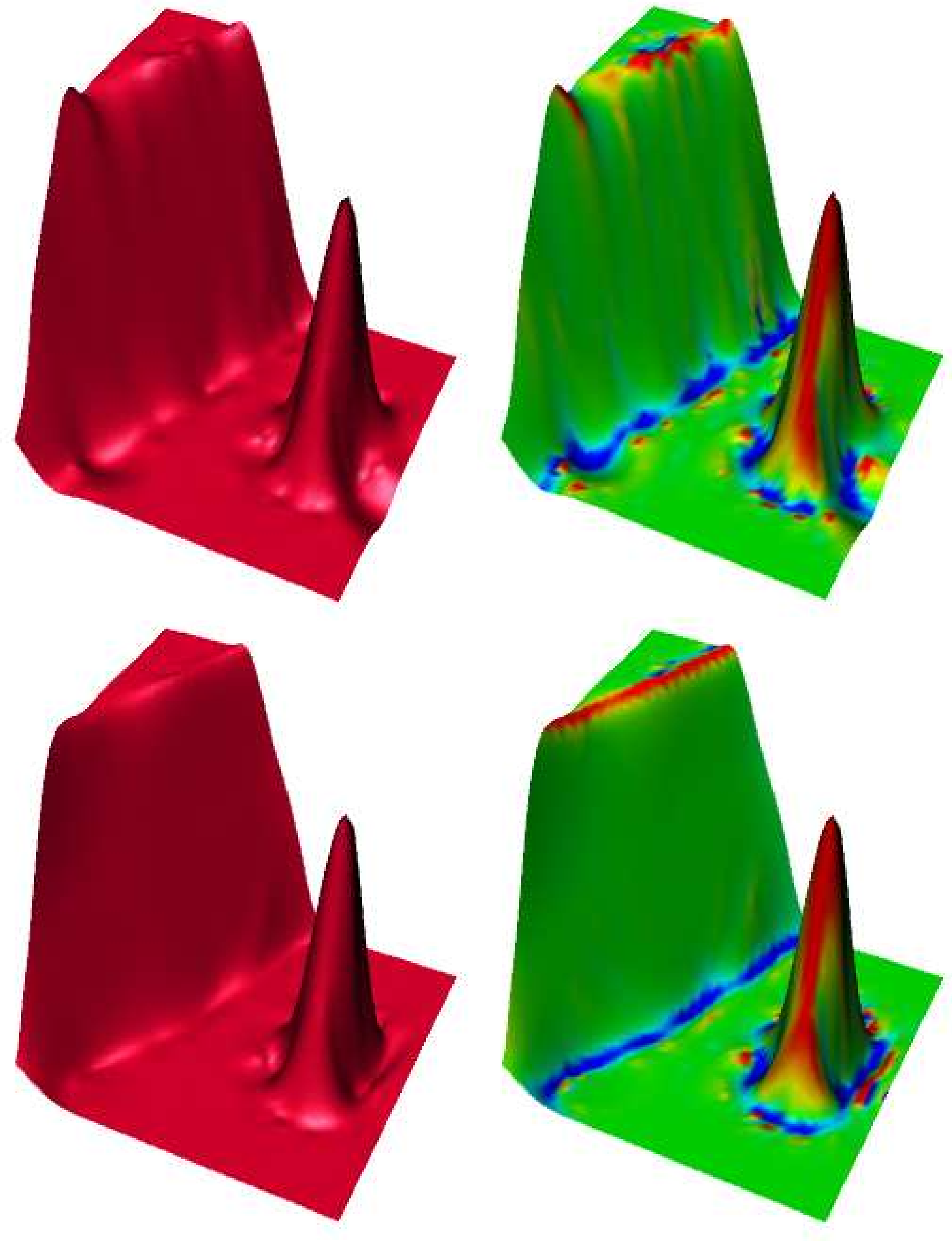

Figure 4.4: The interpolants based on functional optimization: Shaded images (left) and color curvature plots (right) for the simple TPE (top) and the data dependent TPE functional (bottom).

5 Conclusions and Future Work

In this paper we first gave on overview on the different functionals that are used in the area of surface optimization. We developed the data dependent TPE functional with is a good approximation of the exact TPE functional. Based on this functional we developed a new approach to solve the problem of interpolating scattered data. This new method generates interpolating TP B-spline functions. The degrees-of-freedom (DOF) not set by the interpolation constraints are determined using an optimization technique based upon (data dependent) fairness functionals. The main advantages of this method are: the resulting surface is given in a CAD compatible format and the interpolant exhibits very uniform distribution of curvature. Moreover, there is no limitation to the number of data points that can be interpolated by this approach.

At this moment, our algorithm constructs equidistant knot vectors. In cases where the data points are clustered, this may lead to a large number of DOFs for relatively small data points sets. The use of hierarchical B-splines (see [6]) should be investigated to handle this problem. As already pointed out, our algorithm can be extended to the so-called *parametric case*, i.e., to the interpolation of 3D point clouds. The main difference is that in the general situation no 2D data points (parameter values) for the 3D data points are known. This problem of assigning appropriate parameter values to the 3D data points has to be solved first. Once this is done, one can construct interpolating surfaces in nearly the same way as described above. Details will be given in a forthcoming paper.

References

[1] ARGE, E., DÆHLEN, M., AND TVEITO, A. Approximation of scattered data using smooth grid functions. Tech. rep., SINTEF-SI, Oslo, 1994.

[2] BRUNNETT, G., HAGEN, H., AND SANTARELLI, P. Variational design of curves and surfaces. *Surveys Math. Industry 3* (1993), 1–27.

[3] CELNIKER, G., AND GOSSARD, D. Deformable curve and surface finite-elements for free-form shape design. In *Proc. SIGGRAPH* (1991), vol. 25:4, ACM, pp. 257–266.

[4] DE BOOR, C. *A practical guide to splines.* Springer, New York, 1978.

[5] DUCHON, J. Splines minimizing rotation-invariant seminorms in Sobolev spaces. In *Constructive Theory of Functions of Several Variables*, W. Schempp and K. Zeller, Eds., vol. 571 of *Lecture Notes in Mathematics.* Springer Verlag, Berlin, 1977, pp. 85–100.

[6] FORSEY, D. R., AND BARTELS, R. H. Hierarchical B–spline refinement. In *Computer Graphics (SIGGRAPH '88 Proceedings)* (1988), pp. 205–212.

[7] FRANKE, R. Scattered data interpolation: Test of some methods. *Math. Comp.38* (1982), 181–200.

[8] GREINER, G. Surface construction based upon variational principles. In *Curves and Surfaces II* (Boston, 1994), A. L. P.J. Laurent and L. Schumaker, Eds., AK Peters.

[9] GREINER, G. Variational design and fairing. In *Proc. EUROGRAPHICS '94* (1994), vol. 13, Eurographics, Blackwell Publishers.

[10] HOSCHEK, J., AND LASSER, D. *Fundamentals of computer aided geometric design.* AK Peters, Wellesley, Mass., 1993.

[11] KALLAY, M. Constrained optimization in surface design. In *Modeling in Computer Graphics* (Berlin, 1993), B. Falcidieno and T. L. Kunii, Eds., Springer–Verlag.

[12] KLINGENBERG, W. *A course in differential geometry.* Springer-Verlag, Berlin-Heidelberg, 1978.

[13] KOBAYASHI, S., AND NOMIZU, K. *Foundations of differential geometry*, vol. 1. Interscience Publ., New York, 1969.

[14] MORETON, H., AND SÉQUIN, C. Functional optimization for fair surface design. In *Proc. SIGGRAPH* (1992), vol. 26, pp. 167–176.

[15] NIELSON, G. A method for interpolating scattered data based upon a minimum norm network. *Math. Comp.40* (1983), 253–271.

[16] RITCHIE, S. Surface representation by finite elements. Master's thesis, University of Calgary, 1978.

[17] SCHUMAKER, L. L., AND FASSHAUER, G. E. Minimal energy surfaces using parametric splines. *Comp. Aided Geom. Design 13* (1996), 45–79.

[18] STOER, J., AND BURLISCH, R. *Introduction to numerical analysis.* Springer, Heidelberg, 1980.

[19] WELCH, W., AND WITKIN, A. Variational surface modeling. *ACM Computer Graphics 26* (1992), 157–166.