ACM Symposium on Solid Modeling and Applications (2004)
G. Elber, N. Patrikalakis, P. Brunet (Editors)

Spline Approximation of General Volumetric Data

Christian Rössl[†] Frank Zeilfelder[‡] Günther Nürnberger[‡] Hans-Peter Seidel[†]

[†] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[‡] Universität Mannheim, Institut für Mathematik, Mannheim, Germany

Abstract
We present an efficient algorithm for approximating huge general volumetric data sets, i.e. the data is given over arbitrarily shaped volumes and consists of up to millions of samples. The method is based on cubic trivariate splines, i.e. piecewise polynomials of total degree three defined w.r.t. uniform type-6 tetrahedral partitions of the volumetric domain. Similar as in the recent bivariate approximation approaches (cf. [10, 15]), the splines in three variables are automatically determined from the discrete data as a result of a two-step method (see [40]), where local discrete least squares polynomial approximations of varying degrees are extended by using natural conditions, i.e. the continuity and smoothness properties which determine the underlying spline space. The main advantages of this approach with linear algorithmic complexity are as follows: no tetrahedral partition of the volume data is needed, only small linear systems have to be solved, the local variation and distribution of the data is automatically adapted, Bernstein-Bézier techniques well-known in Computer Aided Geometric Design (CAGD) can be fully exploited, noisy data are automatically smoothed. Our numerical examples with huge data sets for synthetic data as well as some real-world data confirm the efficiency of the methods, show the high quality of the spline approximation, and illustrate that the rendered iso-surfaces inherit a visual smooth appearance from the volume approximating splines.

1. Introduction

General *data fitting* is an important problem in many scientific areas and applications. The general goal in this field is to efficiently compute suitable models which approximate given sets of discrete data of different type. This gets challenging for very large data sets with arbitrarily distributed data samples possibly contaminated with some noise resulting from measurement. A very well-known and important example in *computer graphics*, *approximation theory* and *numerical analysis* is the bivariate problem of surface approximation, i.e. fitting the height data at given points which are arbitrarily distributed over a planar domain. The literature shows that even in this case it is a complex task to find appropriate methods which satisfy (almost all) requirements of efficient and exact fitting for data of general type. In this paper we go a step further and consider fitting of *general volumetric data*, i.e. we assume that sets of discrete points are arbitrarily distributed in a volume domain and some associated (scalar) density values at the points are given, and we are interested in finding a suitable non-discrete approximating model of the data, which allows a convenient further processing.

It is obvious that the most important property of any fitting method should be that it approximates well, i.e. the values evaluated from the model should be close to the data values at the given points. Besides this main point of good approximation quality there are a number of additional requirements which should be ideally satisfied by an approximation method. In brief, some of these requirements are: efficient computation, evaluation and representation of the models, applicability to reasonable distributions of general data, the models should satisfy smoothness conditions for high quality visualization, the models should have the potential to automatically reduce noise in contaminated data and for automatic data reduction. Depending on the specific applications this list of desirable properties may even be increased.

Due to the importance of data fitting in the different fields of application there exists a vast literature on this topic. We list some of the methods although we are aware that this list is far from being totally complete. An approach is to use radial basis functions and related hybrid and Shepard-like methods (see e.g. [4, 14, 22, 39]). Recently, such methods have been tuned towards surface reconstruction from volumetric data (see e.g. [6, 11, 33, 44]), which has typical applications in computer graphics and reverse engineering. Other methods are based on different types of splines. We mention local and global methods based on tensor product splines in three variables and the simplex spline approach [35, 36]. If the data is structured (for instance, as a result from some local nearest neighbor estimation, quantization type or gridding algorithm), then the usage of tensor product splines and related methods is often straightforward (see e.g. [13, 24, 25, 34] and the references therein). An example from the area of volume visualization are trilinear splines which interpolate the data on a three-dimensional grid - these are local spline models where the polynomial pieces are of total degree three. In this paper, we also use local spline models based on piecewise polynomials of total degree three - but these are different from tensor product splines (as well as from simplex splines, in general). The cubic splines we propose here are defined w.r.t. tetrahedral partitions of the three-dimensional domain and satisfy smoothness conditions. Due to their mathematical complexity currently there are only a few papers on trivariate splines and many open questions concerning these spaces exist to date (see [1, 7, 16, 20, 41, 42, 45] and the references therein). Nevertheless, we show here that the trivariate splines provide the necessary potential to be useful tools for solving volume fitting problems. For further information on the topic of data fitting we refer to [2, 21, 40, 46] and the references therein.

As noted above, the fitting problem becomes simpler if one considers *gridded* volume data. Recently, we developed *quasi-interpolation methods* for data of this structured type [28, 38] based on quadratic trivariate splines. According to our knowledge, these are the first approaches in the literature where it is shown that splines on tetrahedral partitions with lowest possible degree satisfying many smoothness conditions provide useful tools with advantageous properties for the various requirements of *efficient visualization* in computer graphics. As announced in the future work section of [38], in this paper we continue the agenda and focus on the more difficult problem of efficient approximation of general volumetric data. For developing the method presented here, we approach the problem from two sides. On the one side we benefit from our experience on trivariate splines described in [28, 38]. For instance, the computational examples presented in these papers and other tests showed that the basic algorithms frequently used in the practice of bivariate splines (see [10, 31, 46]) can also be efficiently applied for splines on tetrahedral partitions. This concerns e.g. the *Bernstein-Bézier techniques* well-known in *CAGD* (see e.g. [18, 37]) and the computation of local approximants. On the other hand, it has only been recently that algorithms for the efficient interpolation and approximation of general bivariate data sets appearing in certain real-world settings have been developed which take many of the above requirements into account. These methods (see [10, 15, 19, 26, 27, 30, 32], and the survey article [31] as well as the references therein) are based on *bivariate splines*, i.e. piecewise polynomials satisfying smoothness conditions which are defined w.r.t. planar and three-dimensional triangulations. In fact, the method presented here is the first generalization of the recent bivariate fitting methods [10, 15] to the more complex trivariate setting and therefore falls into the class of *spline extension methods*. Roughly speaking, this *two-step approach* (see [40]) works as follows. In a first step, we independently compute trivariate polynomial approximations to appropriate local portions of the data directly in its Bernstein-Bézier form. This can be done by imposing a *checkerboard-coloring* (see [29]) to the uniform type tetrahedral partition associated with the splines. Following the ideas in [10, 15], we adapt the degree of the local polynomials to the local variation and distribution of the data as well as for the type of data. This makes this step stable and robust and provides some smoothing of the noisy data if this is necessary. In contrast to earlier methods known from the literature based on trivariate piecewise polynomials, we directly use these local polynomial approximants *as pieces* of the trivariate splines. In the second step, these local pieces are glued together using the continuity and smoothness conditions which define the underlying spline space. In this way, the splines are defined on the whole volumetric domain as a result of building extensions of the local representants of the data.

The complexity of this general algorithm is linear in the number of (reasonably distributed) data points. In brief, its main advantages are as follows. No computation or storage for a tetrahedral partition of (a subset) of the data points is needed. Only small linear systems have to be solved - this can be done independently and in parallel, and therefore enables the handling of huge data sets (i.e. the number of data points is of order $\mathcal{O}(10^6)$). The computation, evaluation and representation of the approximating splines is efficient due to the exploitation of the Bernstein-Bézier techniques. The algorithm possess an insensitiveness concerning data contaminated with moderate noise. Moreover, only basic operations and tools available in standard numerical libraries are applied. These facts ensure not only the efficiency of the method but also the simplicity of its implementation.

2. Overview of the algorithm

We briefly sketch the basic idea of the algorithm. The method is based on cubic splines w.r.t. a uniform type tetrahedral partition, i.e. piecewise polynomials of total degree three which satisfy continuity and smoothness conditions across the common triangular faces of neighboring tetrahedra. The tetrahedral partitions are natural generalizations of the four-direction meshes well-known from the bivariate setting and cover the volume domain containing the given data points. Basically, the method consists of two steps. In the first step, we use a checkerboard coloring and choose a subset of tetrahedra for which we compute local least squares polynomial approximations of varying degrees for small portions of the data close to the respective tetrahedron. On the set of these (pairwise distinct) tetrahedra, we define the spline to be equal to the local polynomial approximations. In the second step, we use the conditions of the underlying spline space to uniquely extend the polynomial pieces obtained in the first step to a consistent spline on the whole domain. Hence, the algorithm completely follows the basic ideas known from approximation method of the bivariate setting ([10, 15]). What is new here, is that we generalize these methods to the more complex trivariate setting by using the partition described in [38]. Comparing with [10, 15], we observe that in the trivariate setting the second step (extension to a consistent spline) becomes more complex, while the first step (local polynomial approximation) essentially coincides. The consequences are that we keep the description of the first step very short and informative, while in the second step we also skip a few of the smoothness conditions and replace others by some different natural conditions in our first method dealing with general volumetric data. One motivation for doing this comes from the observations described in [28, 38]. Below we show that the whole approach applies only basic computations and averaging operations and therefore the algorithm is simple and straight forward to implement. Moreover, we note that we use cubics in this paper, because according to our experience these spaces provide at least some of the additional flexibility (comparing with quadratics as they were used in [38]) needed for the efficient approximation of arbitrarily distributed, three-dimensional data.

3. Trivariate Splines

Before going into the details of our algorithm, we give some background information on spaces of cubic splines w.r.t. uniform type tetrahedral partitions Δ, include a brief review of the piecewise Bernstein-Bézier form of trivariate splines and discuss smoothness conditions on adjacent tetrahedra of Δ.

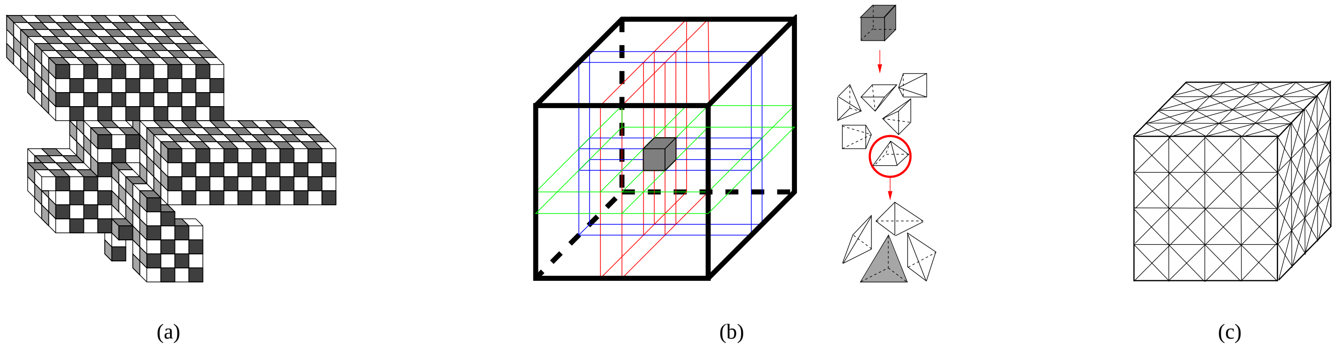

Figure 1: *(a) Example of a more general domain Ω than the unit cube. The domain is decomposed into uniform cubes which are colored* black *and* white. *(b) The tetrahedral partition Δ is obtained by uniformly subdividing each cube of ◊ into 24 tetrahedra. Since six planes are needed, Δ is sometimes called a* type-6 tetrahedral partition. *In the black cubes one tetrahedron is consistently colored black. (c) The intersections of Δ with planes parallel to the three coordinate planes are four-directional meshes which are well-known from the bivariate setting.*

3.1. Type-6 Tetrahedral Partitions Δ

The approximation method is based on cubic splines, i.e. piecewise polynomials of total degree three which are defined on natural uniform tetrahedral partitions Δ. The partitions are the same as those used for reconstruction of gridded data in [38]. For ease of explanation we choose a cubic domain $\Omega = [0,n]^3 \subseteq \mathbb{R}^3$ for now, although more general domains are possible, which are decomposed into cubes (cf. Fig. 1 (a)). Given a set of discrete *volumetric data points* $\mathcal{X} = \{\mathbf{x} = (x_\nu, y_\nu, z_\nu) \in \Omega : \ \nu = 1,\ldots,N\} \subseteq \mathbb{R}^3$ with associated *functional (scalar) values* $f_\mathbf{x} \in \mathbb{R}$, $\mathbf{x} \in \mathcal{X}$, we set $n = \lfloor \sqrt[3]{N/10} \rfloor$ and cover the domain Ω with cubes $Q_{i,j,k}$, $i,j,k = 1,\ldots,n$, of edge length $h = 1/n$. This choice of n ensures that the number of degrees of freedom of the spline space approximately coincides with the number of scattered data points (see Sec. 3.3). On the other hand, this is just a reasonable heuristic choice which performed well for many of the computational examples presented in Sec. 5. We also note that for automatic data reduction different choices of n might be advantageous. In addition, we need a ring of *border cells* surrounding the union ◊ of the cubes $Q_{i,j,k}$ to completely determine the approximating spline on Ω.

In order to define a tetrahedral partition (see [5] for a survey), we split each cube $Q \in ◊$ into six pyramids by connecting its center point v_Q with the four vertices of every face of Q. Then, we insert both diagonals in these six faces of Q and connect their intersection points with v_Q. This subdivides each of the six pyramids in Q into four tetrahedra, forming a natural, uniform tetrahedral partition Δ of Ω, where every cube $Q \in ◊$ contains 24 congruent tetrahedra. Another way to describe the *type-6 tetrahedral partition* Δ is to say that Δ is obtained by slicing Ω with the *six planes* which contain opposite edges of Ω. Fig. 1 (b) illustrates the construction of Δ. The partition Δ is a generalization of the four-directional mesh which is well-known in the bivariate setting (cf. [7, 10, 15]), Fig. 1 (c)) shows this. We will construct a piecewise cubic spline w.r.t. Δ with polynomial pieces defined over every tetrahedron.

We impose a checkerboard coloring (a concept introduced in the context of local Lagrange interpolation by bivariate splines in [29]) to the cubes from ◊: Cubes $Q_{i,j,k}$ where $i+j+k$ is even are called *black cubes*, while the rest of the cubes are called *white cubes*. For every black cube we choose the same tetrahedron, e.g. always the front facing one in the bottom pyramid (cf. Fig. 1 (b) and 9), and call these in the remainder of this paper the *black tetrahedra* of Δ.

3.2. Bernstein-Bézier Form and Smoothness Conditions

The *Bernstein-Bézier techniques* are well established tools from CAGD [18, 37], and it is well known that the related *Bernstein-Bézier form* of the polynomial pieces plays a key role for *multivariate splines* [1, 7, 16, 41, 45, 46]. Every cubic spline s on Δ can be written in its piecewise Bernstein-Bézier form, i.e. for every tetrahedron $T = [v_0, v_1, v_2, v_3] \in \Delta$ with vertices v_μ, $\mu = 0,\ldots,3$, we have

$$s|_T = p = \sum_{i+j+k+\ell=3} a_{ijk\ell}\, B_{ijk\ell} \in \mathcal{P}_3, \qquad (1)$$

where $a_{ijk\ell} \in \mathbb{R}$ are called the *Bernstein-Bézier coefficients* of the polynomial piece p associated with the *Bézier points*

$$(i\,v_0 + j\,v_1 + k\,v_2 + \ell\,v_3)/3, \qquad i+j+k+\ell = 3,$$

which are sometimes also called *domain points*. Here,

$$\mathcal{P}_3 = \ \text{span}\,\{x^i y^j z^k : \ i,\ j,\ k \geq 0,\ i+j+k \leq 3\}$$

denotes the twenty-dimensional space of *cubic trivariate polynomials*, i.e. the total degree is three. Moreover,

$$B_{ijk\ell} = \tfrac{3!}{i!j!k!\ell!}\, \lambda_0^i \lambda_1^j \lambda_2^k \lambda_3^\ell \in \mathcal{P}_3, \qquad i+j+k+\ell = 3,$$

are the *cubic Bernstein basis polynomials* w.r.t. T, where the linear polynomials λ_ν, $\nu = 0,\ldots,3$, determined by the interpolation conditions $\lambda_\nu(v_\mu) = \delta_{\nu,\mu}$, $\mu = 0,\ldots,3$, are called the *barycentric coordinates* with respect to T.

A convenient description of C^1-smoothness for *neighboring polynomials* (i.e. polynomials defined on tetrahedra sharing a triangular face) in Bernstein-Bézier form can be found in [3, 7, 12]. Let $p = s|_T$ be given on T as in (1), and set $\tilde{p} = s|_{\tilde{T}} \in \mathcal{P}_3$ for a neighboring polynomial on $\tilde{T} = [v_0, v_1, v_2, \tilde{v}_3]$ with Bernstein-Bézier coefficients $\tilde{a}_{ijk\ell}$, $i+j+k+\ell = 3$. Then s is a continuous spline on $T \cup \tilde{T}$, if the coefficients associated with Bèzier points on the common triangular face $T \cap \tilde{T} = [v_0, v_1, v_2]$ coincide, i.e. $a_{ijk0} = \tilde{a}_{ijk0}$, $i+j+k=3$. Moreover, s is C^1-smooth across $T \cap \tilde{T}$ iff, in addition, we have

$$\begin{aligned}\tilde{a}_{ijk1} \;=\; & \lambda_0(\tilde{v}_3)\, a_{i+1jk0} + \lambda_1(\tilde{v}_3)\, a_{ij+1k0} + \\ & \lambda_2(\tilde{v}_3)\, a_{ijk+10} + \lambda_3(\tilde{v}_3)\, a_{ijk1}, \qquad i+j+k=2.\end{aligned}$$

Analogously to the univariate and bivariate cases, for each condition of this form there is the geometric interpretation that five points in $\mathbb{R}^4$ lie in the same (three-dimensional) hyperplane, in general. The fourth components of these points are the Bernstein-Bézier coefficients while the first three components are the associated Bézier points. In general, there are five coefficients involved for every single smoothness condition. If one or even two of the barycentric coordinates vanish at $\tilde{v}_3$, the number of involved coefficients is four and three, respectively. For instance, this holds if $\tilde{v}_3$ lies in the plane that contains the triangle $[v_0, v_1, v_3]$ and if $\tilde{v}_3$ lies on the line that contains the edge $[v_0, v_3]$, respectively. In these cases, the smoothness conditions *degenerate* to lower dimensional conditions known from the bivariate and univariate setting, respectively. Fig. 2 illustrates a degenerated and the general case.

For the type-6 tetrahedral partition Δ defined in the previous subsection, there are three cases, i.e. we have to consider neighboring tetrahedra lying in

- two different cubes of $\diamondsuit$,
- the same pyramid (cf. Fig. 1) of a cube,
- two different pyramids of a cube.

We observe that the smoothness conditions for the first two cases (inter-cube and intra-pyramid) degenerate to simple univariate conditions involving three coefficients (Fig. 2, left), while in the third case (inter-pyramid) the smoothness conditions are of the general form involving five coefficients (Fig. 2, right). Due to symmetry, there are essentially two conditions with always the same barycentric coordinates involved, and one can easily see that enforcing smoothness over the common triangular face of neighboring tetrahedra in Δ is to apply one of these two simple averaging formulae (Fig. 2 shows the two stencils).

In the algorithm described in Section 4, we take advantage of the Bernstein-Bézier form not only to express and use certain smoothness conditions. In fact, it enables us to apply many standard techniques from CAGD. Most important is the efficient and robust evaluation using the *de Casteljau algorithm* (see e.g. [18, 37]). It computes not only the value of the polynomial pieces $s|_T = p$ but simultaneously also its piecewise derivatives which is essential for efficient and high-quality visualization purposes. These aspects are discussed in [38] together with implementation issues like the efficient computation of point locations and barycentric coordinates in the three dimensional domain. Although we make extensive use of the Bernstein-Bézier techniques in our current implementation of the algorithm there is obviously no need to repeat these computational aspects again, and we refer the interested reader to [38].

3.3. Cubic Splines on Δ

As one can see from the previous section, the C^1-smoothness for the polynomial pieces of the splines on *two adjacent* tetrahedra of the type-6 tetrahedral partition Δ are relatively simple to describe by using the piecewise Bernstein-Bézier form. On the other hand, if we consider the *complete partition* Δ, these conditions connect the coefficients of an overall C^1-smooth spline in a highly non-trivial way, because for each (interior) tetrahedron T of Δ the conditions have to be satisfied simultaneously across all the four triangular faces of T - and they can obviously not be considered independently. This observation is contrasted to the situation of splines in one variable, in the sense that for smooth multivariate spline spaces of low (and lowest possible) polynomial degree, one can sometimes observe that the splines have to simultaneously satisfy a huge number of smoothness conditions, while on the other hand the number of coefficients involved is relatively low. As a basic work, the analysis of the complicated structure for C^1-splines on Δ of arbitrary polynomial degree has been provided recently [16] (see also [20, 42]). From these results we know that in the particular case of C^1 cubics on Δ the number of degrees of freedom of the spline spaces fits into the formula $6\,n^3 + 24\,n^2 + 18\,n + 4$, which shows that the spaces allow to deal with trivariate data, in principle. On the other hand, some basic observations motivated by the bivariate approximation methods in [10, 15] (see also [42], Remark 7.3) seem to indicate that this number is too small for designing a local approximation method with optimal properties using the overall C^1-smooth cubic splines. As noted above, this makes the extension step of the below algorithm more complex and different to the bivariate case. More precisely, in our first approach for local extension described below (see Sec. 4.2) we use cubic splines on Δ with about $10\,n^3 + \mathcal{O}(n^2)$ free parameters (i.e. the 20 coefficients associated with the complete set of domain points in each of the $n^3/2$ black tetrahedra are chosen), where most of the C^1-smoothness properties (but *not all*) are satisfied and *only few* of them are skipped or replaced by other useful conditions, so that the local approximation of the data is preserved. An additional motivation for proceeding this way comes from the results of our previous work [38] on piecewise quadratic reconstructions from gridded volume data. We note, that it can be easily seen from the description of the extension step in Sec. 4.2 in conjunction with the specific form of the stencils of the averaging rules representing smoothness conditions (cf. Fig. 2) that the resulting splines satisfy a huge number of smoothness conditions including those essential for certain visualization purposes and therefore have an almost similar behavior as mathematically smooth functions. In addition, this is confirmed by the computational examples given in the results section.

4. Approximation Method

We use the basic ideas from the two-step methods [10, 15] and adapt them to the trivariate setting. With the uniform tetrahedral partition Δ defined, in the first step, we determine least squares polynomial approximations for small, local portions $\mathcal{X}_{loc}$ of the given data from $\mathcal{X}$. Computing the trivariate local polynomials p_{loc} with $p_{loc}(\mathbf{x}) \approx f_{\mathbf{x}}$, $\mathbf{x} \in \mathcal{X}_{loc}$, can be done by using the same basic principles as described in [10, 15] (see also [8]) and applying them in a straight forward way to the trivariate setting. In contrast, finding methods for extending the local trivariate polynomial pieces to a consistent cubic spline (second step) are more difficult and different to the two-dimensional case. In the approach described below, we mainly concentrate on the important aspects of computational efficiency.

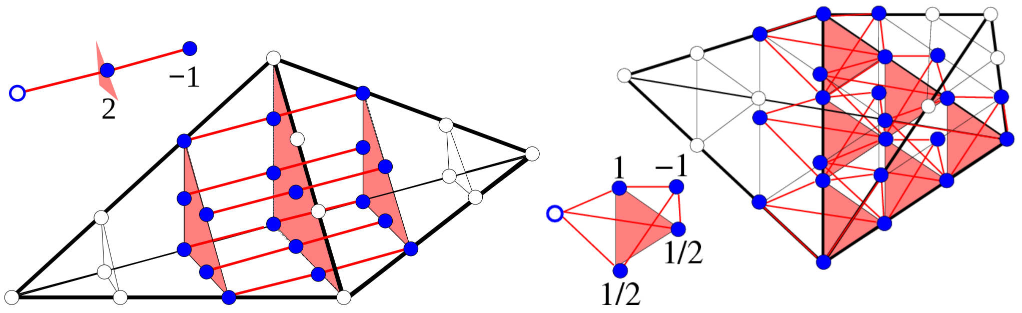

Figure 2: *Smoothness conditions of cubic polynomial pieces on neighboring tetrahedra of* Δ. *The Bézier points associated with coefficients involved in the smoothness conditions are shown as blue dots. For conditions between neighboring tetrahedra either of different cubes or of the same pyramid, three points lie on a line, i.e. two barycentric coordinates vanish, and the conditions degenerate to* smoothness conditions of univariate type *(left). Only for neighboring tetrahedra of different pyramids inside the same cube we have the* general case *with all the five coefficients involved in each of the six conditions (right). The closeups show* stencils *of the two averaging rules representing the conditions. In our algorithm the outlined coefficient is determined from the solid ones by applying the respective smoothness condition.*

4.1. Local Polynomial Approximation

In the first step, we determine least squares polynomial approximations for a huge number of small, local portions of the given data points $\mathcal{X}$ and the associated values $f_{\mathbf{x}}$, $\mathbf{x} \in \mathcal{X}$. To do this, only the black tetrahedra (cf. Sec. 3.1) in Δ are considered. More precisely, for each such tetrahedron T, we choose an appropriate subset $\mathcal{X}_{loc} = \mathcal{X}_{loc}^T$ of $\mathcal{X}$ containing data points which are close to T, and compute $p_{loc} = p_{loc}^T$, trivariate polynomials of degree $d \in \{0, \ldots, 3\}$ in its Bernstein-Bézier form w.r.t. T, such that the error

$$\sum_{\mathbf{x} \in \mathcal{X}_{loc}^T} (p_{loc}^T(\mathbf{x}) - f_{\mathbf{x}})^2$$

becomes minimal. To do this algorithmically, for each of the black tetrahedra $T \in \Delta$ we choose an initial sphere centered at the barycenter of T such that the volumetric domain is completely covered by the union of these spheres. Then, we collect the data points within each such sphere. The finding of these points can be done efficiently by initially sorting the data points from $\mathcal{X}$ into an appropriate uniform grid data structure. Following [10, 15], we analogously balance the number of data points distributed within a particular sphere depending on the local distribution of data points. This is done either by *thinning* or *increasing the radius* of the spheres. In this way, for each black tetrahedron T, we obtain a local portion $\mathcal{X}_{loc}^T$ of the data which is contained in an appropriate sphere $\mathcal{S}_{loc}^T$. Then, we determine the local polynomial p_{loc}^T on T which approximates the data values $f_{\mathbf{x}}$ at the points $\mathbf{x} \in \mathcal{X}_{loc}^T$ in the above *discrete least squares sense*. We solve the arising system of linear equations by computing the *singular value decomposition (SVD)*. Since the corresponding observation matrices are of moderate size (the polynomial degree p_{loc}^T and the cardinality of $\mathcal{X}_{loc}^T$ are both small) this can be done in a fast and robust way. In addition, as is well-known the SVD allows to check if this system is well-conditioned or not (in our current implementation, we follow [15] at this point, although we are aware that this can be improved). If the latter case appears (i.e. there is some hidden redundancy in $\mathcal{X}_{loc}^T$), we proceed by dropping the polynomial degree d and consider a new system for polynomials of degree $d-1$, and iterate this process until either the system is well-conditioned or the polynomial degree is zero. This procedure is initialized with $d = 3$. If the resulting polynomial is of lower degree we can rewrite it as a cubic polynomial by applying (successive) *degree raising* (see e.g. [18]). This local approximation procedure is analogous to the bivariate case [10, 15] (see also [8]), and provides numerical stability of the approximation part of the algorithm. Fig. 8 shows a visualization of a single local polynomial approximant.

4.2. Spline Extension

The approximation step described in the previous subsection determines the polynomial pieces of the approximating spline on the set of all black tetrahedra. More precisely, for each such tetrahedron T, we set the *approximating spline* to be *equal to* the local polynomial approximation p_{loc}^T, i.e. the 20 Bernstein-Bézier coefficients of the spline piece in (1) coincide with the 20 Bernstein-Bézier coefficients of p_{loc}^T. Now, in this second step, we show how to compute the Bernstein-Bézier coefficients of the approximating spline on the remaining tetrahedra of Δ which are *not black* and have a non-empty intersection with Ω. As noted above this can be understood as an extension of the local approximating pieces obtained in the first step, where we use the continuity and many smoothness conditions.

For ease of explanation, we proceed by considering only a black and a white cube of the partition $\Diamond$ (see Sect. 3.1). Figures 9 and 10 show the domain points associated with the coefficients of the polynomial pieces defined on tetrahedra in a black and a white cube, respectively. These two cubes represent all interior cubes of $\Diamond$. As coefficients of the outermost layer (called layer 3) coincide with those of neighboring cubes, we only show the inner layers of the white cube. We use these figures to explain how the remaining coefficients are determined. The coefficients are computed step by step and this is done locally, i.e. simultaneously for all the cubes in $\Diamond$. To understand the below description, it may help to simultaneously think of what happens in each step to the imaginary neighbors of the black and the white cube (which have a common edge or a common vertex with these cubes). We denote the coefficients by a_i, $i = 0, \ldots, 239$, their indices i are shown in the diagram. The indices i are chosen to represent the order in which the a_i are determined, i.e. for $i < j$ the coefficient a_i is computed before (or eventually simultaneously with) a_j, so the value of a_j may depend on the value of a_i. We decided to give the following description of the second step, because it follows the method we basically used to implement it – obviously a pure mathematical description could be done shorter – but would require additional notation.

Since the spline is already determined on the black tetrahedra, it follows that the coefficients $a_0,\ldots,a_{19}$ as well as $a_{20},\ldots,a_{25}$ are already uniquely determined. For the latter coefficients this can be seen by taking into account that there are other black cubes which share a common edge or a vertex with the black cube of consideration. These initial coefficients (resulting from the approximation procedure of the first step) are marked yellow. We will determine the main part of the remaining coefficients by applying the simple averaging rules connected with the smoothness conditions (cf. Fig. 2). The given sequential order implicitly defines the appropriate rule, structural ambiguities do not impose any over-determination since in this case we skip the smoothness conditions or replace by appropriate averages, which are non–standard rules. Only some few a_i will be treated in such a specific, unusual way – on the other hand this makes the whole construction possible. The light green coloring gives a hint on finding the coefficients $a_{26},\ldots,a_{147}$, which are computed before the first non-standard rule is applied.

Using intra-pyramid rules gives $a_{26},\ldots,a_{37}$, e.g. $a_{26}=2a_6-a_2$. An inter-pyramid conditions determines $a_{38}=a_{12}-a_6+\frac{1}{2}(a_3+a_{26})$, then $a_{39}=2a_{38}-a_{26}$, and $a_{40}=\frac{1}{2}(a_2+a_{39})$. An inter-cube condition gives $a_{41}=2a_{38}-a_{12}$ in the neighboring white cube. We continue this way and apply the smoothness conditions to obtain $a_{42},\ldots,...a_{92}$. Hence, we determine all coefficients around the cube vertices, i.e. the spline becomes smooth at all vertices of $\diamondsuit$. Note that as we “walk around” a vertex, we consider the already determined coefficients in the neighboring cubes, i.e. for the diagram we imagine a continuation of cubes in all directions. Using the smoothness conditions, we compute $a_{93},\ldots,a_{105}$ which completes the bottom sides of layers three and two (cf. Fig. 9) for all cubes. Then, we determine the top side of layer three, i.e. the coefficients $a_{106},\ldots,a_{114}$ and of layer two, i.e. $a_{115},\ldots,a_{136}$. After that, the inner layers one and zero of all black cubes can be computed, i.e. the coefficients $a_{137},\ldots,a_{147}$ and the spline becomes smooth at the midpoints of black cubes. We have now fixed all coefficients marked light green, where we used smoothness conditions from the C^1-spline spaces. At that point, one can see that there would be some hidden overdetermination for the overall C^1- splines (cf. [42], Remark 7.3) - therefore, we have to proceed differently.

Now, consider the unknown five coefficients $a_{148},\ldots,a_{152}$ on the layer two of the black cubes (marked by light red squares). Applying the C^1-conditions on the same layer and between layers two and one, we obtain the six equations

$$\begin{aligned}
a_{152} &= 2\,a_{149}-a_{33}\\
a_{152} &= 2\,a_{148}-a_{11}\\
a_{152} &= 2\,a_{151}-a_{116}\\
a_{152} &= 2\,a_{150}-a_{118}\\
a_{148}+a_{149} &= \tfrac{1}{2}\,(a_{12}+a_{152})+a_{17}\\
a_{150}+a_{151} &= \tfrac{1}{2}\,(a_{58}+a_{152})+a_{141}
\end{aligned}$$

for the five unknowns. As the system is over-determined, we suggest to *average smoothness conditions* as follows. Straightforward substitution provides

$$\begin{aligned}
a_{148} &= \tfrac{1}{2}\,(a_{12}-a_{33})+a_{17}\\
a_{149} &= \tfrac{1}{2}\,(a_{12}-a_{11})+a_{17}\\
a_{150} &= \tfrac{1}{2}\,(a_{58}-a_{116})+a_{141}\\
a_{151} &= \tfrac{1}{2}\,(a_{58}-a_{118})+a_{141}
\end{aligned}$$

and hence determines $a_{148},\ldots,a_{151}$. Back-substitution gives four conditions on a_{152} that are averaged to

$$a_{152} = \tfrac{1}{2}(a_{12}-a_{11}-a_{33}+a_{58}-a_{116}-a_{118})+a_{17}+a_{141}.$$

We apply the same averaging of coefficients obtained from smoothness conditions symmetrically to determine $a_{153},\ldots,a_{157}$ on the front left side of the black cubes (marked by red squares).

Next we compute the coefficients a_{158} and a_{159} in the black cubes (marked by light blue rhombs). Due to an intra-pyramid smoothness condition, we can use $a_{158}=2\,a_{159}-a_{135}$. In order to uniquely determine both coefficients, in addition, we impose the individual C^2-super-smoothness condition

$$4\,a_{159} = a_{39}+4\,a_{158}-a_{78}$$

which is a standard procedure to eliminate undesirable degrees of freedom for splines (see [10, 27, 41]). This is illustrated by using the dashed line in Fig. 9 showing the coefficients that are involved, here. Now, we uniquely determine $a_{160},\ldots,a_{167}$ using the smoothness conditions involving these coefficients around the vertical edge. Note that this is possible due to the careful choice of a_{152} and a_{159}. Analogously, we determine a_{168} and a_{169} (marked by blue rhombs) using a C^2-condition (illustrated as a dashed line), and walking around the corresponding edge uniquely determines the coefficients $a_{170},\ldots,a_{175}$ via smoothness conditions.

We now complete the outermost layer of the black cubes by applying intra-pyramid rules, and we obtain $a_{176},\ldots,a_{193}$. For the computation of the coefficients $a_{194},\ldots,a_{204}$ on layer two of the black cubes we consider intra-pyramid conditions only, and we *skip* seven inter-pyramid conditions. Note that this does not affect the smoothness across the common faces of black and white cubes. Finally, we determine the inner levels one and zero of the white cubes by using smoothness conditions, and we obtain $a_{205},\ldots,a_{224}$ and $a_{225},\ldots,a_{239}$, respectively.

Now, all the remaining coefficients of the spline (essential for the representation on Ω) are uniquely determined, and hence we have extended the local polynomial approximations from the first step to a spline defined on the whole domain Ω. The extension to the spline turns out to be a repeated averaging of coefficients using very simple and natural rules most of which representing smoothness conditions, making this step easy to implement and very efficient. Let us point out that the resulting consistent spline is C^1 between cubes sharing a common square face, as well as inside the pyramids (consisting of four tetrahedra sharing a common edge) and at the midpoints of all cubes. Moreover, many additional smoothness conditions are automatically satisfied by applying the above method. In the diagram of Fig. 9 and 10 the coefficients are indexed for illustration purposes, and we assume that all a_i (for fixed index i and variable cube index) are computed simultaneously for all the cubes. In the implementation this would require an iteration over all cubes and to compute each a_i individually. We can minimize the number of iterations to six by reordering the computation of coefficients (or appropriately permutating the 240 indices) while using the same rules to compute the coefficients.

5. Results

In this section we demonstrate the efficiency of the algorithm and the high quality of the spline approximations. In the following, all computation times are measured in seconds on a (single) 3 GHz Intel Xeon CPU using double precision arithmetic. Tests on an SGI Onyx using eight 400 MHz R12k processors concurrently show that we get nearly optimal speedup from parallelizing the algorithm. The iso-surfaces of the approximating splines are visualized as very fine triangular meshes generated by applying the *Marching Cubes algorithm* [23].

In order to investigate the quality of the approximation, we first consider the smooth trivariate test function of exponential type

$$\begin{aligned} f(x,y,z) &= \tfrac{1}{2}\, e^{-10((x-\frac{1}{4})^2+(y-\frac{1}{4})^2)} \\ &+ \tfrac{3}{4}\, e^{-16((x-\frac{1}{4})^2+(y-\frac{1}{4})^2+(z-\frac{1}{4})^2)} \\ &+ \tfrac{1}{2}\, e^{-10((x-\frac{3}{4})^2+(y-\frac{1}{8})^2+(z-\frac{1}{2})^2)} \\ &- \tfrac{1}{4}\, e^{-20((x-\frac{3}{4})^2+(y-\frac{3}{4})^2)}, \end{aligned}$$

for all $(x,y,z) \in \Omega_f = [-\frac{1}{2},\frac{1}{2}]^3$ (cf. [17]). Fig. 3 shows several iso-surfaces of the spline approximation s_f of f.

We sample f at N randomly distributed points $\mathbf{x}$ and approximate the values $f_{\mathbf{x}} = f(\mathbf{x})$ at these data points with s_f. Here, the number of cubes in every dimension is chosen as described in Sect. 3.1 so that the degrees of freedom of the spline approximately matches the number of data points. Tab. 1 shows different measurements of approximation errors and computation times for increasing numbers N of random samples and the respective choice of n. The third column shows the average error measured at the data points, the fourth columns lists the maximum error to the data, and the fifth column contains the maximal error to f in the uniform norm. The latter error is approximately computed by choosing 20 uniformly distributed points in each tetrahedron of Δ, evaluating the error for all these points, and computing the maximal error over all these approximative errors. The computed errors are obtained by considering the essential tetrahedra, i.e. tetrahedra contained in cubes from the complete interior of Ω_f. Note that passing from the i-th row to $(i+1)$-th row of the table (doubling N), the side length of the cubes decreases by the factor $2^{-(1/3)} \approx 0.79$. The last column shows the time for the local least squares approximation measured in seconds. Every row in the table is an average of 50 independent scattered tests, each of which uses a different random distribution of the data points. The time for determining the coefficients in the extension step (see Sect. 4.2) are not listed explicitly, since it is clearly linear in the number of cubes. In our computations, we observed that this is less than 5% of the time required for determining the local polynomial approximations (first step of algorithm). The test shows the quality of the spline approximation as well as the efficiency of its computation, particularly confirming the linear complexity of the algorithm. Moreover, we give a test that shows that our algorithm provides the potential to deal with noisy input data. This is illustrated by the results shown in Fig. 4. In this test, the domain is decomposed into 29^3 cubes to approximate 128 000 samples as before (cf. Tab. 1), and we added uniformly distributed noise to the sampled function values of f. Similar results were obtained for different smooth test functions. Obviously, for cubic trivariate polynomials, our method yields errors which are negligible. Examples for simple but non-trivial test functions are of truncated power type, e.g.

$$g(x,y,z) = (x-\tfrac{1}{2})_+^5 + (x(y-\tfrac{1}{2})(z-1))_+^5 + (x(y-1)(z-\tfrac{1}{2}))_+^5,$$

where $(x,y,z) \in \Omega_g = [0,1]^3$. Choosing $N = 8000$, for this simple test function we obtain the following errors of the approximating spline s_g: $\mathrm{err_{mean}} = 0.00000950$, $\mathrm{err_{data}} = 0.00010017$, and $\mathrm{err_{max}} = 0.00011770$.

We proceed by applying our method to some real-world data sets. An example for an interesting test in computer graphics is volume-based surface reconstruction. Here, we assume that samples of the *signed distance* to a surface are given, and the goal is to (locally) find a trivariate model representing the data whose zero-set approximates the surface. We consider this as a very intuitive test because the desired result is the shape of a known object and hence needs no extra interpretation. However, we currently do not directly compete with existing surface reconstruction methods, as the goal of our algorithm is more general, and our current setup is not optimized for the specific requirements of efficient reconstruction of surfaces. We consider an existing, high-resolution triangular mesh that was initially acquired by digitizing a real-world object as the Max-Planck bust. We then sample the signed distance to this surface not only in vicinity of the surface but *randomly distributed* in an extended bounding box. This way we generate a huge amount of data even for regions of the volume that are very distant from the zero-set and would be negligible for surface reconstruction. Hence, this is obviously a difficult test for any method. For the test, we explicitly want to cover the whole volume and stress the algorithm with large input. Given the Max-Planck data set, we randomly distribute $N = 2*10^6$ data points at which we measure the signed distance in a straight forward (but rough) way. The domain is decomposed into $58\times 97\times 73$ cubes, hence about 205 000 local least squares polynomial approximations are computed to determine the approximating spline s_{planck}. The average number of data points used for the local approximation in this test is 66. The approximation takes about 133 seconds, so more than 1500 local approximations are performed per second. Figures 5 and 6 visualize the results of our algorithm – we observe that the reconstructed surfaces inherit a visual smooth appearance from the trivariate splines. In addition, we provide a similar test using the *mechpart* data set, which a well-known benchmark in CAGD. The original data is a discrete height field over a two-dimensional 82×50 grid. A known difficulty for the reconstruction is the coarseness of this data in conjunction with the relatively high variation of the heights. As a test, we distributed $N = 512\,000$ points randomly in the volume bounding box and measured signed distances w.r.t. an almost regular (two-dimensional) triangulation of the data. Fig. 7 shows the reference data and zero-sets of spline approximations s_{mp} of the signed distance for two different partitions Δ.

We integrated our algorithm in the AMIRA visualization system [43], and note that all the visualizations given in this paper have been created with AMIRA. Using this framework enables experiments like the interactive approximation or visualization of local pieces of the spline as for instance single polynomials on prescribed tetrahedra of Δ. Fig. 8 shows an example, where we illustrate the behavior of the extension step of our algorithm.

N	n	$\mathrm{err}_{\mathrm{mean}}$	$\mathrm{err}_{\mathrm{data}}$	$\mathrm{err}_{\mathrm{max}}$	time
1 000	4	0.05612090	0.16392300	0.18640600	0.05
2 000	5	0.01758270	0.07843640	0.08240110	0.08
4 000	7	0.00247124	0.02297960	0.02867510	0.17
8 000	9	0.00076832	0.00766123	0.00922203	0.32
16 000	11	0.00030479	0.00374684	0.00429214	0.57
32 000	14	0.00010208	0.00147255	0.00168576	1.10
64 000	18	0.00003375	0.00054775	0.00062212	2.20
128 000	23	0.00001175	0.00021318	0.00023703	4.38
256 000	29	0.00000441	0.00008587	0.00009686	8.52
512 000	37	0.00000160	0.00003636	0.00004275	17.14
1 024 000	46	0.00000065	0.00001530	0.00001765	32.90
2 048 000	58	0.00000025	0.00000618	0.00000747	65.26
4 096 000	74	0.00000009	0.00000292	0.00000375	133.49

Table 1: *Approximation errors of the splines s_f and computation times in seconds.*

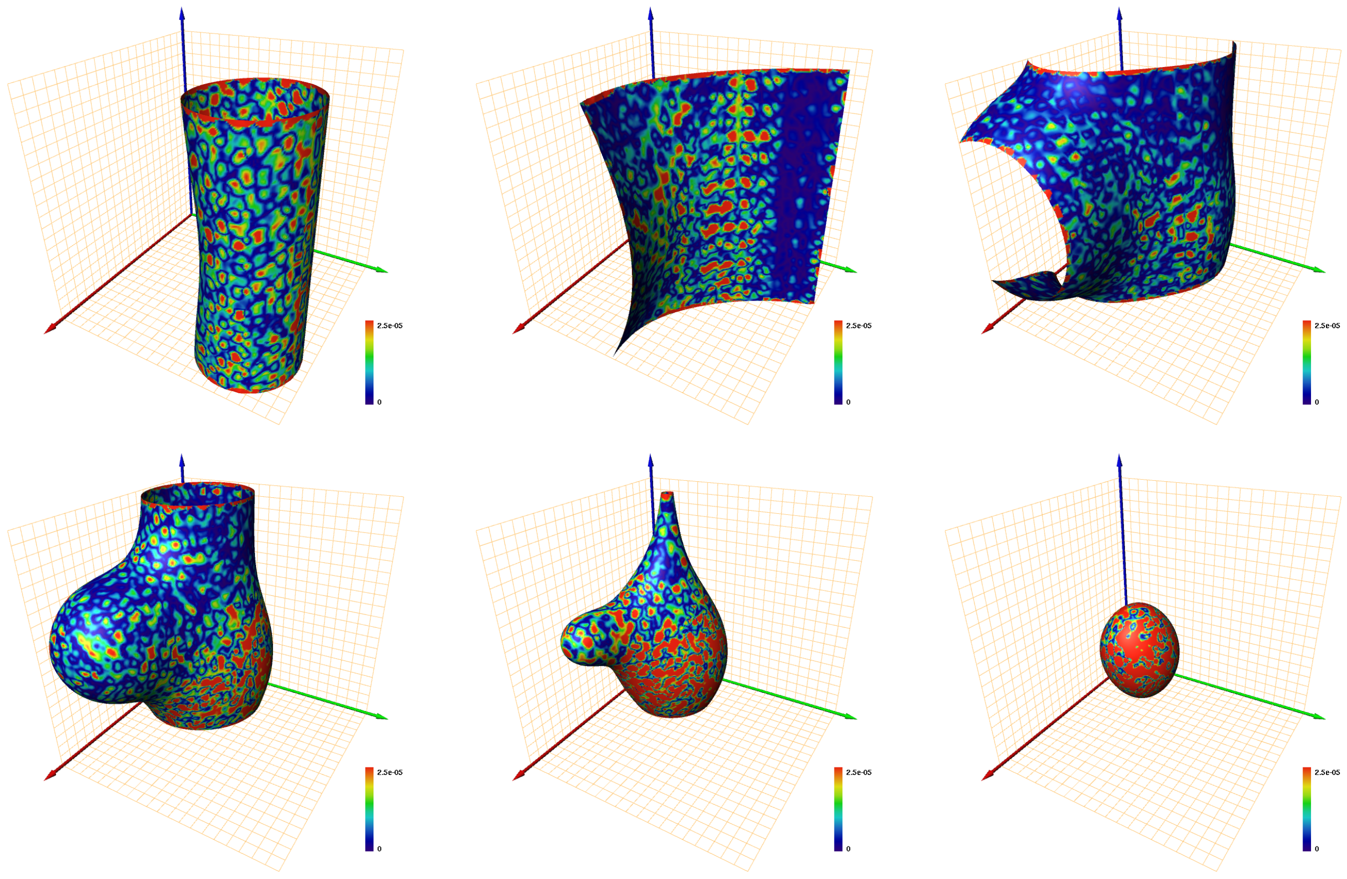

Figure 3: *Iso-surfaces of an approximation s_f to the test function f sampled at $N = 128\ 000$ randomly distributed points. The color code visualizes the approximation error for the surface points. The iso-values starting from top left are -0.1, 0, 0.1, 0.3, 0.5, and 0.8.*

6. Conclusions and Future Work

We present a new method for the efficient approximation of huge volumetric data sets distributed over an arbitrarily shaped domain. Our two-step algorithm is of linear algorithmic complexity w.r.t. the number of samples, it uses a natural, uniform tetrahedral partition of the domain which is given implicitly, it requires only the (independent) solution of small linear systems, it automatically adapts to local variation and distribution of the data, it enables the exploitation of well-established techniques known in CAGD, and it automatically smoothes noisy data. Our method is based on trivariate, cubic splines which can be efficiently represented by using its piecewise Bernstein-Bézier form. It is known that finding local constructions based on these spaces is a complex task. In this first approach to the problem of local approximation of general volumetric

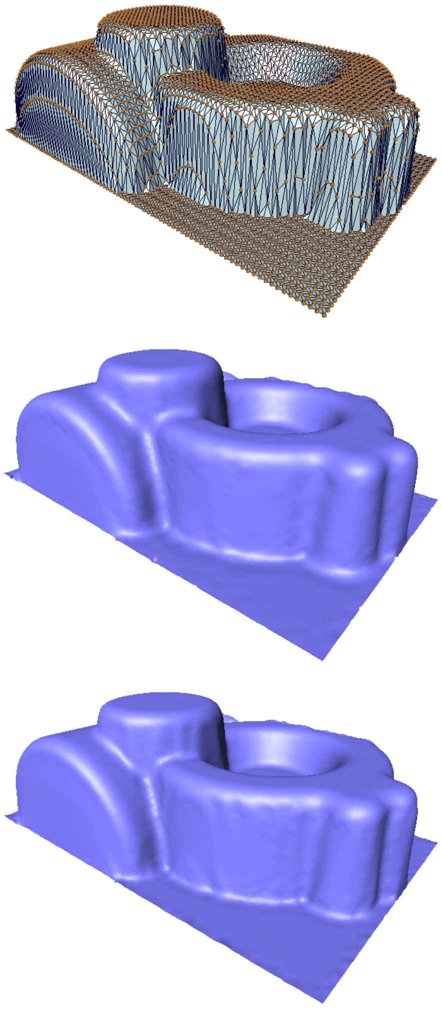

Figure 7: *Approximation of the* mechpart *data set. Top: The original surface that was sampled at* $N = 512\,000$ *randomly distributed points. Center and bottom: Zero-sets of the approximating splines* s_{mp}. *The domain is decomposed into* $64 \times 39 \times 22$ *and* $42 \times 26 \times 15$ *cubes, the computation times are* 26s *s and* 20 *s, respectively.*

data, we balance computational simplicity against overall smoothness. Motivated by recent results for gridded data ([38]), we construct a consistent cubic splines which satisfy almost all smoothness conditions. Our results confirm the high quality of the approximation and show visually smooth iso-surfaces of the reconstructed real-world objects.

In future work, we will focus on further improvements of the scattered volume data approximation. According to our current knowledge, overall smooth trivariate spline models require some additional degrees of freedom, so that we might either need a higher (but as small as possible) degree of the piecewise polynomials or different partitions. Both options are currently subject of intensive research, and at that point of time it seems to be a complex task to end up with an intuitive and computationally simple smooth reconstruction taking the requirements mentioned in the introduction into account. Further, we note that our current implementation can

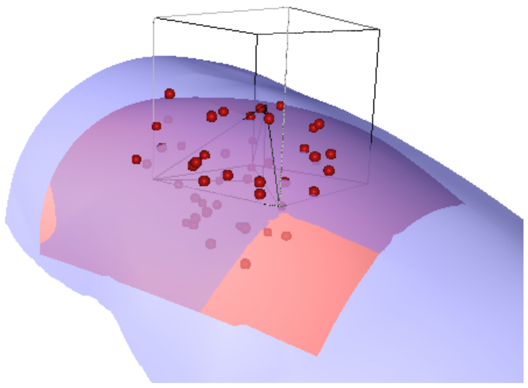

Figure 8: *An iso-surface of a least squares approximating polynomial piece on a black tetrahedron (transparent red) together with the positions of the samples (red spheres) that were used for the local fitting procedure (using a local portion* $\mathcal{X}_{loc}$ *of the extended* mechpart *data set). For this visualization the polynomial is extrapolated out of the respective tetrahedron as indicated. The transparent blue surface shows the same iso-surface for the complete approximating spline, which shows that the local difference is small and the extending behaves in a natural and smooth way.*

be improved by either applying the average operators introduced in [10] and perhaps by making use of different local approximations similar as in [9]. Moreover, one can think of integrating our method in straightforward hierarchical constructions over nested sequences of cube partitions, tuning the approach towards surface reconstruction and providing direct visualization of the trivariate splines, e.g. by ray-casting similar to [38].

References

[1] P. Alfeld. A trivariate C^1 Clough-Tocher interpolating scheme. *CAGD*, 1:169–181, 1984. 2, 3

[2] P. Alfeld. Scattered data fitting in two and three variables. In *Curves and Surfaces: Olso 1989*. Academic Press, 1990. 2

[3] C. de Boor. B-form basics. In G. Farin, editor, *Geometric Modelling*, pages 131–148. SIAM, 1987. 3

[4] M. D. Buhmann. Radial Basis Functions. *Acta Numerica*, pages 1–38, 2000. 1

[5] H. Carr, T. Möller, and J. Snoeyink. Simplicial subdivisions and sampling artifacts. In *IEEE Visualization 2001*, pages 99–106, 2001. 3

[6] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH'01*, pages 67–76, 2001. 1

[7] C.K. Chui. *Multivariate Splines*. CBMS 54, SIAM, 1988. 2, 3

[8] O. Davydov. On the approximation power of local least squares polynomials. In J. Levesley, I.J. Anderson, and J.C. Mason, editors, *Algorithms for Approximation IV*, pages 346–353, 2002. 4, 5

[9] O. Davydov, R. Morandi, and M. Sestini. Local hybrid ap-

proximation for scattered data fitting with bivariate splines. *(preprint)*, 2004. 9

[10] O. Davydov and F. Zeilfelder. Scattered data fitting by direct extension of local polynomials with bivariate splines. *Adv. Comp. Math. (to appear)*, 2004. 1, 2, 3, 4, 5, 6, 9

[11] H.Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *International Conference on Computer Vision (ICCV)*, pages 606–613, 2001. 1

[12] G. Farin. Triangular Bernstein-Bézier patches. *CAGD*, 3(2):83–127, 1986. 3

[13] A.T. Foley. Scattered Data Interpolation and Approximation with Error Bounds. *CAGD*, 3:163–177, 1986. 1

[14] R. Franke and H. Hagen. Least Squares Surface Approximation using Multiquadrics and Parameter Domain Distortion. *CAGD*, 16(3):177–196, 1999. 1

[15] J. Haber, F. Zeilfelder, O. Davydov, and H.-P. Seidel. Smooth approximation and rendering of large scattered data sets. In *IEEE Visualization 2001*, pages 341–347, 2001. 1, 2, 3, 4, 5

[16] T. Hangelbroek, G. Nürnberger, C. Rössl, H.-P. Seidel, and F. Zeilfelder. Dimension of C^1 splines on type-6 tetrahedral partitions. *Journal of Approximation Theory (to appear)*, 2004. 2, 3, 4

[17] D.J. Holliday and G.M. Nielson. Progressive volume models for rectilinear data using tetrahedral Coons volumes. In *Data Visualization 2000*, pages 83–92. Springer, 2000. 7

[18] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, 1993. 2, 3, 4, 5

[19] N. Kohlmüller, G.Nürnberger, and F. Zeilfelder. Construction of cubic 3D spline surfaces by Lagrange interpolation at selected points. In *Curve and Surface Fitting, Saint-Malo 2002*, pages 245–254, 2003. 2

[20] M.-J. Lai and A. Le Méhauté. A new kind of trivariate C^1 spline. *Adv. Comp. Math. (preprint)*, 2004. 2, 4

[21] P. Lancaster and K. Šalkauskas. *Curve and Surface Fitting*. Academic Press, 1986. 2

[22] S. K. Lodha and R. Franke. Scattered Data Techniques for Surfaces. In H. Hagen, G. Nielson, and F. Post, editors, *Proc. Dagstuhl Conf. Scientific Visualization*, pages 182–222, 1999. 1

[23] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH'87*, 21(5):79–86, 1987. 7

[24] W. Martin and E. Cohen. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. In *Solid Modelling and Applications 2001*, pages 234–240, 2001. 1

[25] G. Nürnberger. *Approximation by Spline Functions*. Springer, 1989. 1

[26] G. Nürnberger, V. Raveskaya, L.L. Schumaker, and F. Zeilfelder. Lagrange Interpolation by C^2-Splines of degree 7 on triangulations. In *Proc. Adv. Constr. Approx. (to appear)*, 2004. 2

[27] G. Nürnberger, V. Raveskaya, L.L. Schumaker, and F. Zeilfelder. Local Lagrange interpolation by bivariate splines of arbitrary smoothness. *(submitted)*, 2004. 2, 6

[28] G. Nürnberger, C. Rössl, H.-P. Seidel, and F. Zeilfelder. Quasi-interpolation by quadratic piecewise polynomials in three variables. *submitted*, 2004. 2

[29] G. Nürnberger, L.L. Schumaker, and F. Zeilfelder. Local Lagrange Interpolation by Bivariate C^1 Cubic Splines. In *Mathematical Methods In CAGD*, pages 393–404. Vanderbilt University Press, 2001. 2, 3

[30] G. Nürnberger, L.L. Schumaker, and F. Zeilfelder. Lagrange interpolation by C^1 cubic splines on triangulated quadrangulations. *Adv. Comp. Math. (to appear)*, 2004. 2

[31] G. Nürnberger and F. Zeilfelder. Developments in bivariate spline interpolation. *J. Comput. Appl. Math.*, 121:125–152, 2000. 2

[32] G. Nürnberger and F. Zeilfelder. Lagrange interpolation by bivariate C^1-splines with optimal approximation order. *Adv. Comp. Math. (to appear)*, 2004. 2

[33] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH'03*, pages 463–470, 2003. 1

[34] S. Park and K. Lee. High-dimensional trivariate NURBS representation for analyzing and visualizing fluid flow data. *Computers & Graphics*, 21(4):473–482, 1997. 1

[35] R. Pfeifle and H.-P. Seidel. Fitting triangular B-splines to functional scattered data. *Eurographics 1995*, 14(3):15–23, 1995. 1

[36] R. Pfeifle and H.-P. Seidel. Scattered data approximation with triangular B-splines. *Advance Course on Fairshape*, pages 253–263, 1996. 1

[37] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002. 2, 3, 4

[38] C. Rössl, F. Zeilfelder, G. Nürnberger, and H.-P. Seidel. Reconstruction of volume data with quadratic super splines. *Transactions on Visualization and Computer Graphics (to appear)*, 2004. 2, 3, 4, 9

[39] R. Schaback. Remarks on meshless local construction of surfaces. In *The Mathematics of Surfaces IV*, pages 34–58. Springer, 2000. 1

[40] L.L. Schumaker. Fitting surfaces to scattered data. *Approximation Theory II*, pages 203–268, 1976. 1, 2

[41] L.L. Schumaker and T. Sorokina. Quintic spline interpolation on type-4 tetrahedral partitions. *Adv. Comput. Math. (preprint)*, 2004. 2, 3, 6

[42] L.L. Schumaker and T. Sorokina. A trivariate box macro element. *(preprint)*, 2004. 2, 4, 6

[43] D. Stalling, M. Westerhoff, and H.-C. Hege. Amira — an Object Oriented System for Visual Data Analysis. In C.R. Johnson and C.D. Hansen, editors, *Visualization Handbook*. Academic Press (preprint), 2003. 7

[44] G. Turk and J. O'Brien. Modeling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, 2002. 1

[45] A. Worsey and G. Farin. An n-dimensional clough-tocher interpolant. *Const. Approx. 3*, (2):99–110, 1987. 2, 3

[46] F. Zeilfelder. Scattered data fitting with bivariate splines. In E. Quak A. Iske, M. Floater, editor, *Tutorials on Multiresolution and Geometric Modelling*, pages 243–286. Springer, 2002. 2, 3

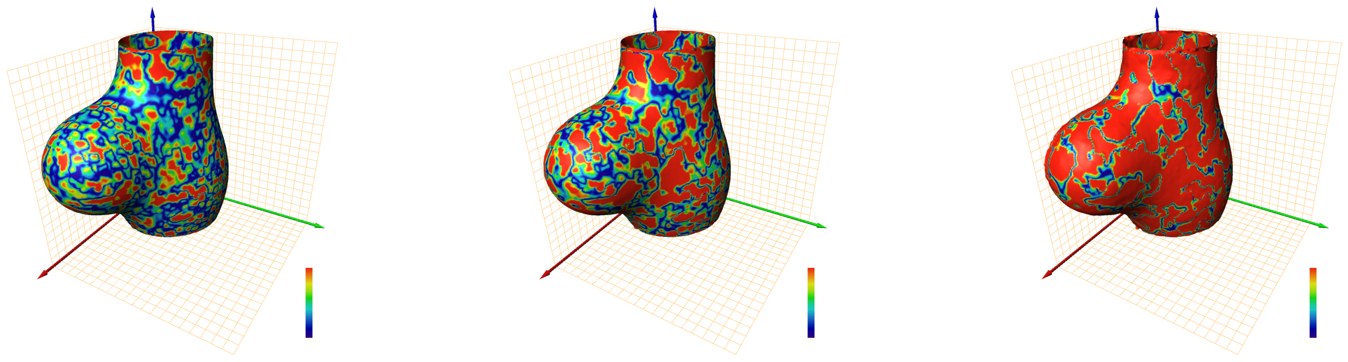

Figure 4: *Approximation of noisy data. As for Fig. 3, $N = 128\,000$ randomly distributed samples are used for the spline approximation, and uniformly distributed noise is added. The maximal amplitude of noise is (from left) 0.5%, 1% and 2.5% relative to the maximum range of f, where $f(x,y,z) \in [-0.25, 1.27], (x,y,z) \in \Omega_f$. We observe average approximation errors (err_{mean}) of about 0.0038, 0.0075, and 0.019 respectively. The pictures show the iso-surface with iso-value 0.3 extracted from the approximating splines s_f.*

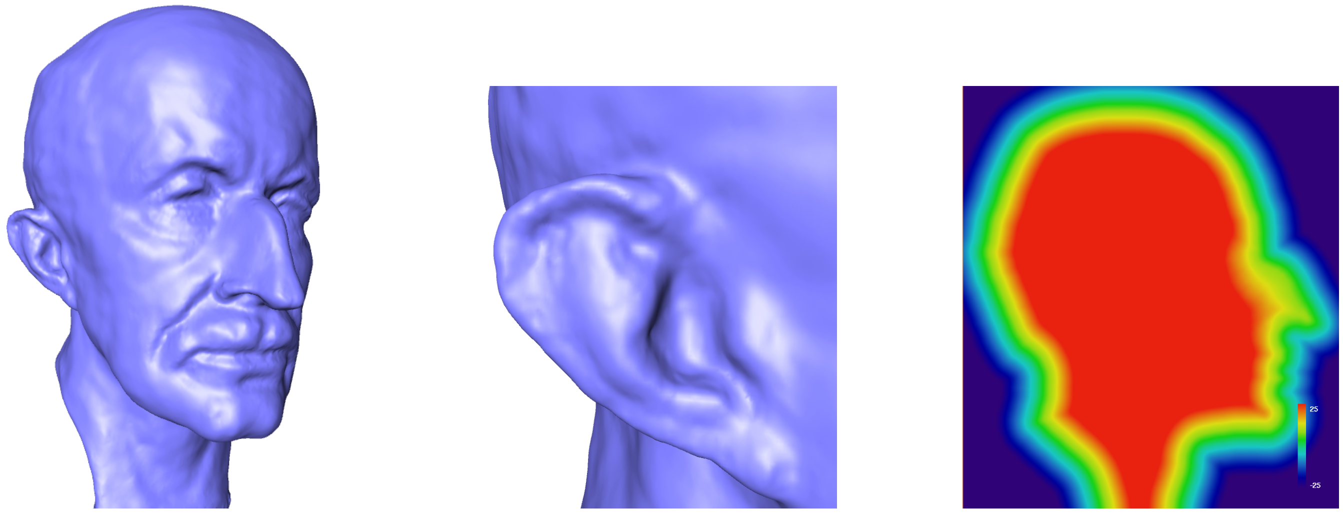

Figure 5: *Visualization of the spline approximation s_{planck} for the Max-Planck data set. The domain is decomposed into $58 \times 97 \times 73$ cubes. The left and center image show different views of the zero-set of the spline s_{planck}. The right image shows a slice through the approximative volume model $s_{planck}(x,y,const)$ for signed distance values, where the distances are color coded.*

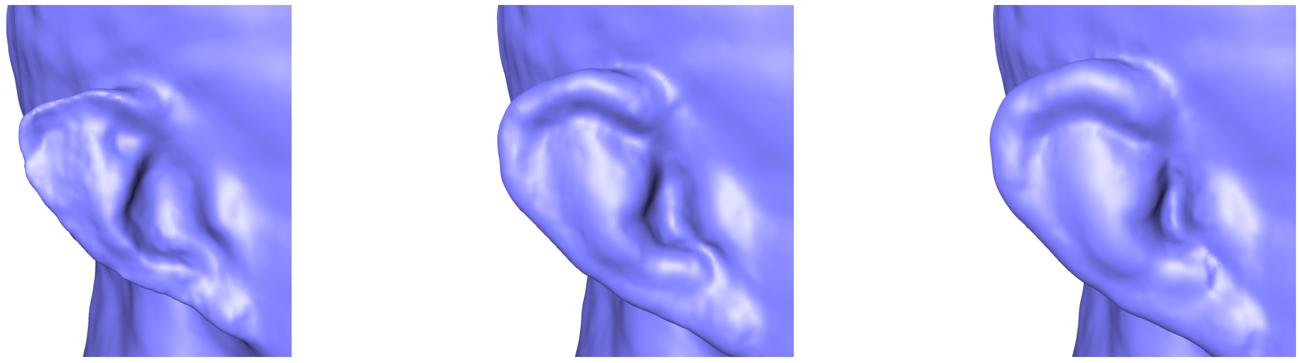

Figure 6: *Close-ups of different iso-surfaces of the spline approximation s_{planck} for the Max-Planck data set as used in Fig. 5. The iso-values are (from left) 2, -2, and -4. The corresponding iso-surfaces can be considered as offset surfaces to the zero-set of the spline s_{planck} shown in Fig. 5.*

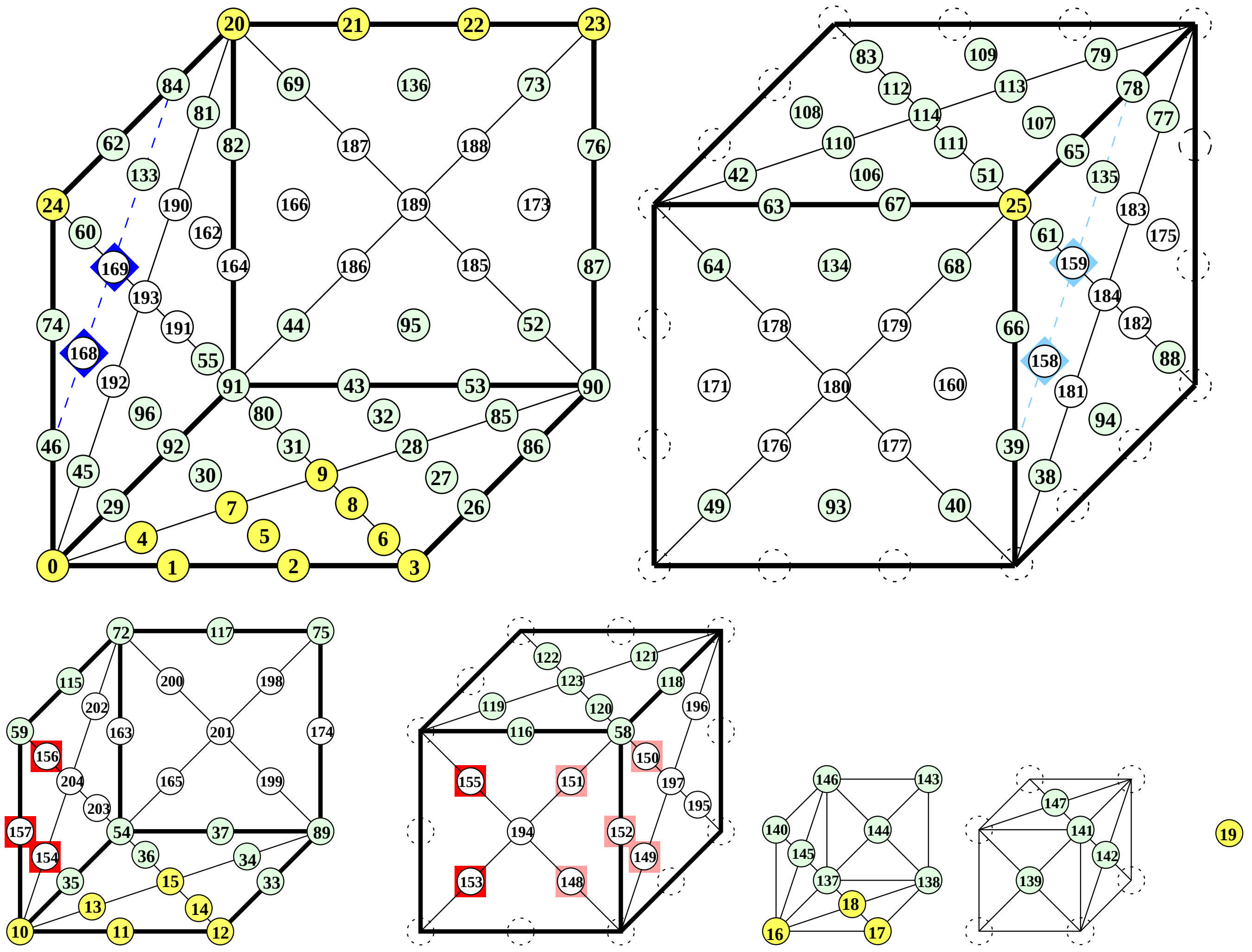

Figure 9: *The four layers* 3, 2, 1, 0 *of Bézier points for a black cube, each cube layer is cut into its back-facing and front-facing part. The dots show the* 175 *associated Bernstein-Bézier coefficients, and their numeric labels. The meaning of the colors are described in Sec. 4.2. The* 24 *polynomial pieces of the splines inside the cube are represented by the* 10, 6 *and* 3 *domain points in the respective triangles and the midpoint over all layers, e.g. points with label* 0–19 *on the black tetrahedron (cf. Sec. 3.1 and Fig. 1 (b)). Note that coefficients on the edges of the outermost layer (top row) coincide with those of some nearby black cubes. The innermost layer (numbered with* 0*) consists of a single point (bottom right). Fig. 10 shows the remaining coefficients of the white cubes.*

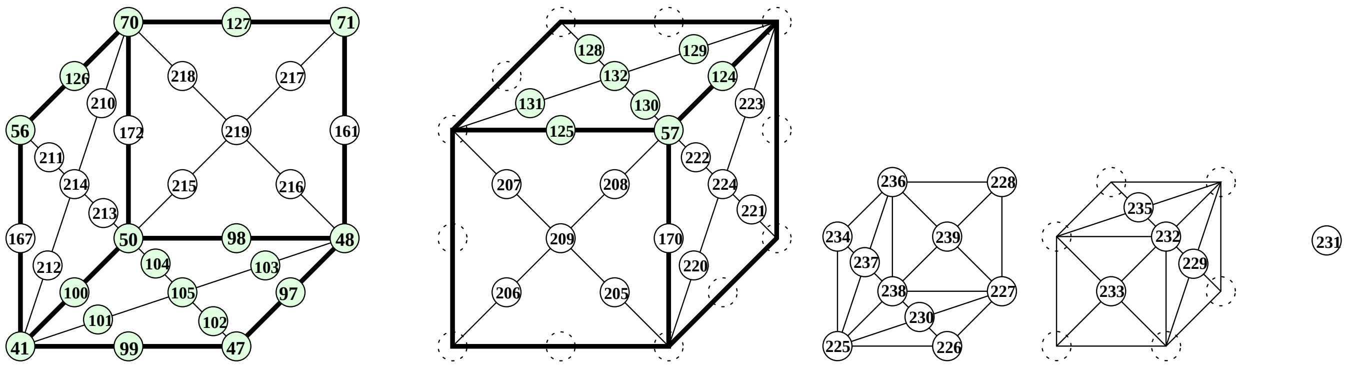

Figure 10: *The inner layers* 2, 1, 0 *of a white cube contain* 65 *Bézier points, see also Fig. 9 and Sec. 4.2. The white cube shares faces with six black cubes, and their coefficients on the outermost layer coincide by the continuity. For this reason it is sufficient to consider the three inner layers only.*