I N F O R M A T I K

Visualization of Volume Data with Quadratic Super Splines

Christian Rössl   Frank Zeilfelder
Günther Nürnberger   Hans-Peter Seidel

MPI–I–2003–04-006   April 2003

FORSCHUNGSBERICHT   RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85   66123 Saarbrücken   Germany

Authors' Addresses

Christian Rössl, Hans-Peter Seidel
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
`{roessl,hpseidel}@mpi-sb.mpg.de`

Günther Nürnberger, Frank Zeilfelder
Institut für Mathematik,
Universität Mannheim, Lehrstuhl IV
68131 Mannheim, Germany
`{nuernberger,zeilfeld}@euklid.math.uni-mannheim.de`

Abstract

We develop a new approach to reconstruct non-discrete models from gridded volume samples. As a model, we use quadratic, trivariate super splines on a uniform tetrahedral partition Δ. The approximating splines are determined in a natural and completely symmetric way by averaging local data samples such that appropriate smoothness conditions are automatically satisfied. On each tetrahedron of Δ, the spline is a polynomial of total degree two which provides several advantages including the efficient computation, evaluation and visualization of the model. We apply Bernstein-Bézier techniques well-known in Computer Aided Geometric Design to compute and evaluate the trivariate spline and its gradient. With this approach the volume data can be visualized efficiently e.g. with isosurface ray-casting. Along an arbitrary ray the splines are univariate, piecewise quadratics and thus the exact intersection for a prescribed isovalue can be easily determined in an analytic and exact way. Our results confirm the efficiency of the method and demonstrate a high visual quality for rendered isosurfaces.

Keywords

volume rendering, reconstruction, quadratic super splines, tetrahedral partition, Bernstein-Bézier techniques, isosurface rendering, ray-casting

Visualization of Volume Data with Quadratic Super Splines

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

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

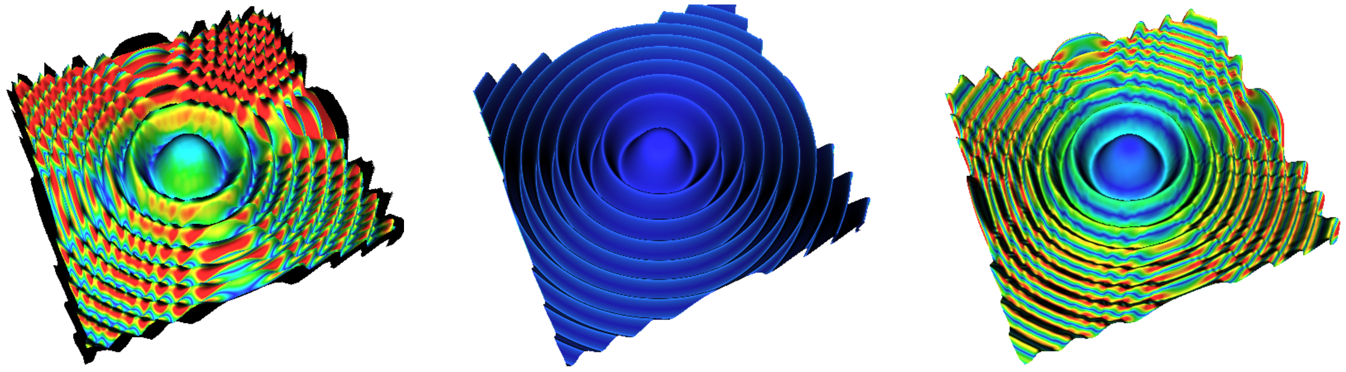

Figure 1: Isosurfaces of the synthetic Marschner-Lobb benchmark [22] (41^3 samples, isovalue $\frac{1}{2}$). The approximation error to the function in the uniform norm is color coded (from red=0.075 to blue=0) for the standard trilinear model (left) and our new quadratic super splines (right). The center image shows the original function, which is in fact rendered from our model using $(4 \times 41)^3$ samples. At this resolution we observe no visual difference to the perfect reconstruction.

Abstract

We develop a new approach to reconstruct non-discrete models from gridded volume samples. As a model, we use quadratic, trivariate super splines on a uniform tetrahedral partition Δ. The approximating splines are determined in a natural and completely symmetric way by averaging local data samples such that appropriate smoothness conditions are automatically satisfied. On each tetrahedron of Δ, the spline is a polynomial of total degree two which provides several advantages including the efficient computation, evaluation and visualization of the model. We apply Bernstein-Bézier techniques well-known in Computer Aided Geometric Design to compute and evaluate the trivariate spline and its gradient. With this approach the volume data can be visualized efficiently e.g. with isosurface ray-casting. Along an arbitrary ray the splines are univariate, piecewise quadratics and thus the exact intersection for a prescribed isovalue can be easily determined in an analytic and exact way. Our results confirm the efficiency of the method and demonstrate a high visual quality for rendered isosurfaces.

CR Categories: G.1.2 [Numerical analysis]: Approximation—Spline and piecewise polynomial approximation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Splines

Keywords: volume rendering, reconstruction, quadratic super splines, tetrahedral partition, Bernstein-Bézier techniques, isosurface rendering, ray-casting

1 Introduction

Three dimensional scalar fields defined over a set of discrete samples arise in many applications such as scientific visualization or medical imaging. Volume rendering is an important technique for visualizing these data sets. A fundamental problem is to find a *non-discrete model* or *reconstruction* of the data, e.g. a density function. An ideal model would provide both, efficient approximation of huge, often noisy data sets, as well as efficient and exact evaluation of function values and gradients which are required for high-quality visualization. Finding an appropriate model is a extremely difficult task for general data sets.

The problem is less complex if the data is structured so that the samples are arranged on a regular three dimensional grid. For instance, CT or MRI sensors, seismic applications, or results from numerical simulations typically generate this type of gridded data which must subsequently be visualized by volume rendering. In this paper, we present new models of such gridded volume data, namely quadratic, trivariate super splines on uniform tetrahedral partitions which yield efficient approximations that can be evaluated and implemented easily.

Reconstruction of volume data has been an active area of research for the last decades and many different models have been proposed. The reconstruction of a discrete sampling, in general, is well studied in signal processing, and Fourier analysis leads to optimal models. However, optimal models are often not feasible in practice as they are based on global properties. In order to keep computational costs low, *local* methods have been studied extensively (cf. [20, 22, 25, 26]). It turns out that local (piecewise) polynomial constructions are often preferred for their simplicity, efficiency and satisfying reconstructions.

In this context, the simplest model is a piecewise constant approximation based on the closest sample or on some averaging of nearby samples. The next natural model is to use trivariate linear polynomials $\sum_{i+j+k\le 1} a_{i,j,k}\, x^i y^j z^k$, where $a_{i,j,k} \in \mathbb{R},\ i+j+k \le 1$. Using these functions, a *tetrahedral partition* Δ is needed, and the model becomes a linear, *trivariate spline on* Δ (see [3, 11, 12], for instance). More sophisticated models are needed if gradient information is required, e.g. for high quality shading. One of the most popular models in volume visualization is to use trilinear interpolants $\sum_{i,j,k=0,1} a_{i,j,k}\, x^i y^j z^k$, where $a_{i,j,k} \in \mathbb{R},\ i,j,k = 0,1$, i.e. piecewise polynomials with *total degree* three of specific type (cf. [22, 32], and the references therein). Approaches of this type often use central differences of the surrounding data samples to faithfully determine the gradient at a given point location.

Alternatively, models have been constructed which satisfy smoothness properties. In this case, the necessary gradient information is directly available from the model, but the data stencil needed for the reconstruction generally increases. In order to keep computational costs low, local methods have been proposed. In some of these approaches (cf. [20, 22, 25, 26]) the models are tricubic splines (sometimes also called cubic filters), i.e. piecewise polynomials of the form $\sum_{i,j,k=0}^{3} a_{i,j,k}\, x^i y^j z^k$, where $a_{i,j,k} \in \mathbb{R},\ i,j,k = 0,\dots,3$, are used. These are special polynomials of total degree nine. Recently, smooth approximation models using triquadratic tensor splines have been proposed to further reduce the polynomial degree. In this case, the data stencil consists of 27 grid points which coincides with the number of coefficients $a_{i,j,k} \in \mathbb{R},\ i,j,k = 0,\dots,2$. These methods are based on the piecewise monomial representation (cf. [2, 27]) or on the B-spline expansion of tensor splines (cf. [37]), and the total degree of the polynomial pieces is six.

Moreover, reconstructions with high smoothness are discussed in [26, 37], a mathematical framework using NURBS was developed in [23], and trivariate Coons patches were proposed in [15]. The literature shows that designing a model for the visualization of volume data is always a compromise between computational efficiency and visual quality where the most successful methods are based on local reconstructions. For further information on the field, we refer the interested reader to the recent books [1, 6], the surveys [5, 17, 28, 24, 36], and the references therein.

In this paper, we present a new approach to efficiently visualize gridded volume data using a *local spline model*. In contrast to existing approaches, the splines used here are piecewise polynomials of *lowest possible total degree*, namely the polynomial pieces have the form $\sum_{i+j+k\le 2} a_{i,j,k}\, x^i y^j z^k$, where $a_{i,j,k} \in \mathbb{R},\ i+j+k \le 2$. This means that the total degree is *two*. The *quadratic splines* are defined with respect to a tetrahedral partition Δ, hence its polynomial pieces are given on tetrahedra. Splines of this natural type have not yet been studied in the context of local volume data reconstruction. Based on our theoretical investigations of the structure concerning smooth trivariate splines of arbitrary degree (cf. [14], and the references therein) and the facts known for bivariate splines (see [30], and the references therein), we choose an appropriate uniform tetrahedral partition Δ (see Fig. 2) and design a *super spline model* which we show to be appropriate for efficient volume visualization. We develop a natural and completely symmetric reconstruction method for these trivariate splines. Their coefficients are computed locally and directly by *repeated averaging* of the given data while appropriate smoothness properties necessary for the visualization are automatically satisfied. Here, we take advantage of the (trivariate) *Bernstein-Bézier representation* of the quadratic polynomial pieces. This piecewise representation allows us to exploit the Bernstein-Bézier techniques well known from Computer Aided Geometric Design (CAGD) (see e.g [16]) to efficiently represent, compute, evaluate and visualize our volume spline model.

Our approach allows the efficient and high-quality visualization of volume data which we illustrate by rendering isosurfaces of well-known synthetic and measured test data sets using *ray-casting*. Along an arbitrary ray the splines are univariate, piecewise quadratics and consequently its exact intersection for a prescribed isovalue can be easily determined in an analytic and exact way determined with by solving *quadratic equations*. Note that all the methods described above (except for those based on linear, trivariate splines on Δ) obviously need to solve higher order equations through either approximative numerical methods, or – for cubic and quartic equations – implementation of Cardano's and Ferrari's exact formulae, respectively (cf. [35]). Finally, the gradient, necessary for quality shading, is determined efficiently n our method thanks to Bernstein-Bézier techniques.

The paper is organized as follows. In Section 2, we describe quadratic super splines on a uniform tetrahedral partition Δ and its piecewise Bernstein-Bézier form. The new approach based on these splines is given in Section 3. We proceed by describing computational aspects for the further processing of the splines including the computation of point locations in Δ and barycentric coordinates (Section 4), and the efficient evaluation of the polynomial pieces and their derivatives (Section 5). Precise and efficient ray-isosurface intersection for rendering is discussed in Section 6. The paper concludes with computational results (Section 7) and comments on future investigations (Section 8).

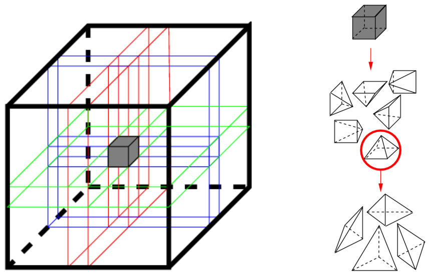

Figure 2: The tetrahedral partition Δ is obtained by subdividing each cube of $\diamondsuit$ uniformly into 24 tetrahedra.

2 Quadratic Trivariate Super Splines and Bernstein-Bézier Form

Our reconstruction is based on quadratic splines, i.e. piecewise polynomials of total degree two, on a natural uniform tetrahedral partition Δ. The partition Δ is carefully chosen such that the local reconstruction described in the next section is possible. According to our experience, a simpler tetrahedral partition (such that each cube is subdivided into five or six tetrahedra, for instance) would not allow the construction of quadratic splines with the same properties as the described in the next section. On the other hand, we observed in various tests that these properties are essential for high-quality visualization.

Let $\diamondsuit$ be a uniform cube partition of the cubic domain $\Omega = [0,n]^3 \subseteq \mathbb{R}^3$, where every cube $Q \in \diamondsuit$ has side length 1. We split each of the n^3 cubes Q into six (Egyptian) pyramids by connecting its center point v_Q with the four vertices of every square faces 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 $Q \in \diamondsuit$ contains congruent 24 tetrahedra. A more intuitive way to describe Δ is to say that Δ is the tetrahedral partition obtained by slicing Ω with the six planes which contain opposite edges of Ω. The partition Δ is a generalization of the four-directional mesh which is well-known in the bivariate setting (cf. [7, 8, 13]). Fig. 2 illustrates the construction of Δ. We are interested in splines on Δ.

The space of *quadratic super splines with respect to* Δ is now defined by

$$\mathscr{S}_2(\Delta) = \{\, s \in C(\Omega) : s|_T \in \mathscr{P}_2, \text{ for all } T \in \Delta, \text{ and } \\ s \text{ is smooth at } v, \text{ for all } v \text{ vertex of } \diamondsuit \,\},$$

where $\mathscr{P}_2 = \operatorname{span}\{x^i y^j z^k :\ i,\ j,\ k \ge 0,\ i+j+k \le 2\}$ denotes the ten-dimensional space of quadratic polynomials, i.e. the space of *trivariate polynomials of total degree two*. In our approximating method described in the next section we use splines from

$\mathscr{S}_2(\Delta)$ which possess many additional, natural smoothness properties. Mathematically speaking, this means that we deal with appropriate subspaces of $\mathscr{S}_2(\Delta)$, where the number of free parameters is considerably lower.

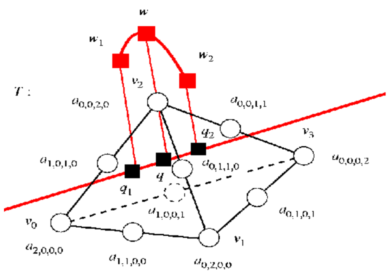

Figure 3: The ten Bézier points (white dots) of a quadratic polynomial inside a tetrahedron T are associated with the Bernstein-Bézier coefficients. The restriction of this trivariate polynomial piece to an arbitrary ray (red line) is obviously a quadratic, univariate polynomial (red curve) which is uniquely determined by the values (red boxes) at three points (black boxes).

A spline $s \in \mathscr{S}_2(\Delta)$ can be written in its piecewise *Bernstein-Bézier form* (cf. [7, 4, 10, 16]), i.e. for every tetrahedron $T = [v_0, v_1, v_2, v_3]$ of Δ, we have

$$s|_T = p = \sum_{i+j+k+l=2} a_{i,j,k,l}\, B_{i,j,k,l}, \tag{1}$$

where $a_{i,j,k,l} \in \mathbb{R}$ are called the *Bernstein-Bézier coefficients* of the polynomial piece $p \in \mathscr{P}_2$ associated with the *Bézier points* $\frac{i}{2}\, v_0 + \frac{j}{2}\, v_1 + \frac{k}{2}\, v_2 + \frac{l}{2}\, v_3$, $i+j+k+l = 2$. (See Fig. 3, where we indicate the ten Bernstein-Bézier coefficients by white dots.) Here, the ten polynomials

$$B_{i,j,k,l} = \frac{2!}{i!j!k!l!}\, \lambda_0^i \lambda_1^j \lambda_2^k \lambda_3^l \in \mathscr{P}_2,, \qquad i+j+k+l = 2,$$

are the *quadratic Bernstein polynomials with respect to* T, and $\lambda_\nu \in \mathscr{P}_1 = \text{span}\ \{1, x, y, z\}$, $\nu = 0, \ldots, 3$, are the *barycentric coordinates with respect to* T. The barycentric coordinates are determined by the interpolation conditions $\lambda_\nu(v_\mu) = \delta_{\nu,\mu}$, $\mu = 0, \ldots, 3$ ($\delta_{\nu,\mu}$ denotes Kronecker's symbol), and it is easy to see that for any point $q \in \mathbb{R}^3$, the barycentric coordinates $\lambda_\nu(q) \in \mathbb{R}$, $\nu = 0, \ldots, 3$, of q are uniquely determined as the solution of the 4×4 linear system

$$\lambda_0(q) \binom{v_0}{1} + \lambda_1(q) \binom{v_1}{1} + \lambda_2(q) \binom{v_2}{1} + \lambda_3(q) \binom{v_3}{1} = \binom{q}{1}. \tag{2}$$

The Bernstein-Bézier representation (1) for (piecewise) polynomials is well known and frequently used in CAGD (cf. [16, 33]) and multivariate spline theory (see, for instance [7, 8, 14, 18, 19, 29, 30, 34]). In the remainder of this paper we use this piecewise representation, where we take advantage of the Bernstein-Bézier techniques to efficiently represent, construct, evaluate and visualize the volume spline model described in the next section.

3 Reconstruction by Super Splines

Given gridded volume data, i.e. the data points are of the form $(\frac{2i+1}{2}, \frac{2j+1}{2}, \frac{2k+1}{2}) \in \mathbb{R}^3$, with corresponding data values $f_{i,j,k} \in \mathbb{R}$, $i, j, k = -1, \ldots, n$, the outline of our reconstruction method is as follows: The coefficients in the piecewise representation (1) of the reconstruction s from $\mathscr{S}_2(\Delta)$ are determined by repeated averaging of the data values. First, for every vertex v of $\diamondsuit$, we determine

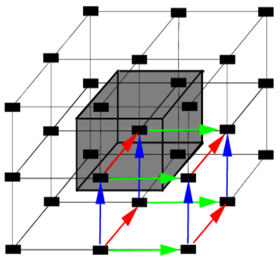

Figure 4: In each cube $Q \in \diamondsuit$ (grey) the splines are reconstructed by using a stencil of 27 data samples (black boxes). The derivative of the splines at the grid points of $\diamondsuit$ in each of the three space directions are determined as the average of four differences. For instance, the x derivative at the lower right vertex of Q is obtained from averaging the differences illustrated by the green arrows. Similarly, the y derivative and z derivative at this point are obtained from averaging the red and blue arrows, respectively.

the Bernstein-Bézier coefficients of s close to v by using an averaging of the data values at the center points of the eight cubes which have v as a common vertex. This uniquely determines the value $s(v)$ and the three derivatives $(\frac{\partial s}{\partial x})(v)$, $(\frac{\partial s}{\partial y})(v)$, and $(\frac{\partial s}{\partial z})(v)$. Then, we use repeated averaging of the Bernstein-Bézier coefficients associated with these nodal values at the eight vertices of every cube $Q \in \diamondsuit$ to uniquely determine the 65 coefficients of $s|_Q$ in its piecewise representation (1) while satisfying additional natural, appropriate smoothness conditions. Hence, similarly to [2, 27], where a triquadratic tensor spline model is used as a reconstruction, the 27 data values at the centers of the cubes which have a non-empty intersection with Q are needed to reconstruct $s|_Q$ for every cube $Q \in \diamondsuit$ (see Fig. 4). Note that although in one variable our method would coincide with these approaches, however this is completely different in the multivariate case. An illustration of our reconstruction is given in Fig. 5, where we show the Bézier points from one face of the three different layers within an arbitrary cube Q of $\diamondsuit$. Here, the different colors indicate the order of determining the corresponding coefficients of the splines as it is described below.

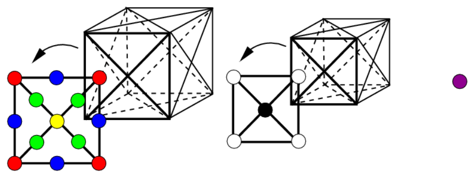

Figure 5: The Bernstein-Bézier coefficients of the polynomial pieces of s associated with the Bézier points on the three different layers are determined in the following order: blue, red, green, yellow, white, black, magenta.

The details of our natural and completely symmetric reconstruction are as follows. Let $Q \in \diamondsuit$ be an arbitrary cube. For every edge e of Q, we first determine the Bernstein-Bézier coefficient a_e associated with the Bézier point at the midpoint of e (blue dot in Fig. 5) by averaging the four data values f_0, f_1, f_2, f_3, which correspond to the data points at the center points of the four cubes in $\diamondsuit$ with common edge e, i.e. we set

$$a_e = \tfrac{1}{4}\, (f_0 + f_1 + f_2 + f_3). \tag{3}$$

We then determine the Bernstein-Bézier coefficient a_v associated with the Bézier point at every vertex v of Q (red dot in Fig. 5). This is done by choosing two edges e_1 and e_2 with endpoint v which lie on the same line segment of $\diamondsuit$ and by averaging the two Bernstein-Bézier coefficients a_{e_1}, a_{e_2}, i.e. we set $a_v = \frac{1}{2}\,(a_{e_1} + a_{e_2})$. Note that a_v is uniquely determined and independent from the chosen line segment of $\diamondsuit$ since for each of the three possible choices of edges e_1 and e_2 with endpoint v, we obtain due to uniformity

$$a_v = \tfrac{1}{8}\,(f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7), \tag{4}$$

where $f_0, \ldots, f_7$, are the data values at the center points of the eight cubes with vertex v. Moreover, the derivatives $(\frac{\partial s}{\partial x})(v)$, $(\frac{\partial s}{\partial y})(v)$, and $(\frac{\partial s}{\partial z})(v)$ are uniquely determined in an automatic way. For instance, it follows from a standard relation (cf. [10]) of $(\frac{\partial s}{\partial x})(v)$ with the two Bernstein-Bézier coefficients associated with the points $v = (i, j, k)$ and $(\frac{2i+1}{2}, j, k)$ that

$$(\tfrac{\partial s}{\partial x})(v) = \tfrac{1}{4}\,\Big(f_{i,j-1,k-1} - f_{i-1,j-1,k-1} + f_{i,j,k-1} - f_{i-1,j,k-1} + f_{i,j-1,k} - f_{i-1,j-1,k} + f_{i,j,k} - f_{i-1,j,k}\Big). \tag{5}$$

This means that $(\frac{\partial s}{\partial x})(v)$ is determined as an average of four simple differences which approximate the derivative in x-direction. Similar interpretations hold for $(\frac{\partial s}{\partial y})(v)$ and $(\frac{\partial s}{\partial z})(v)$ (see Fig. 4). Note that in contrast to the standard central differences approach in volume graphics for approximating derivative information, as in [2, 27], no information from an intermediate data sample is lost.

We proceed by setting the remaining five Bernstein-Bézier coefficients associated with points on each of the six faces of Q. Let $\mathscr{F}$ be a square face of Q, d the point where the two diagonals in $\mathscr{F}$ intersect, and m_1, m_2, two midpoints of edges in the interior of $\mathscr{F}$ which lie on the same diagonal in $\mathscr{F}$. The Bernstein-Bézier coefficient a_{m_1} associated with the Bézier point m_1 (green dot in Fig. 5) is determined by averaging the two Bernstein-Bézier coefficients a_{e_1}, a_{e_2}, where e_1 and e_2 are the edges of $\mathscr{F}$ from $\diamondsuit$ which intersect at the vertex of Q closest to m_1, i.e. we set

$$a_{m_1} = \tfrac{1}{2}\,(a_{e_1} + a_{e_2}). \tag{6}$$

Analogously, we determine the coefficient a_{m_2}. Then, we set for the Bernstein-Bézier coefficient a_d associated with the Bézier point d (yellow dot in Fig. 5), $a_d = \frac{1}{2}\,(a_{m_1} + a_{m_2})$. It is well known in bivariate spline theory (see e.g. [8, 29, 30]) that a_d is uniquely determined independently from the two possible choices for m_1 and m_2. Moreover, this setting implies the smoothness within the faces of Q. In particular, the directional derivative $(\frac{\partial s}{\partial \varsigma})(d)$ is uniquely determined, where ς is an arbitrary vector in three-dimensional space which lies in the plane through the origin parallel to $\mathscr{F}$.

We proceed by setting the remaining 15 Bernstein-Bézier coefficients associated with points from the interior of Q. First, let c be a midpoint of an edge of Δ which connects the center v_Q with a vertex v of Q, and let e be the common edge of any two faces $\mathscr{F}$, $\mathscr{F}^*$ of Q with vertex v. Moreover, set m_1 and m_1^* to the midpoints of the edges in the interior of $\mathscr{F}$ and $\mathscr{F}^*$ with endpoint v, respectively. Using the same notation as above, the Bernstein-Bézier coefficient a_c associated with c (white dot in Fig. 5) is determined by

$$a_c = (a_{m_1} + a_{m_1^*}) - \tfrac{1}{2}\,(a_v + a_e).$$

We note that it follows from a standard relation (cf. [4, 10], see also [14]) that this setting (together with (6)) *now* guarantees that $s \in \mathscr{S}_2(\Delta)$. Moreover, a_c is uniquely determined independent of the three possible choices of $\mathscr{F}$ and $\mathscr{F}^*$. The coefficient a_g associated with the midpoint g of the edge which connects the intersection point d of the diagonals of a face $\mathscr{F}$ of Q with v_Q (black dot in Fig. 5) is now determined by setting

$$a_g = \tfrac{1}{4}\,(a_{c_0} + a_{c_1} + a_{c_2} + a_{c_3}),$$

where $c_0, \ldots, c_3$, are the midpoints of the edges which connect the vertices of $\mathscr{F}$ with v_Q. This setting is motivated by the fact that it is the average of two smoothness conditions which would have been satisfied simultaneously by an overall smooth spline (cf. [14]), and hence the approximation properties of the model are preserved by an argument of weak-interpolation (cf. [31]). It remains to determine the Bernstein-Bézier coefficient a_{v_Q} at the center v_Q of Q (magenta dot in Fig. 5). We set

$$a_{v_Q} = \tfrac{1}{3}\,(a_{g_0} + a_{g_1} + a_{g_2} + a_{g_3} + a_{g_4} + a_{g_5}) - \tfrac{1}{8}\,(a_{c_0} + a_{c_1} + a_{c_2} + a_{c_3} + a_{c_4} + a_{c_5} + a_{c_6} + a_{c_7}),$$

where $g_0, \ldots, g_5$, are the midpoints of the edges which connect the intersection point of the diagonals of the six faces of Q with v_Q, and $c_0, \ldots, c_7$, are the midpoints of the eight edges which connect the vertices of Q with v_Q. The latter setting is motivated by the fact that it is the average of twelve smoothness conditions which would have been satisfied simultaneously by an overall smooth spline (cf. [14]) and hence the approximation properties of the model are preserved by an argument of weak-interpolation type (cf. [31]).

Now all the coefficients of the spline s are set appropriately. The computation of the 65 coefficients for a single cube $Q \in \diamondsuit$ of $s|_Q$ requires 66 multiplications with constants and 121 additions. The implementation of the model is straightforward.

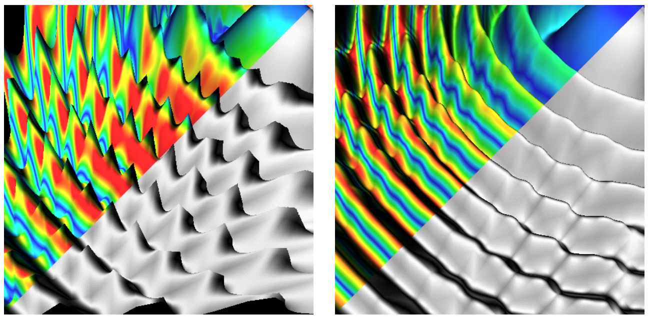
Figure 6: Zoomed regions of Fig. 1 (same color code for upper image parts): Trilinear (left) and quadratic (right) reconstruction. The data set contains only 41^3 samples of the high-frequency Marschner-Lobb test function [22].

4 Point Location and Barycentric Coordinates

In order to compute the value of the spline s and the gradient at a given point $q \in \Omega$ (see Section 5), we need to know the location of q in the partition Δ and its local barycentric coordinates. Hence, we have to determine a cube $Q \in \diamondsuit$ with $q \in Q$, a tetrahedron $T \subseteq Q$ with $q \in T$, and the barycentric coordinates $\lambda_\nu(q) \in \mathbb{R}$, $\nu = 0, \ldots, 3$, of q with respect to T.

The indices of the cube Q with $q \in Q$ are found by rounding the coordinates of $q = (x_q, y_q, z_q)$, i.e. we have $Q = Q_{[x_q],[y_q],[z_q]}$, where $[b]$ denotes the maximal integer $\leq b$. The uniformity of Δ allows a translation of q such that the remaining computations can be performed for (the tetrahedral partition of) the unit cube $Q_0 = [-\frac{1}{2}, \frac{1}{2}]^3$, hence in what follows we may assume that $q \in Q_0$.

For finding the tetrahedron which contains q, we use the observation mentioned in Section 2 that the partition of Q_0 in 24 congruent tetrahedra is obtained by slicing with the six planes

$$P_\nu(x, y, z) = 0, \qquad \nu = 0, \ldots, 5, \tag{7}$$

where

$$P_0(x,y,z) = x+y, \quad P_1(x,y,z) = x-y, \quad P_2(x,y,z) = x+z,$$
$$P_3(x,y,z) = x-z, \quad P_4(x,y,z) = y+z, \quad P_5(x,y,z) = y-z.$$

The orientation of q with respect to these planes is determined by performing one addition to compute $P_\nu(q)$ followed by a sign check for each of the six planes. This gives a 6-bit binary code for the orientation of q and the tetrahedron $T \subseteq Q_0$ with $q \in T$ is found by a simple table lookup. The whole operation requires six additions, six sign checks and five bit shifts.

For determining the barycentric coordinates $\lambda_\nu(q),\ \nu = 0,\ldots,3$, of q with respect to T, the vertices of $T = [v_0, v_1, v_2, v_3]$ are organized such that v_0 is the origin, v_1 and v_2 are two corner vertices of Q_0, and v_3 is the intersection point of the diagonals in a face of Q_0, and we use the precomputed general solution of the system (2). For instance, if $v_1 = (-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, $v_2 = (\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, and $v_3 = (0, 0, -\frac{1}{2})$, we have

$$\lambda_0(q) = 1 + 2\, z_q, \quad \lambda_1(q) = -x_q - y_q, \quad \lambda_2(q) = x_q - y_q,$$

and $\lambda_3(q) = 2\ (y_q - z_q)$. Similar expressions for the barycentric coordinates involving five of the variable factors from the ordered list

$$L = (L_\nu)_{\nu=0}^{5} = [x_q,\ y_q,\ z_q,\ -x_q,\ -y_q,\ -z_q],$$

are obtained for the other 23 tetrahedra in Q_0. We exploit this simple fact for generating another lookup table with 24 entries of the precomputed solutions. If T is the tetrahedron from the above example, its entry E in this table is given by

$$E = [\ (2,2)\ |\ (3,4)\ |\ (0,4)\ ],$$

with the interpretation that the barycentric coordinates of q with respect to T are computed from L by setting

$$\lambda_0(q) = 1 + L_2 + L_2, \quad \lambda_1(q) = L_3 + L_4, \quad \lambda_2(q) = L_0 + L_4,$$

and $\lambda_3(q) = 1 - \lambda_0(q) - \lambda_1(q) - \lambda_2(q)$. These are seven additions and five essential table lookups. In this way, we determine the barycentric coordinates of q avoiding costly rotation or transformation operations, without branching over 24 cases, and without performing any multiplication (not even by -1).

5 Evaluation of Polynomial Pieces and its Gradients

Once the location of a point q in a tetrahedron $T = [v_0, v_1, v_2, v_3] \in \Delta$ and its barycentric coordinates $\lambda_\nu(q),\ \nu = 0,\ldots,3$, have been determined, the value of the spline s from Section 3 at q and the gradient of its polynomial pieces can be computed. Obviously, this information is needed to properly visualize s (see Section 6). This is done for our model by applying (the trivariate version) of well established algorithms from CAGD.

For the trivariate polynomial $p = s|_T \in \mathscr{P}_2$ in the form (1) the *de Casteljau algorithm* (cf. [9], see also [16]) to determine the value $p(q) = a_{0,0,0,0}^{[2]}$ reads as follows:

***de Casteljau Algorithm**: For $r = 1,2$, compute*

$$a_{i,j,k,l}^{[r]} = \lambda_0(q)\ a_{i+1,j,k,l}^{[r-1]} + \lambda_1(q)\ a_{i,j+1,k,l}^{[r-1]} + \lambda_2(q)\ a_{i,j,k+1,l}^{[r-1]} + \lambda_3(q)\ a_{i,j,k,l+1}^{[r-1]}, \qquad i+j+k+l = 2-r, \tag{8}$$

where $a_{i,j,k,l}^{[0]} = a_{i,j,k,l},\ i+j+k+l = 2.$

In general, this algorithm needs a total number of 20 multiplications and 15 additions to determine the value of p at q. If one or even two of the barycentric coordinates of q vanish, then the algorithm degenerates to its bivariate and univariate version, respectively. In these cases, q lies in the interior of a triangular face of T or on an edge of T, and the number of necessary arithmetic operations reduces to 12 multiplications and 8 additions, and 6 multiplications and 3 additions, respectively.

For the proper shading of surfaces obtained from the volume model (see Section 6) at the point q, i.e. it is necessary to compute the gradient

$$(\nabla p)(q) = ((\tfrac{\partial p}{\partial x})(q), (\tfrac{\partial p}{\partial y})(q), (\tfrac{\partial p}{\partial z})(q))^\top. \tag{9}$$

If we let ς_ν be a vector in direction of the edge $v_{\nu+1} - v_0$ of T with length $\|v_{\nu+1} - v_0\|$ ($\|.\|$ is the Euclidian distance), then the partial derivative of p in direction of ς_ν (cf. [16]) denoted by $\frac{\partial p}{\partial \varsigma_\nu} \in \mathscr{P}_1$ is given as

$$\frac{\partial p}{\partial \varsigma_\nu} = 2 \sum_{i+j+k+l=1} (a_{i,j+j_\nu,k+k_\nu,l+l_\nu} - a_{i+1,j,k,l})\ \lambda_0^i \lambda_1^j \lambda_2^k \lambda_3^l,$$

where $(j_\nu, k_\nu, l_\nu) = (\delta_{\nu,\mu})_{\mu=0}^2,\ \ \nu = 0,1,2.$ Hence, unique $\alpha_0,\ \alpha_1, \alpha_2 \in \mathbb{R}$ exist such that, for instance,

$$\frac{\partial p}{\partial x} = \alpha_0\ \frac{\partial p}{\partial \varsigma_0} + \alpha_1\ \frac{\partial p}{\partial \varsigma_1} + \alpha_2\ \frac{\partial p}{\partial \varsigma_2}.$$

This shows that no more than a total number of 21 multiplications and 21 additions are required to compute the gradient in (9) for a given point q. Again, we use a lookup table for the precomputed numbers α_μ for the different tetrahedra. Since each tetrahedron of Δ has two edges which are axis parallel, the total number of arithmetic operations required for the gradient is a bit smaller than 42.

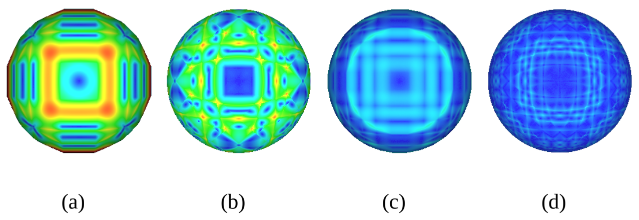

Figure 7: Isosurfaces showing the error of the respective model's gradient for the trilinear (a),(c) and for our quadratic (b),(d) reconstruction of 8^3 (a),(b) and 16^3 (c),(d) samples of a sphere $f(x,y,z) = \|(x,y,z)^\top\|$ $(x,y,z \in [-\frac{1}{2}, \frac{1}{2}])$. The angular deviation from the perfect gradient ∇f is color coded (from red=1° to blue=0°) on the isosurface $f(x,y,z) = 0.4$. As expected, the underlying grid structure imposes visible artifacts for this extreme diagram.

6 Visualization: Isosurface Rendering by Precise Ray-Casting

A visualization technique for volume data frequently used in computer graphics is to *render isosurfaces* from a given reconstruction model. *Ray-casting* is an image-space technique to compute particular views of these surfaces. Other methods such as the marching cubes algorithm are described in [5, 6, 21], for instance. Ray-casting considers the model along arbitrary rays r,

$$r = r(t): \quad t \mapsto q_0 + t\, r_0, \qquad t \geq 0, \tag{10}$$

where the goal is to find the smallest (intersection) parameter $t^* \geq 0$, such that the model along r coincides with a prescribed *isovalue*. Here, $q_0 \in \mathbb{R}^3$ is the position of the viewer and $r_0 \in \mathbb{R}^3$ is the (normalized) viewing direction determined as the difference of the current pixel position in the projection plane and q_0. Therefore $q^* = r(t^*)$ is the point closest to the viewer position, where the model intersects the *isosurface*. A standard *ray-casting algorithm* generates rays through all pixel positions, examines the model along each ray in order to find the closest intersection point q^* with the isosurface, and (if q^* exists) finally evaluates the gradient for proper shading of the isosurface at the current pixel position.

In order to show the potential of our method for the efficient visualization of volume data, we apply ray-casting on the reconstruction model $s \in \mathscr{S}_2(\Delta)$ from Section 3. In the following, we focus on the specific advantages of our model in contrast to other reconstructions, namely the efficient and exact computation of the intersection point q^* of s along r, and the effective determination of exact gradient information at q^*. Since the approximation s along r is a quadratic, univariate spline, and by the choice of the underlying space $\mathscr{S}_2(\Delta)$, it follows that these computations can be made by solving a very simple equation and applying the tools described in Sections 4 and 5.

Let an arbitrary ray r as in (10) be given, and let us assume that $Q \in \diamondsuit$ lies within the current region of interest when casting r through Ω. This means that r intersects Q at two points which are immediately available. In the following, we call these points enter and exit point of Q, respectively. We must then process all the tetrahedra in Q which intersect r. A naive approach would be to intersect r with the six cutting planes from (7) and to obtain a sequence of all intersection points with the tetrahedra in Q by sorting the corresponding, non-negative ray parameters. In order to avoid unnecessary computations, we first determine a tetrahedron T_0 in Q from the enter point of Q as described in Section 4. The intersected face of T_0 is axis aligned. In this case, the second intersection point of r with T_0 lies in an other non-axis aligned face of T_0. The three candidate faces lie in one of the six cutting planes from (7). If needed, we analogously determine another tetrahedron in Q containing the second intersection point from T_0, and proceed similarly. We eventually iterate until r meets the tetrahedron which contains the exit point of Q. As Q is sliced by six planes, previously computed results can be reused here, and we calculate at most six intersection parameters at a cost of two additions and one division each.

Given r as in (10) and a prescribed isovalue which we may assume to be zero, for the current tetrahedron $T \in \Delta$, we have to determine the closest point $q^* \in T$ to the viewer, where the trivariate polynomial piece $p = s|_T \in \mathscr{P}_2$ vanishes along r, and we have to find out quickly when such a point q^* does not exist in T. Let $q_1 = r(t_1)$ and $q_2 = r(t_2)$, where $t_1 < t_2$, be the two intersection points of r with T. Then, the restriction of p to the line segment $[q_1, q_2]$ is a quadratic, univariate polynomial (see Fig. 3). It is therefore obvious that we only have to consider a quadratic equation, whose roots can be found in an analytic and exact way with considerably small computational effort. For setting up the necessary equation, we first compute the values w_1, w, and w_2 of p at the three points q_1, $q = \frac{q_1+q_2}{2}$, and q_2 in T, i.e. $w_1 = p(q_1)$, $w = p(q)$, and $w_2 = p(q_2)$. This is done by applying de Casteljau's algorithm from Section 5. We quickly access the 10 coefficients of p via an index table into the 65 coefficients for the whole cube. Since the points q_1 and q_2 both lie within a triangular face of T, we first perform the bivariate version of the de Casteljau algorithm twice. The third run of the algorithm is done for the point q. This is the only run which is of trivariate type, in general. We use some previously computed results such that the total number of required operations reduces to 15 multiplications and 13 additions. Note that except for the triangle T_0 (containing the enter point of a cube Q) the above bivariate version of de Casteljau's algorithm has to be performed only once per tetrahedron, since the second intersection point q_2 of T becomes a point of type q_1, when we pass to the adjacent tetrahedron. The intersection point q^* is now determined as follows. Using a precomputation of Newton's interpolation form, we find the unique quadratic polynomial on an appropriate interval $[0, \delta]$, which interpolates the three values w_1, w, and w_2 at the points 0, $\frac{\delta}{2}$, and δ. From this, we obtain the quadratic equation

$$\alpha\,\tau^2 + \delta\,\beta\,\tau + \delta^2\,\gamma = 0, \qquad \tau \in [0, \delta], \tag{11}$$

where $\alpha = 2\,(w_1 + w_2 - 2\,w)$, $\beta = 4\,w - 3\,w_1 - w_2$, and $\gamma = w_1$. Hence, once w_1, w, and w_2 are determined, the equation (11) is set up by using 10 additions. If (11) degenerates to a linear equation, i.e. $\alpha = 0$, we obtain $t^* = t_1 + \frac{\delta}{2}\,(\frac{w_1}{w_1 - w})\,(t_2 - t_1)$. Otherwise, we get

$$t^* = t_1 + \frac{\delta}{2\alpha}\,\left(-\beta \pm \sqrt{\beta^2 - 4\,\alpha\gamma}\right)\,(t_2 - t_1), \tag{12}$$

and we choose the (smaller) solution in $[t_1, t_2]$ to fix q^*, if it exists. The latter is not the case if $\beta^2 - 4\,\alpha\gamma < 0$, or, otherwise, if the solution(s) from the above equations do not lie in $[t_1, t_2]$. The necessary arithmetic operations are at most 5 multiplications, 6 additions and one square root evaluation.

Still, in the worst case, all the tetrahedra $T \subseteq Q$ along r have to be processed in order to check if $s|_Q$ is *not* intersected by the isosurface. We can easily accelerate this process by applying a quick, conservative test on whether s restricted to r cannot intersect the isosurface locally in a tetrahedron or cube. If $p = s|_T$ is given in the form (1), we check $\sigma\,a_{i,j,k,l} > 0$, $i + j + k + l = 2$, where $\sigma \in \{-1, 1\}$. If this sign criterion is satisfied, then we do not have to consider T and can skip it because of the well-known *convex hull property* of the Bernstein-Bézier form. A similar test can be applied to all the 65 coefficients of $s|_Q$, where the minimum and maximum coefficients can be precomputed and stored for each cube, e.g. in a min-max-octree for optimized ray-casting with eventually varying isovalues.

Once an intersection point $q^* = r(t^*)$ has been found, we determine the gradient $(\nabla p)(q^*)$ as defined in (9) following Section 5. A well-known result from differential geometry shows that the normal vector n^* at q^* is given by $n^* = (\nabla p)(q^*)/\|(\nabla p)(q^*)\|$. The normal n^* is required for shading computations, e.g. using the standard Phong illumination model. The results given in the next section show that the isosurfaces are visually smooth due to the high quality normals obtained from the local gradients.

7 Results

We applied our new reconstruction by quadratic super splines to a number of well-known volume data sets. The figures show the visualization of isosurfaces using classical perspective ray-tracing as previously outlined. All local calculations such as evaluation and intersection are performed efficiently. However, our overall ray-casting algorithm is not yet tuned for speed and not competitive to more sophisticated systems like e.g. [2, 32] which may even aim towards interactive frame rates (see [38], for a recent survey). As there are numerous optimizations of the general ray-casting algorithm, a discussion is beyond the scope of this paper. Any optimization can be combined with our model in a straightforward way with a direct benefit in ray-casting performance. In particular, this includes hierarchical space partitioning or efficient cube traversal by an object-order ray-casting algorithm as applied for triquadratic tensor spline models (cf. [2, 27]).

We simply preprocess the data for every single isovalue, constructing all cubes and storing relevant ones which potentially intersect the isosurface (typically only some few percent for our experiments). This allows us to provide timings for the construction of a single cube and to estimate a faithful lower bound for more sophisticated preprocessing as the generation of a min-max-octree. All runtimes are measured on a 2.8GHz Intel Xeon CPU, where we observe $0.27\mu s$ for the construction of the spline on a single cube (Section 3) plus an average of $0.13\mu s$ for the convex hull tests to determine the relevance of a cube (Section. 6). We report per frame timings (average $38.7\mu s$ per ray) for quadratic reconstruction as well as the isovalues and the percentages of relevant (and precomputed) cubes for the Figures 8, 9 and 10 which are rendered into a 512x512 viewport. For all respective figures, we computed higher quality, non-local gradients on the trilinear model (see below) to ensure a fair visual comparison. The difference between the models becomes most visible for high frequency areas (e.g. *bonsai*'s leaves, arteries) with a feature size of only few samples. Fig. 1 and 6 show a synthetic benchmark, and Fig. 7 emphasizes the quality of the gradients.

Regarding the number of floating point operations, our quadratic approach is close to the simple trilinear interpolation and much cheaper than a triquadratic model. The same is true for the computation of the gradients. However, as the trilinear model does not satisfy smoothness conditions, local gradient evaluation is inexact for general data, while the costs for better gradients using central differences by evaluation in six neighboring cells is more expensive. The price for our approach is a slight overhead for finding the point location in a tetrahedron and the need to store 65 coefficients instead of 27 (triquadratic) or working directly on the data (trilinear). The evaluation of roots along a ray is exact and inexpensive for quadratic polynomials, non-trivial for cubics [35] (trilinear) and analytically impossible for degree six polynomials (triquadratic), i.e. a numerical root finding algorithm must be applied. In addition, the univariate, quadratic polynomials allow efficient integration by applying quadrature formulae and evaluation of the extreme values along a ray. The necessary computations can be performed in a straightforward way by following the method from Section 6.

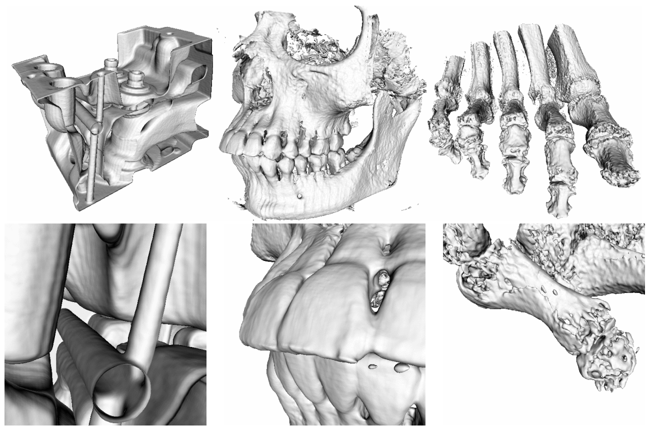

Figure 10: More isosurfaces rendered from our quadratic model. *engine* courtesy of General Electric ($141 \times 198 \times 110$ samples, isovalue $c = 80$, 2.5s (full) and 1.95s (close-up), 9.7% relevant cubes). *skull* courtesy of Siemens Medical System, Forchheim, Germany (256^3, $c = 40$, 4.3s and 5.41s, 5.2%). *foot* courtesy of Philips Research, Hamburg, Germany (256^3, $c = 90$, 8.7s and 6.8s, 2.37%).

8 Conclusions and Future Work

We presented a new model for the reconstruction of discrete volume data given on a regular grid which is a typical problem in volume rendering. In contrast to earlier approaches, our method approximates the data by quadratic, trivariate super splines on a tetrahedral partition. The reconstruction is natural, completely symmetric, and efficient. The local spline model can be evaluated efficiently including precise, local gradients due to appropriate smoothness properties. The new approach uses piecewise polynomials of total polynomial degree two and it compares to existing trilinear and triquadratic approaches based on piecewise polynomials of total degree three and six, respectively. We exploit this fact for efficient and precise isosurface ray-casting. Our results show that the model is effective, efficient, simple in implementation, and appropriate for high-quality volume rendering.

In future work, we will study the much more difficult problem of efficient reconstruction from *scattered data*. As we shift the focus from pure reconstruction to approximation issues, the use of lower dimensional spaces providing automatic data compression becomes an interesting topic of future research.

Acknowledgements

The authors would like to thank Joel Carranza for proofreading the manuscript.

References

[1] C.L. Bajaj. *Data Visualization Techniques*. John Wiley & Sons, 1999.

[2] L. Barthe, B. Mora, N. Dodgson, and M.A. Sabin. Triquadratic reconstruction for interactive modelling of potential fields. In *Proc. Shape Modeling International 2002*, pages 145–153, 2002.

[3] G.-P. Bonneau, S. Hahmann, and G. Nielson. BLaC-Wavelets: A multiresolution analysis with non-nested spaces. In *Proc. IEEE Visualization 1996*, pages 43–48, 1996.

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

[5] K. Brodlie and J. Wood. Recent Advances in Volume Visualization. *Computer Graphics Forum*, 20(2):125–148, 2001.

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

[7] Charles K. Chui. *Multivariate Splines*. CBMS 54, SIAM, 1988.

[8] O. Davydov and F. Zeilfelder. Scattered data fitting by direct extension of local polynomials with bivariate splines. *Adv. Comp. Math.*, to appear, 2003.

[9] P. de Casteljau. Courbes et surfaces à poles. *André Citroën, Automobiles SA, Paris*, 1963.

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

[11] T. Gerstner and M. Rumpf. Multiresolutional Parallel Isosurface Extraction based on Tetrahedral Bisection. In *Proc. VolVis 1999*, pages 1–11, 1999.

[12] R. Grosso, C. Lürig, and T. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In *Proc. IEEE Visualization 1997*, pages 387–394, 1997.

[13] J. Haber, F. Zeilfelder, O. Davydov, and H.-P. Seidel. Smooth approximation and rendering of large scattered data sets. In *Proc. IEEE Visualization 2001*, pages 341–347, 571, 2001.

[14] T. Hangelbroek, G. Nürnberger, C. Rössl, H.-P. Seidel, and F. Zeilfelder. On the dimension of C^1 splines of arbitrary degree on a tetrahedral partition. *submitted, preprint available as tech.rep. from http://www.mpi-sb.mpg.de*, 2003.

[15] 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.

[16] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, 1993.

[17] A.E. Kaufman. State-of-the-art in volume graphics. In M.C., A.E. Kaufman, and R.Yagel, editors, *Volume Graphics*, pages 3–28. Springer, 2000.

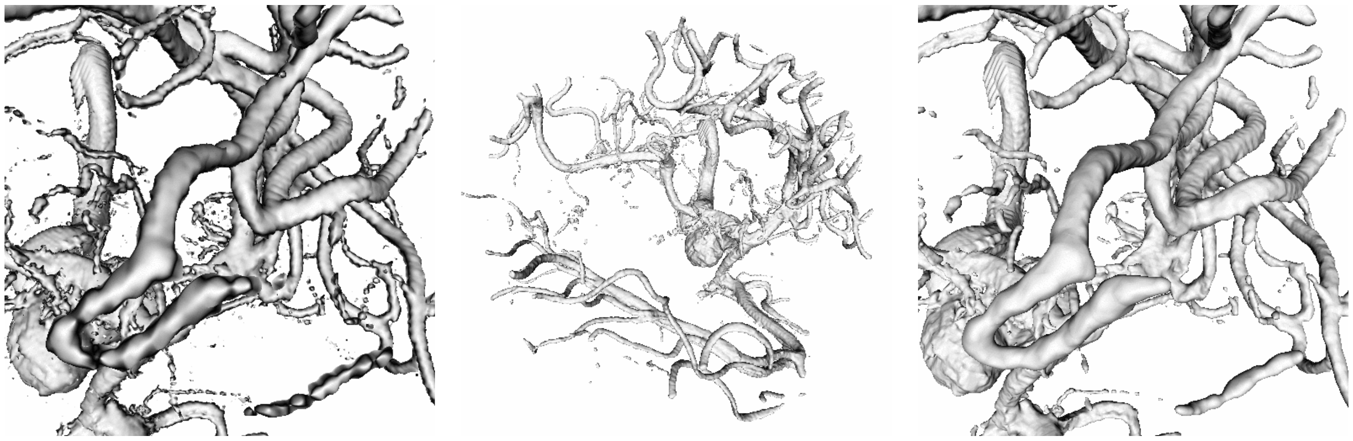

Figure 8: Isosurface of the *aneurism* data set (courtesy of Philips Research, Hamburg, Germany; 256^3 samples, isovalue 50). Trilinear (left) and our quadratic (center and right) reconstruction (12.7*s* and 11.7*s*, 0.66% relevant cubes).

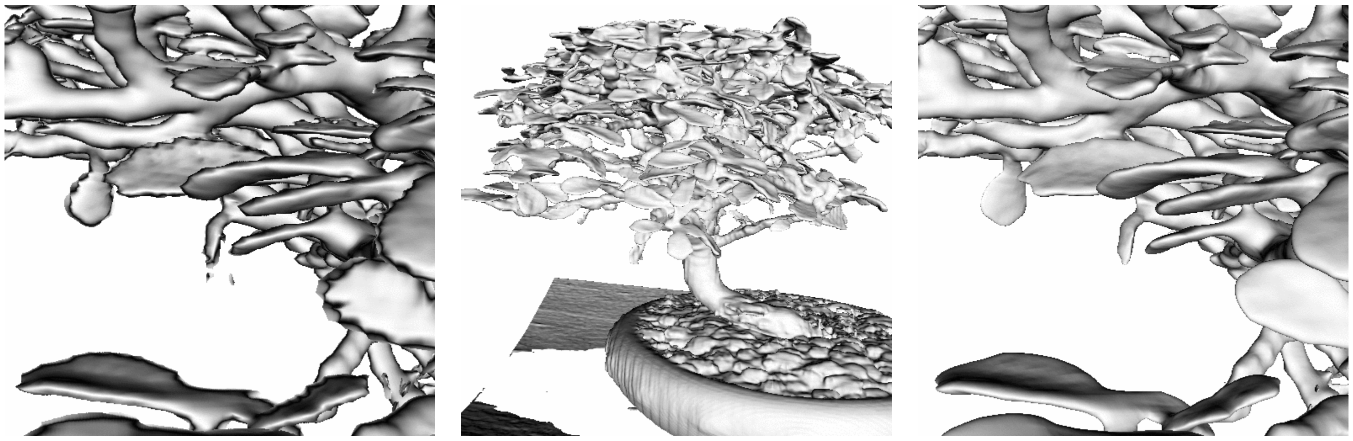

Figure 9: Isosurface of the *bonsai* data set (courtesy of Stefan Röttger, VIS, University of Stuttgart, Germany; 256^3 samples, isovalue 40). Trilinear (left) and our quadratic (center and right) reconstruction (7.1*s* and 9.1*s*, 3.1% relevant cubes).

[18] 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*. Vanderbilt University Press Nashville, to appear.

[19] M.-J. Lai and A. Le Méhauté. A new kind of trivariate C^1 spline. *(preprint)*, 2003.

[20] E. LaMar, B. Hamann, and K.I. Joy. High-quality rendering of smooth isosurfaces. *Journal of Visualization and Computer Animation*, 10:79–90, 1999.

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

[22] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proc. IEEE Visualization 1994*, pages 100–107, 1994.

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

[24] M. Meissner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical comparison of popular volume rendering algorithms. In *Symposium on Volume Visualization and Graphics 2000*, pages 81–90, 2000.

[25] D.P. Mitchell and A.N. Netravali. Reconstruction filters in computer graphics. *Proc. SIGGRAPH 88*, pages 221–228, 1988.

[26] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proc. Symposium on Volume Visualization 1998*, pages 143–151, 1998.

[27] B. Mora, J.-P. Jessel, and R. Caubet. Visualization of isosurfaces with parametric cubes. In *Proc. Eurographics 2001*, pages 377–384, 2001.

[28] G.M. Nielson. Volume modelling. In M. Chen, A.E. Kaufman, and R.Yagel, editors, *Volume Graphics*, pages 29–50. Springer, 2000.

[29] G. Nürnberger, L.L. Schumaker, and F. Zeilfelder. Lagrange interpolation by C^1 cubic splines on triangulated quadrangulations. *Adv. Comp. Math.*, to appear, 2003.

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

[31] Günther Nürnberger and Frank Zeilfelder. Lagrange interpolation by bivariate C^1-splines with optimal approximation order. *Adv. Comp. Math.*, to appear, 2003.

[32] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Proc. IEEE Visualization 1998*, pages 233–238, 1998.

[33] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[34] L.L. Schumaker and T. Sorokina. Quintic spline interpolation on type-4 tetrahedral partitions. *(preprint) to appear in Adv. Comput. Math.*, 2003.

[35] Jochen Schwarze. Cubic and quartic roots. In Anrew Glassner, editor, *Graphics Gems*, pages 404–407. Academic Press, 1990.

[36] T. Theußl, T. Möller, J. Hladuvka, and M. Gröller. Reconstruction issues in volume visualization. In *Data Visualization: State of the Art, Proc. of IEEE Visualization 2002*, 2002.

[37] P. Thévenaz and M. Unser. High-quality isosurface rendering with exact gradients. In *Proc. IEEE Visualization 2001*, pages 854–857, 2001.

[38] I. Wald and P. Slusallek. State of the art in interactive ray tracing. In *STAR, EUROGRAPHICS 2001*, pages 21–42. 2001.

MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-002	C. Theobalt, M. Li, M. Magnor, H. Seidel	A Flexible and Versatile Studio for Synchronized Multi-view Video Recording
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects
MPI-I-2003-4–003	I. Ivrissimtzis, W. Jeong, H. Seidel	Neural Meshes: Statistical Learning Methods in Surface Reconstruction
MPI-I-2003-2-002	M. Jaeger	A Representation Theorem and Applications to Measure Selection and Noninformative Priors
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-011	P. Krysta, A. Czumaj, B. Voecking	Selfish Traffic Allocation for Server Farms
MPI-I-2003-1-010	H. Tamaki	A linear time heuristic for the branch-decomposition of planar graphs
MPI-I-2003-1-009	B. Csaba	On the Bollobás – Eldridge conjecture for bipartite graphs
MPI-I-2003-1-008	P. Sanders	Soon to be published
MPI-I-2003-1-007	H. Tamaki	Alternating cycles contribution: a strategy of tour-merging for the traveling salesman problem
MPI-I-2003-1-006	H. Tamaki, M. Dietzfelbinger	On the probability of Rendezvous in Graph
MPI-I-2003-1-005	M. Dietzfelbinger, P. Woelfel	Almost Random Graphs with Simple Hash Functions
MPI-I-2003-1-004	E. Althaus, T. Polzin, S.V. Daneshmand	Improving Linear Programming Approaches for the Steiner Tree Problem
MPI-I-2003-1-003	R. Beier, B. Vcking	Random Knapsack in Expected Polynomial Time
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting
MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi

MPI-I-2002-1-008	P. Sanders, J.L. Trff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	M. Hoefer	Performance of heuristic and approximation algorithms for the uncapacitated facility location problem
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Trff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property
MPI-I-2002-1-002	F. Grandoni	Incrementally maintaining the number of l-cliques
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials
MPI-I-2001-2-006	H. Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators
MPI-I-2001-2-004	H. de Nivelle	Translation of Resolution Proofs into Higher Order Natural Deduction using Type Theory
MPI-I-2001-2-003	S. Vorobyov	Experiments with Iterative Improvement Algorithms on Completely Unimodel Hypercubes
MPI-I-2001-2-002	P. Maier	A Set-Theoretic Framework for Assume-Guarantee Reasoning
MPI-I-2001-2-001	U. Waldmann	Superposition and Chaining for Totally Ordered Divisible Abelian Groups
MPI-I-2001-1-007	T. Polzin, S. Vahdati	Extending Reduction Techniques for the Steiner Tree Problem: A Combination of Alternative-and Bound-Based Approaches
MPI-I-2001-1-006	T. Polzin, S. Vahdati	Partitioning Techniques for the Steiner Problem
MPI-I-2001-1-005	T. Polzin, S. Vahdati	On Steiner Trees and Minimum Spanning Trees in Hypergraphs
MPI-I-2001-1-004	S. Hert, M. Hoffmann, L. Kettner, S. Pion, M. Seel	An Adaptable and Extensible Geometry Kernel
MPI-I-2001-1-003	M. Seel	Implementation of Planar Nef Polyhedra
MPI-I-2001-1-002	U. Meyer	Directed Single-Source Shortest-Paths in Linear Average-Case Time
MPI-I-2001-1-001	P. Krysta	Approximating Minimum Size 1,2-Connected Networks
MPI-I-2000-4-003	S.W. Choi, H. Seidel	Hyperbolic Hausdorff Distance for Medial Axis Transform
MPI-I-2000-4-002	L.P. Kobbelt, S. Bischoff, K. Khler, R. Schneider, M. Botsch, C. Rssl, J. Vorsatz	Geometric Modeling Based on Polygonal Meshes
MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-2-001	F. Eisenbrand	Short Vectors of Planar Lattices Via Continued Fractions
MPI-I-2000-1-005	M. Seel, K. Mehlhorn	Infimaximal Frames: A Technique for Making Lines Look Like Segments
MPI-I-2000-1-004	K. Mehlhorn, S. Schirra	Generalized and improved constructive separation bound for real algebraic expressions