SMURF – a SMart surFace model for advanced visualization techniques

**Helwig Löffelmann, Thomas Theußl,
Andreas König, and Eduard Gröller**

Institute of Computer Graphics, Vienna University of Technology,
Karlsplatz 13/186/2, A-1040 Wien, Austria.
`http://www.cg.tuwien.ac.at/home/`
`mailto:{helwig|theussl|koenig|groeller}@cg.tuwien.ac.at`

ABSTRACT

Highly elaborated visualization techniques that are based on surfaces often are independent from the origin of the surface data. For re-using advanced visualization methods for surfaces of various kind, we developed an abstract surface interrogation layer called SMURF. In this paper we discuss the steps necessary to unify multiple types of surfaces under a shared general purpose interface.

Keywords: visualization, surfaces

1 INTRODUCTION

Surfaces are important geometric primitives for scientific visualization [13]. Useful techniques are available to render surfaces of various kind. For instance, scalar data volumes from medical applications are represented using iso-surfaces [11]. Three-dimensional vector fields from flow analysis are visualized by the use of stream surfaces [8].

Interrante et al., for example, showed how curvature-based techniques enhance the use of surfaces for the visualization of volumetric data [9, 10]. Surface curvature also plays an important role in surface design, surface fairing, surface trimming, surface evaluation / analysis, and surface visualization [2, 4, 5, 18].

In this paper we present a software system, which simplifies the development of advanced visualization techniques, that are based on surfaces, by introducing a generic surface interrogation layer called SMURF (see Fig. 1). Independently from the type of surface (see Sect. 2) the most important surface interrogation queries (see Sect. 3) are available. We discuss the challenges of unifying various surface types for specifying a common interface,

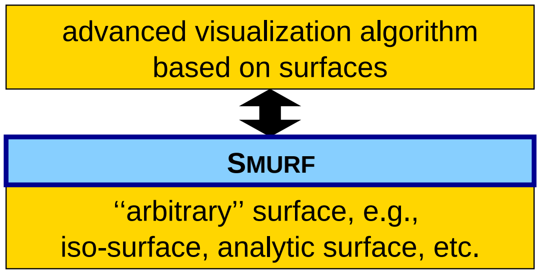

Figure 1: SMURF is a C++ interface between visualization algorithms and surface implementation.

present the class hierarchy (see Sect. 4), describe possible applications, and give some preliminary results together with a piece of sample code (see Sect. 5).

The advantage of specifying a generic interface like SMURF is that visualization techniques are easily ported from one application to another. Algorithms like modulating the opacity of the surface according to its curvature properties are not bound to one application, but can be reused for other surfaces as well. A similar approach in the area of mesh access is described by Rumpf et al. [15].

2 SURFACE TYPES

In the following we describe some of the most important surface types in computer graphics and visualization. Surfaces can be given implicitly, for example, as an iso-surface of scalar volume data, or explicitly, i.e., analytically. Using SMURF the following surface types can be dealt with:

CASE 1: **Implicitly defined iso-surfaces for discrete scalar volume data** – scalar data values are given at certain discrete locations, e.g., on a regular grid, or as scattered data. A certain interpolant $f(\mathbf{x})$ of these values is considered to implicitly define an iso-surface $\mathbf{s}$ corresponding to a certain iso-value $f_{\mathbf{s}}$.

CASE 2: **Implicitly defined iso-surfaces for analytic scalar volume functions** – a scalar function $f(\mathbf{x})$ is given (as a "black box") to be evaluated at arbitrary locations in 3D. A scalar continuum over 3D is assumed as the application of the function to "all" points. A certain iso-value $f_{\mathbf{s}}$ specifies the iso-surface $\mathbf{s}$.

CASE 3: **Implicitly defined stream surfaces for discrete vector fields** – vectorial data $\mathbf{v}(\mathbf{x}_i)$ is given at certain discrete locations $\mathbf{x}_i$, for example, on a curvilinear grid. For a specific initial set $\mathbf{s}(u,0)$ the corresponding stream surface $\mathbf{s}(u,t)$ is implicitly defined as the integral set $\mathbf{s}(u,0)+\int_0^t \mathbf{v}(\mathbf{s}(u,\tau))\,\mathrm{d}\tau$ of a certain interpolant $\mathbf{v}(\mathbf{x})$ of the discrete values $\mathbf{v}(\mathbf{x}_i)$.

CASE 4: **Implicitly defined stream surfaces for analytically specified dynamical systems** – a vectorial function $\mathbf{v}(\mathbf{x})$ is given (as a "black box") to be evaluated at arbitrary locations in 3D. A vectorial continuum over 3D is assumed as the application of the function to "all" points. A certain initial set $\mathbf{s}(u,0)$ is implicitly integrated to define the stream surface $\mathbf{s}(u,t)=\mathbf{s}(u,0)+\int_0^t \mathbf{v}(\mathbf{s}(u,\tau))\,\mathrm{d}\tau$.

CASE 5: **Explicitly defined parametric surfaces** – such a surface is defined by a function $\mathbf{s}:\mathbf{R}^2\rightarrow\mathbf{R}^3$.

CASE 6: **Explicitly defined surfaces given by implicit form** – an equation $f(\mathbf{x})=0$ defines a surface in 3D (note that this case is similar to case 2).

CASE 7: **Explicitly defined discrete surface approximations** – a set of polygons or a mesh is used to explicitly specify an approximation of a smooth surface.

There are other surface types as well, for example, explicitly expressing one coordinate in terms of the others ($z(x,y)$). Usually they can be either transformed into one of the above mentioned cases, or appear rather rarely. Therefore, they are not considered separately.

3 SURFACE INTERROGATION

Elaborated methods usually are based on surface properties up to the order of two, i.e., the calculation of surface-point locations, surface normals, and surface curvature properties. Surface interrogation, i.e., the evaluation of these properties for certain points of the surface, involves a number of algorithms [7].

3.1 Surface-point location

Algorithms used for visualization of volumetric data can be separated into image-based and object-based methods. Image-based techniques usually are based on *ray casting*, whereas object-based techniques merely use projection for rendering. In this case often incremental *surface curve traversal* is used.

Ray casting

The intersection of a certain ray (given by **eye** and **dir**) and the surface yields a sorted list of surface-points (hit list). Depending on the type of surface, the intersection calculation is done differently:

Analytic solution (cases 5, 6, and 7) – in the case of an explicitly specified surface, usually the intersection between the ray and the surface can be calculated analytically. In the case of a parametric surface (case 5) the following two equations must be solved in u and v:

$$\mathbf{s}(u,v)\cdot\mathbf{n}_1 = \mathbf{eye}\cdot\mathbf{n}_1$$
$$\mathbf{s}(u,v)\cdot\mathbf{n}_2 = \mathbf{eye}\cdot\mathbf{n}_2$$
$$\text{where } \mathbf{n}_i\cdot\mathbf{dir} = 0 \text{ and } \mathbf{n}_i\cdot\mathbf{n}_j = \delta_{ij}$$

Depending on the complexity of $\mathbf{s}$, solving the above equations usually is not possible in closed form. There are numerical techniques to compute the list of intersections in terms of u and v [6].

In the implicit case (case 6) the following equation has to be solved in terms of λ:

$$f(\mathbf{eye} + \lambda\,\mathbf{dir}) = 0$$

In the polygonal case (case 7) basically all polygons have to be intersected with the ray to evaluate the hit list (line/plane intersections). Of course, spatial coherence can be exploited using special data structures for storing the polygons to speed up the intersection process [1, 17].

Root finding (cases 1 and 2) – considering a ray being cast into the density volume, e.g., a continuum that interpolates discrete data values (case 1) or the application of f to the entire domain (case 2), implicitly yields a scalar density function at all points of the ray. This function can be sampled to investigate their intersections with the iso-value. An interpolant is used to reconstruct this function along the ray, for instance, linear approximation. Of course, a higher order interpolants can be assumed for intersection, also.

In case 1 (discrete data values) it would be even more accurate to directly project density interpolant f to the ray, i.e., consider $f(\mathbf{eye} + \lambda\,\mathbf{dir})$, and search its intersections with the iso-value in terms of λ. This is usually not easily done, since even in the case of trilinear interpolation the resulting function is of order three [16].

Stream surface intersection (cases 3 and 4) – the most demanding problem within the surface-point location task is stream surface intersection. There are two possible ways. In the first approach the stream surface is integrated on demand, i.e., approximated as a set of polyons or mesh – see case 7) and stored for further interrogation.

Another approach is possible as well: instead of explicitly generating the stream surface itself, a "back-stream surface" could be computed considering the ray as an initial condition. Any intersection of this "back-stream surface" and the original initial set directly corresponds to an intersection of the stream surface and the ray via a stream line (cf. Fig. 2). Depending on various characteristics of the given situation, one of these two approaches might be favourable.

Surface-curve traversal

In addition to ray casting, incremental surface-curve traversal was chosen as second surface-point access strategy within SMURF. Starting with some initial surface-point $\mathbf{p}$, a neighboring location with a specific distance *dist* is searched in a certain direction $\mathbf{dir}$. Surface-point $\mathbf{p}$, surface normal $\mathbf{n}$, and direction $\mathbf{dir}$ define a plane which intersects the surface in a

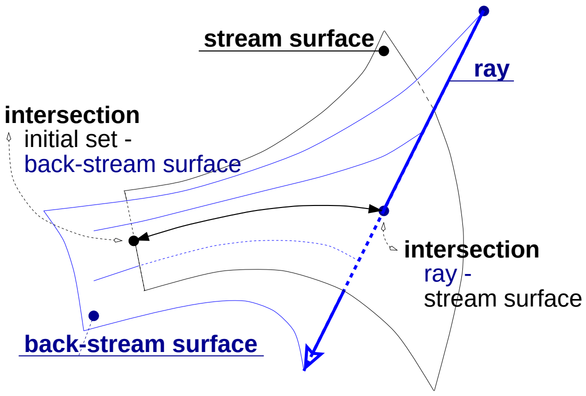

Figure 2: intersecting ray with implicitly specified stream surface.

certain surface-curve. Of both points on the curve which are *dist* away from $\mathbf{p}$ the one which is most aligned with **dir** is considered to be the searched location. See Fig. 3 for an illustration of this procedure.

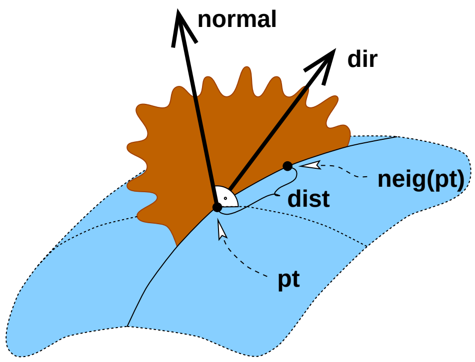

Figure 3: surface-curve traversal.

3.2 Surface normal

Various computer graphics algorithms use surface normals, e.g., for shading or back face culling. The aquisition of a normal corresponding to a certain surface-point again depends on the type of surface:

Analytic solution – if the surface is given explicitly, usually the surface normal at a certain point can be computed analytically. In the parametric case (case 5) the cross product $\partial\mathbf{s}/\partial u|_\mathbf{p} \times \partial\mathbf{s}/\partial v|_\mathbf{p}$ of two tangent vectors yields a (not yet normalized) surface normal at point $\mathbf{p}$. Of course, this is only possible if both tangents are not colinear. In the implicit case (case 6) the gradient $\nabla f|_\mathbf{p}$ is a surface normal of the iso-surface through $\mathbf{p}$ (not normalized).

Normal reconstruction from densities (cases 1 and 2) – assuming a function $f(\mathbf{x})$ which can be evaluated at arbitrary points $\mathbf{x}$ – f is either the "black box" of case 2 or an interpolant (case 1) – surface normals can be computed using central differences, for example:

$$\mathbf{n}(\mathbf{p}) = \nabla f|_\mathbf{p} \approx$$
$$\approx \frac{1}{2}\begin{pmatrix} f(\mathbf{x}+\mathbf{e}_1) - f(\mathbf{x}-\mathbf{e}_1) \\ f(\mathbf{x}+\mathbf{e}_2) - f(\mathbf{x}-\mathbf{e}_2) \\ f(\mathbf{x}+\mathbf{e}_3) - f(\mathbf{x}-\mathbf{e}_3) \end{pmatrix}$$
$$\mathbf{e}_\mathrm{i} = (\delta_{1\mathrm{i}}\ \delta_{2\mathrm{i}}\ \delta_{3\mathrm{i}})^\mathrm{T}$$

Higher order approximations of the gradient are possible as well. In general, an arbitrary complex derivative filter can be applied for normal reconstruction [3, 12].

Normal reconstruction from polygons (case 7) – a standard procedure for reconstructing normals within polygons is used for Phong shading [14]: in the vertices of a polygon a weighted sum of all normals of adjacent polygons is computed. These vertex normals are interpolated within the polygon to approximate the normals of the surface which is approximated by the polygons.

Stream surface normals (cases 3 and 4) – In the case of directly approximating the stream surface by a set of polygons, again techniques for case 7 can be used. In the other case (using the duality shown in Fig. 2) the surface normal can be composed of two tangent vectors. One is equal to the vectorial data value at the point of interest. The other can be approximated by investigating neighboring stream lines.

3.3 Surface curvature

Second order surface properties, i.e., curvature information, is used to enhance surface-based visualization. Shape and location of a surface can be better perceived, for example, if curvature directed strokes are applied to the surface [10].

Surface curvature can be expressed in several terms, e.g., Gauss or mean curvature. By definition [5], surface curvature is a property which is dependent on a specific tangent direction. Principal directions are those tangent directions which yield either maximum or minimum curvature. To compute the curvature properties of a surface, carefully chosen steps are necessary, especially, if reconstruction, i.e., interpolation, is used.

Depending on the various cases, curvature can be computed analytically (cases 5 and 6) [5], or by the use of second order derivative filters (cases 1 and 2). With stream surfaces (cases 3 and 4) higher order approximations must be used also. Case 7 (polygons) is usually difficult as a larger neighborhood should be considered for curvature approximation – note that polygons themselves do not have any curvature different from zero.

4 SMURF CLASSES

The different surface types (Sect. 2) and surface interrogation schemes (Sect. 3) were integrated in a C++ class hierarchy called SMURF (see Fig. 6). An abstract base class provides the interface, virtual functions are used to represent the different surface interrogation schemes within the various sub-classes.

To distinguish discrete scalar volume data-sets (case 1) from analytic scalar volume functions (case 2) a class `ScalVolData` hides the interface. Therefore, class `IsoSmurf` can treat these two cases the same way. The same holds for discrete vector fields (case 3) and analytically specified dynamical systems (case 4) via class `VectVolData`. This means, that all the seven cases of various surface types which were presented in Sect. 2 are mapped to four sub-classes of SMURF, i.e., ISOSMURF, STREAMSMURF, ANASMURF, and POLYSMURF. See Fig. 6 for the relations between these classes.

An important concept in the implementation is the one of lazy evaluation, i.e., computing not more than necessary. For example, ray casting can be terminated after finding the desired intersection point. Further surface properties are evaluated on demand, and stored for future use. Therefore, a class hierarchy `SmurfPoint` is introduced which mirrors the `Smurf` class hierarchy and serves as memory element for the specific surface types. Again, Fig. 6 illustrates the relations between these classes.

5 RESULTS, APPLICATIONS

By hiding the intrinsic differences between the surface types identified in Sect. 2 SMURF supports the user with the following tasks:

Implementation of advanced visualization techniques – SMURF eases this task by providing an interface for obtaining surface properties independently of

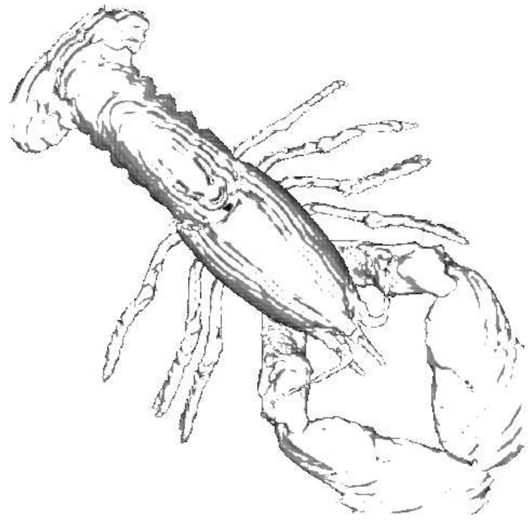

Figure 4: Contour display of lobster CT scan – see also Fig. 5.

```
Smurf *pSmurf = new IsoSmurf("lobster.dat",threshold);
for (p=pFirstPixel(); p!=NULL; p=pNextPixel())
{ VEC3 dir               = normalize(*p-eye);
  SmurfHitListHandle HLH = pSmurf->getHitListH(eye,dir);
  SmurfPointHandle   PH  = pSmurf->getPointH(HLH,0);
  VEC3 normal            = pSmurf->getNormal(PH);
  if (-dir*normal < 0.6) p->set(1-(-dir*normal));
  else                   p->set(0);                       }
```

Figure 5: code used for drawing lobster contours depicted in Fig. 4.

the surface type. Fig. 7 shows an iso-surface with crosses aligned to the principal directions (similar to Beck et al. [2]). Fig. 4 was generated using the code depicted in Fig. 5 – any other surface type could be rendered using the same code by just changing the very first line.

Comparision of algorithms – for instance, reconstruction schemes can be easily compared by sampling an analytic function and applying a visualization algorithm to both the scalar data volume and the analytic function. In Fig. 8 this concept was used to compare linear and cubic interpolation with respect to function reconstruction and computation of Gauss curvature with the corresponding analytic function. Linear reconstruction of densities is clearly seen in Fig. 8(a), whereas there is no perceivable difference between Fig. 8(b) and (c). Comparing the curvature plot, subtle differences can be obtained even between Fig. 8(b) and (c).

Fig. 9 compares the quality of curvature reconstruction (by calculating principal curvature lines of a cylinder) depending on linear and cubic density reconstruction. In Fig. 9(a) small errors accumulated during numerical integration of the curvature lines are clearly visible. For color plates, please refer to URL `http://www.cg.tuwien.ac.at/research/vis/misc/Smurf/`.

6 FUTURE WORK

One obvious disadvantage of such a general scheme is that it principally suffers from inefficiency. Nevertheless, performance can be improved by including, e.g., intelligent caching strategies and implementation short-cuts. Furthermore, it should be possible to exploit ray-to-ray coherence of visualization algorithms. Thus, again, caching and addressing of external data must be allowed by the scheme.

Another idea is to extend the SMURF concept to a 'set of SMURFs' class with a similar interface. Consecutive intersections along a ray are reported in correct order from different surfaces, e.g., stacked iso-surfaces or multiple stream surfaces. Even surfaces of different type could be easily combined using this concept, for example, patient data together with objects from (virtual) surgery planning.

7 CONCLUSIONS

Our general purpose surface interface SMURF allows to easily re-use highly elaborated visualization techniques that are based on surfaces in 3D, e.g., curvature-directed strokes or plotting curvature lines, with surface originating from various applications, e.g., iso-surfaces from medical applications or stream surfaces from flow visualization. Surface properties up to the order of two, i.e., curvature information, are available to the user in a transparent way. Ray casting as well

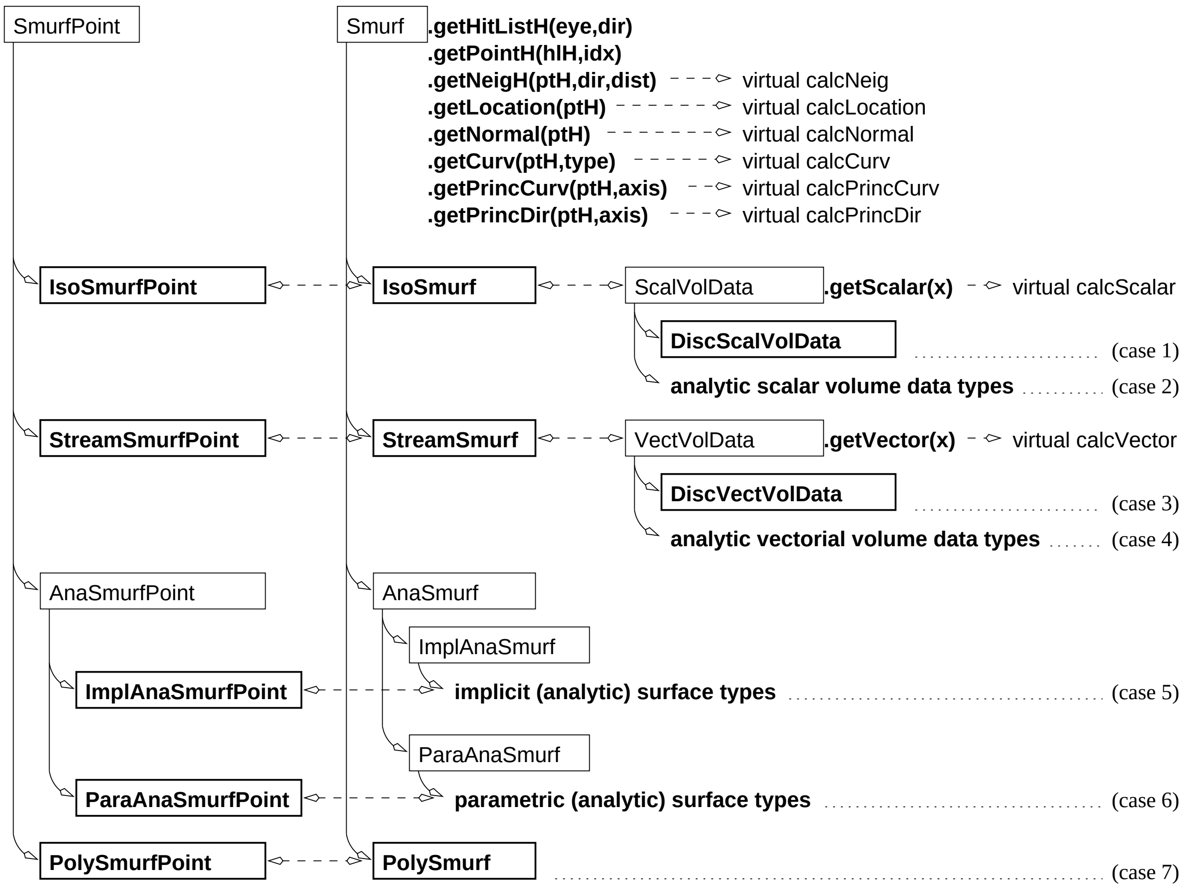

Figure 6: SMURF class hierarchy and interface.

as incremental surface-curve traversal are provided as surface access strategies.

We compared various surface interrogation algorithms necessary to unify multiple surface types under a unique interface. Analytic evaluation is opposed to reconstruction schemes from sampled data.

ACKNOWLEDGEMENTS

The work presented in this publication has been supported by V^{is}M^{ed}. Please refer to `http://www.vismed.at` for further information on this project.

REFERENCES

[1] J. Arvo and D. Kirk. A survey of ray tracing acceleration techniques. In A. Glassner, editor, *An introduction to ray tracing*, pages 201–262. Academic Press, 1989.

[2] J. Beck, R. Farouki, and J. Hinds. Surface analysis methods. *IEEE Computer Graphics and Applications*, 6(12):18–36, 1986.

[3] M. Bentum, B. Lichtenbelt, and T. Malzbender. Frequency Analysis of Gradient Estimators in Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, 1996.

[4] J. Dill. An application of color graphics to the display of surface curvature. *Proceedings SIGGRAPH*, 15(3):153–161, 1981.

[5] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 3rd edition, 1993.

[6] A. Glassner, editor. *An Introduction to Ray Tracing*. Academic Press, 1987.

[7] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Wordenweber, and

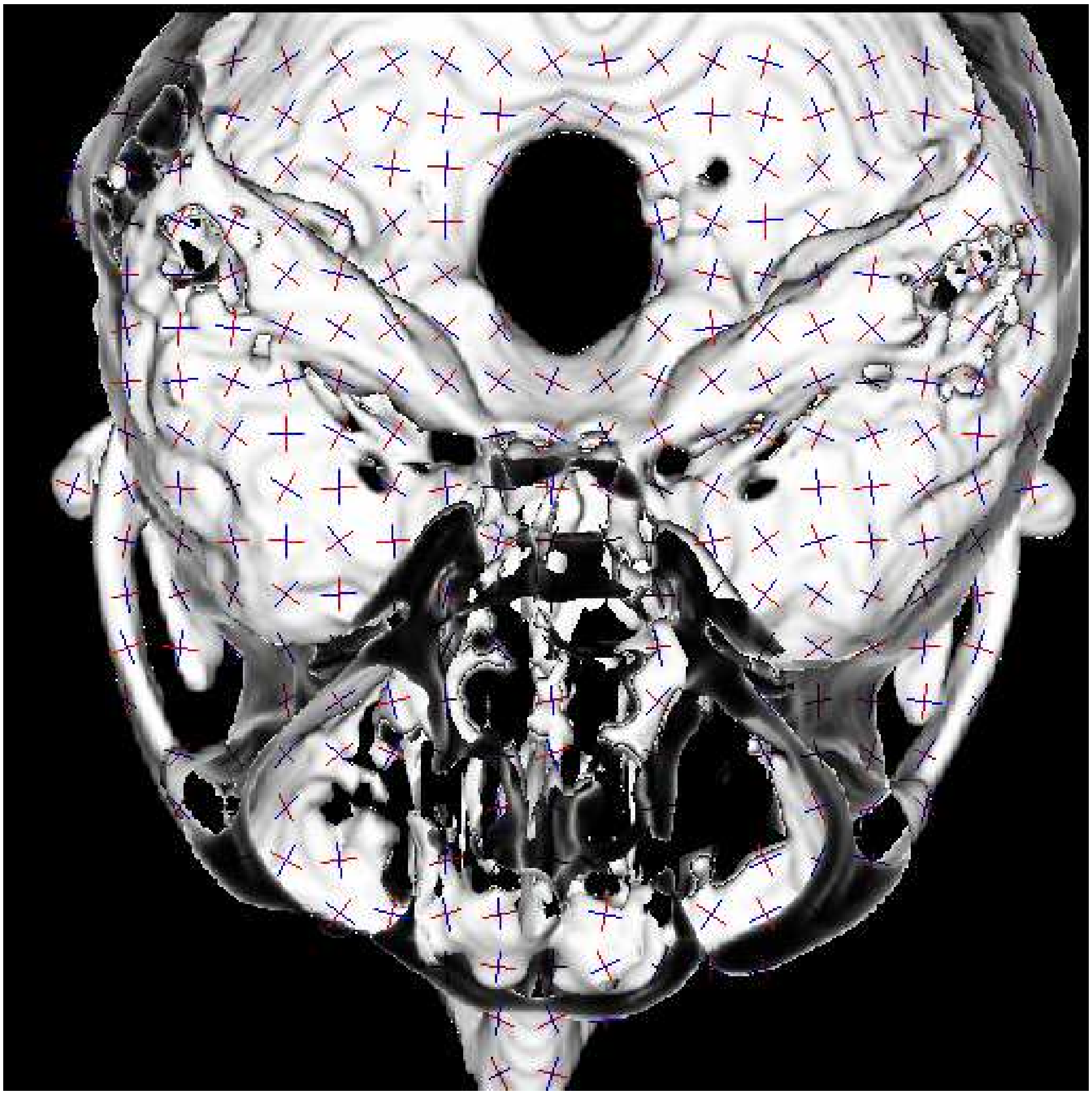

Figure 7: Iso-surface computed for ten slices scanned from a human head with curvature crosses.

P. Hollemann-Grundstedt. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, 1992.

[8] J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proceedings IEEE Visualization*, pages 171–177, 1992.

[9] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings SIGGRAPH*, pages 109–116, 1997.

[10] V. Interrante, H. Fuchs, and S. Pizer. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):98–117, 1997.

[11] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings SIGGRAPH*, 21(4):163–169, 1987.

[12] T. Möller, R. Machiraju, K. Müller, and R. Yagel. Evaluation and Design of Filters Using a Taylor Series Expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997.

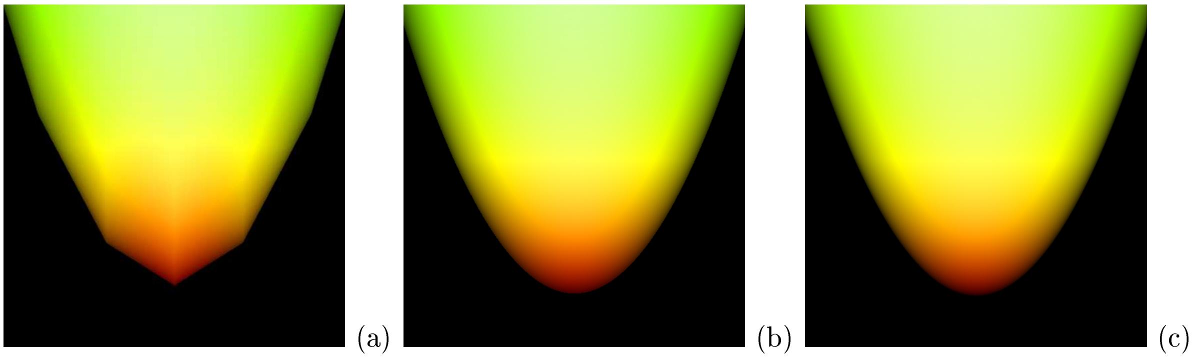

Figure 8: Gauss curvature plot using linear (a) and cubic (b) function reconstruction vs. analytic computation (c).

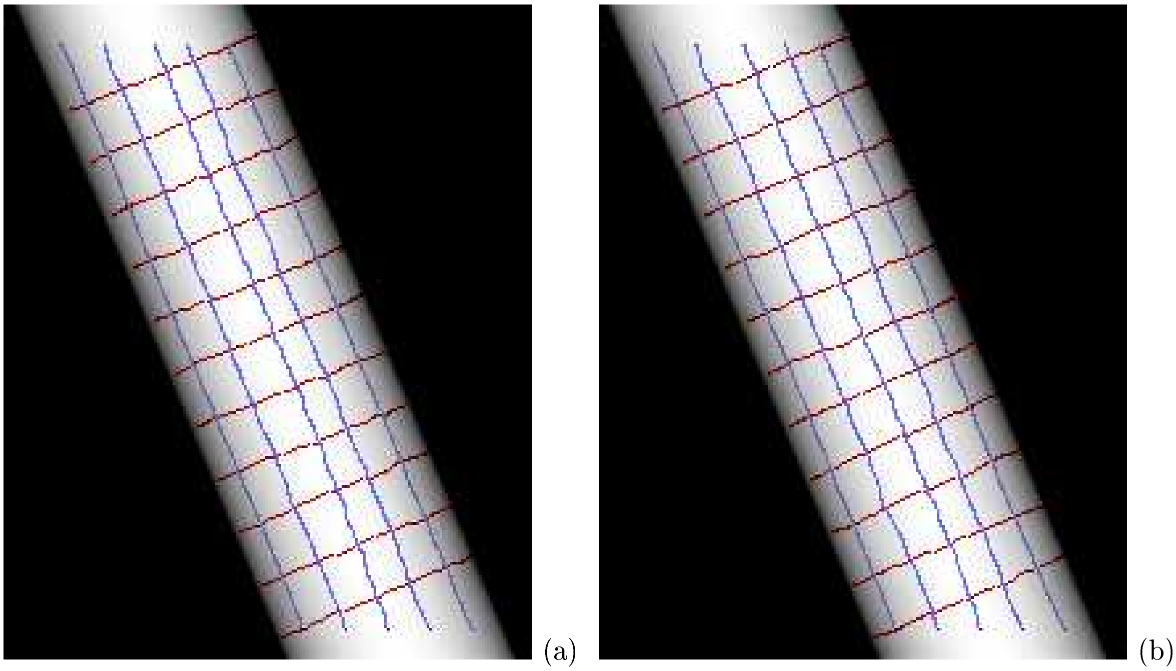

Figure 9: Quality of curvature calculations depending on the function reconstruction scheme – linear (a) vs. cubic (b) reconstruction

[13] G. Nielson and B. Shriver. *Visualization in Scientific Computing*. IEEE Computer Society Press, 1990.

[14] B.-T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[15] M. Rumpf, A. Schmidt, and K. Siebert. Functions defining arbitrary meshes - A flexible interface between numerical data and visualization. *Computer Graphics Forum*, 15(2):129–142, 1996.

[16] G. Sakas, M. Grimm, and A. Savopoulos. Optimized maximum intensity projection (MIP). In *Proceedings* EUROGRAPHICS *Rendering Workshop*, pages 51–63, 1995.

[17] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, 1990.

[18] L. Seidenberg, R. Jerad, and J. Magewick. Surface curvature analysis using color. In *Proceedings IEEE Visualization*, pages 260–267, 1992.