1

SmoothViz: Visualization of Smoothed Particles Hydrodynamics Data

Lars Linsen[1], Vladimir Molchanov[1], Petar Dobrev[1], Stephan Rosswog[1], Paul Rosenthal[2] and Tran Van Long[3]
[1]*Jacobs University, Bremen*
[2]*Chemnitz University of Technology*
[3]*University of Transport and Communication, Hanoi*
[1,2]*Germany*
[3]*Vietnam*

1. Introduction

Smoothed particle hydrodynamics (SPH) is a completely mesh-free method to simulate fluid flow (Gingold & Monaghan, 1977; Lucy, 1977). Rather than representing the physical variables on a fixed grid, the fluid is represented by freely moving interpolation centers ("particles"). Apart from their position and velocity these particles carry information about the physical quantities of the considered fluid, such as temperature, composition, chemical potentials, etc. As the method is completely Lagrangian and particles follow the motion of the flow, the particles represent an unstructured data set at each point in time, i.e., the particles do not exhibit a regular spatial arrangement nor a fixed connectivity. For a recent detailed review of modern formulations of the SPH method see Rosswog (2009).

For the analysis of the simulation results, data visualization plays an important role. However, visualization methods need to account for the highly adaptive, unstructured data representation in SPH simulations. Reconstructing the entire data field over a regular grid is not an option, as it would either use grids of immense sizes that cannot be handled efficiently anymore or it inevitably would introduce significant interpolation errors. Such errors should be avoided, especially as they would occur most prominently in areas of high particle density, i.e., areas of highest importance are undersampled. Adaptive grids may be an option as interpolation errors can be kept low, but the adaptivity requires special treatments during the visualization process.

In this chapter, we introduce visualization methods that operate directly on the particle data, i.e., on unstructured point-based volumetric data. Section 3 introduces an approach to directly extract isosurfaces from a scalar field of the SPH simulation. Isosurfaces extraction is a common visualization concept and is suitable for SPH data visualization, as one is often interested in seeing boundaries of certain features.

Because of the use of radial kernel functions in SPH computations (which is crucial for exact conservation of energy, momentum, and angular momentum) together with a poor a resolution, one can observe that the extracted isosurfaces may be bumpy, especially in regions of low particle density. We approach this issue by introducing level-set methods for

scalar field segmentation that include a smoothing term and extracting isosurfaces from the smooth level-set function. Again, the level-set method is only operating on the positions of the particles and does not use any auxiliary grid to perform the computations. In Section 4, we describe the general approach of smooth isosurface extraction from SPH data based on level-set segmentation and in Section 5, we detail methods for improving the speed of the level-set approach narrow-band processing, a local isosurface extraction approach based on variational level sets, and a non-iterative second-order approximation of the signed distance function which is needed throughout the level-set processing.
The surfaces that are extracted from particle data are in form of a point cloud representation. Point-based rendering methods that display such surfaces without the necessity to first generate a triangular mesh from the point clouds have become popular in computer graphics within the last decades. We have developed an approach that uses image-space operations to create desired renderings of large point clouds at interactive rates without any pre-computations, i.e., not even computing neighborhoods of points. This property is desirable, as we want to interactively extract different surfaces and display them immediately. Section 6 provides the description of our approach including rendering features such as transparency and shadows.
Since SPH simulations include a multitude of fields, it is of interest to investigate them simultaneously and to explore their correlations. In Section 7, we investigate how multi-field features can be detected and visualized. Detection is based on a clustering in the multi-dimensional attribute space. The hierarchy of density clusters can be investigated using coordinated views of the cluster tree, parallel coordinates of the multi-dimensional attribute space, and a visualization of the volumetric physical space. The features are displayed in physical space using surface extraction and rendering.
Finally, in Section 8, we explain how multiple scalar and volume fields can be explored interactively using a visual system based on the methods described in this chapter. In addition to the methods already mentioned, we support some further common visualization functionality for scalar and vector fields.

2. Related work

In the astrophysics SPH community, visualization of slices through the volume, isosurface extraction, direct volume rendering techniques, and particle rendering as color-mapped points are most commonly used for the display of single scalar fields (Navratil et al., 2007; Walker et al., 2005). A tool that provides such functionality (except for isosurfaces) is the freely available visualization tool SPLASH (Price, 2007). The direct volume rendering is executed by a ray casting approach, where integration along the rays is performed by integrating the SPH kernel function. The high adaptivity of the SPH data forces one to use many rays to not lose details in densely populated regions, which makes this purely software-based direct volume rendering approach slow. Rotation, zooming, and similar desired features cannot be achieved at interactive framerates (requiring about 20 frames per second). Navratil et al. (2007) apply an inverse-distance-based interpolation for resampling the data to a regular grid prior to isosurface extraction. Also, volume rendering approaches tend to resample over a regular grid (Cha et al., 2009). However, due to the highly varying particle density (commonly ten orders of magnitude), the precision of these approaches that resample over a static grid is limited.

The generation of tetrahedral meshes from particle data also has a long tradition. Du & Wang (2006) give an overview over various approaches. Widely accepted are the results given by Delaunay tetrahedrization (Delaunay, 1934), whose implementation is also included into the Computational Geometry Algorithms Library (CGAL, 2011). More recent approaches try to improve existing Delaunay tetrahedrization algorithms with respect to robustness, quality, and efficiency. Robustness against numerical errors during Delaunay insertion (Pav & Walkington, 2004) or for boundary recovery (Sapidis & Perucchio, 1991) is desired. Quality criteria with respect to some design goals are often ensured by post-processing steps (Maur & Kolingerová, 2001). The incremental insertion method (Borouchaki et al., 1995; George et al., 1991) is one of the most efficient implementations. Still, computational costs are high. Co & Joy (2005) presented an approach for isosurface extraction from point-based volume data that uses local Delaunay triangulations, which keeps the number of points for each Delaunay triangulation step low and thus improves the overall performance. An approach that also operates locally, but is not based on tetrahedral meshes is given by Rosenberg & Birdwell (2008). They presented an approach based on extracting isosurfaces while marching through slices, which works at interactive rates for smaller number of particles. Our approach (Rosenthal & Linsen, 2006) was the first to extract isosurfaces directly from SPH data, which still outperforms the algorithms listed above.

In terms of volume rendering approaches, splatting of transparent particle sprites is a popular technique (Fraedrich et al., 2009; Hopf & Ertl, 2003; Hopf et al., 2004). A slice-based approach was presented by Biddiscombe et al. (2008). The approach that is closest to the volume rendering approach we propose in here is the work by Fraedrich et al. (2010). Instead of reconstructing the field on a static grid, they use a view-dependent grid. Hence, when the viewing parameters change, the reconstruction is recomputed, which allows for application to the highly adaptive structure of SPH data.

For the visualization of flow fields, direct and 2D streamline visualization methods are supported by the SPH data visualization tool SPLASH (Price, 2007). Other flow visualization methods for SPH data rely on reconstructing over a grid or on extracting and displaying integral lines using the SPH kernel. Schindler et al. (2009) make use of the SPH kernel by presenting a method for vortex core line extraction which operates directly on the SPH representation. It generates smoother and more spatially and temporally coherent results. The underlying predictor-corrector scheme is specialized for several variants of vortex core line definitions.

3. Direct isosurface extraction

Isosurface extraction is a standard visualization method for scalar volume data and has been subject to research for decades. We proposed a method that directly extracts surfaces from SPH simulation data without 3D mesh generation or reconstruction over a structured grid (Rosenthal, 2009; Rosenthal & Linsen, 2006; Rosenthal et al., 2007). It is based on spatial domain partitioning using a kd-tree and an indexing scheme for efficient neighbor search.

In every point in time, the result of an SPH simulation is an unstructured point-based volume data set. More precisely, it is a set of trivariate scalar fields $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, whose values are given for a large, finite set of sample points $\mathbf{x}_i$, whose positions are unstructured, i.e., they are not arranged in a structured way, nor are any connectivity or neighborhood informations known for the sample point locations. To visualize such a scalar field, our intention is to extract an

isosurface $\Gamma_{\text{iso}} = \{\mathbf{x} \in \mathbb{R}^3 \ : \ f(\mathbf{x}) = v_{\text{iso}}\}$ with respect to a real isovalue v_{iso} out of the range of f.

Our approach consists of a geometry extraction and a rendering step. The geometry extraction step computes points $\mathbf{p}_k \in \mathbb{R}^3$ on the isosurface, i.e., $f(\mathbf{p}_k) = v_{\text{iso}}$, by linearly interpolating between neighbored pairs of samples. The neighbor information is retrieved by partitioning the 3D domain into cells using a *k*d-tree. The cells are merely described by their index and bit-wise index operations allow for a fast determination of potential neighbors. We use an angle criterion to select appropriate neighbors from the small set of candidates. The output of the geometry step is a point cloud representation of the isosurface. The final rendering step uses point-based rendering techniques to visualize the point cloud.

In the following, all integers indexed with d, such as a_{d} or 100_{d} denote binary numbers. The operator $\oplus$ denotes the bitwise Boolean exclusive-or operator. Finally, the operators $\ll$ and $\gg$ denote the bit-shift operators, which are recursively defined by

0. $a_{\text{d}} \ll 0 \ = \ a_{\text{d}} \quad \text{and} \quad a_{\text{d}} \gg 0 \ = \ a_{\text{d}}$.
1. $a_{\text{d}} \ll j \ = \ (a_{\text{d}} \ll (j-1)) * 2$.
2. $a_{\text{d}} \gg j \ = \ (a_{\text{d}} \gg (j-1)) \operatorname{div} 2$.

The indexing scheme of the *k*d-tree represents its construction. The father of node with binary index b_{d} has index $b_{\text{d}} \gg 1$ and its children have indices $b_{\text{d}} \ll 1$ and $(b_{\text{d}} \ll 1) \oplus 1_{\text{d}}$. Figure 1 shows a 2D example. Thus, we can navigate through the tree using fast binary operations. Moreover, qualitative propositions about the locations of cells can be made. For instance the cells with index 1111_{d} and 1000_{d} lie in diagonally distant corners of the *k*d-tree. Thus, most information is implicitly saved in the indexing scheme. We exploit this property for fast neighbor search.

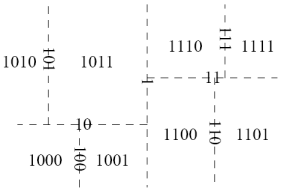


Fig. 1. Indexing scheme for two-dimensional *k*d-tree.

For validation, we have applied our approach to a sphere data set, which consists of randomly distributed sample points in a $200 \times 200 \times 200$ cube. The sample values describe the distance to the center of a sphere. We extract an isosurface from the distance field using isovalue 70. The generated and rendered sphere can be seen in Figure 2. Our direct isosurface extraction algorithm for scattered data produces results of quality close to the results from standard isosurface extraction algorithms for gridded data (like marching cubes). In comparison to 3D mesh generation algorithms (like Delaunay tetrahedrization), our algorithm is about one order of magnitude faster for our examples.

4. Smooth isosurface extraction

SPH uses radial smoothing kernels since they ensure the exact conservation of the physically conserved quantities (Rosswog, 2009). This has as a side effect that the particles are constantly re-adjusting their positions which can lead to "noise" in the particle velocities. Moreover,

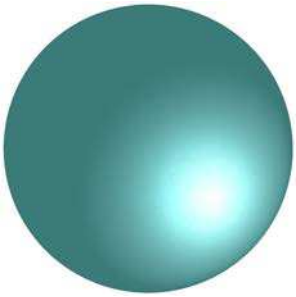


Fig. 2. Isosurface extracted from an example 16M particles.

in particular in sparsely sampled regions the radial kernels can produce some "bumpiness" when direct isosurface extraction from SPH scalar fields is applied. Figure 3(a) shows the result of direct isosurface extraction from SPH data. Here, points on the extracted isosurface are rendered as circular splats to show the noise issue. Hence, it is desirable to add a controllable smoothing term to the isosurface extraction procedure. Smooth surface extraction using partial differential equations (PDEs) is a well-known and widely used technique for visualizing volume data. Existing approaches operate on gridded data and mainly on regular structured grids. When considering unstructured point-based volume data, where sample points do not form regular patterns nor are they connected in any form, one would typically resample the data over a grid prior to applying the known PDE-based methods. We proposed an approach that directly extracts smooth surfaces from unstructured point-based volume data without prior resampling or mesh generation (Rosenthal & Linsen, 2008b).
When operating on unstructured data one needs to quickly derive neighborhood information, which we retrieve from the *k*d-tree. We exploit neighborhood information to estimate gradients and mean curvature at every sample point using a four-dimensional least-squares fitting approach. This procedure finally results in a closed formula for the gradient approximation. For a one-dimensional function φ, represented through the points $(x_1, \varphi_1), \ldots, (x_n, \varphi_n)$, we get

$$\frac{\mathrm{d}\varphi}{\mathrm{d}x} = \frac{n\sum_{i=1}^{n} x_i\varphi_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} \varphi_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \,. \tag{1}$$

It can be shown that this scheme is a generalization of common finite differencing schemes. Having gradients $\nabla\varphi$ of the level-set function φ and mean curvature κ_φ computed, one can apply a PDE-based method that combines hyperbolic advection to an isovalue of a given scalar field and mean curvature flow. A time step is performed with respect to the equation

$$\frac{\partial\varphi}{\partial t} = \left(a\left(f - f_{\mathrm{iso}} - \varphi\right) + b\kappa_\varphi\right)\left|\nabla\varphi\right| \,, \tag{2}$$

which models hyperbolic normal advection, weighted with factor $a > 0$, and mean curvature flow, weighted with factor $b > 0$. This leads to a level-set segmentation algorithm. Since we are solving a partial differential equation by means of an explicit time integration scheme, the

time step is restricted by the Courant-Friedrichs-Lewy (CFL) stability criterion Courant et al. (1928). Due to different conditions in different parts of the fluid, the allowed time steps can vary considerably between different particles. In order to avoid the computational burden of advancing all particles globally on the shortest allowed time step, one can also advance different elements on their own individual time steps, a technique that is commonly used in expensive SPH simulations.

We extract a smooth surface by successively fitting a smooth level-set function to the data set. This level-set function is initialized as a signed distance function. For each sample and for every time step we compute the respective gradient, the mean curvature, and a stable time step. With this information the level-set function is manipulated using an explicit Euler time integration. The process successively continues with the next sample point in time. If the norm of the level-set function gradient in a sample exceeds a given threshold at some time, the level-set function is reinitialized to a signed distance function. After convergence of the evolution, the resulting smooth surface is obtained by extracting the zero isosurface from the level-set function using direct isosurface extraction from unstructured point-based volume data (as described above) and rendering the extracted surface using point-based rendering methods. Figure 3(b) shows the result of our smooth isosurface extraction for the same data set as in Figure 3(a) and using the same splat-based rendering method. It can be observed that the objective of extracting a smooth version of an isosurface has been achieved (especially in the zoomed-in view).

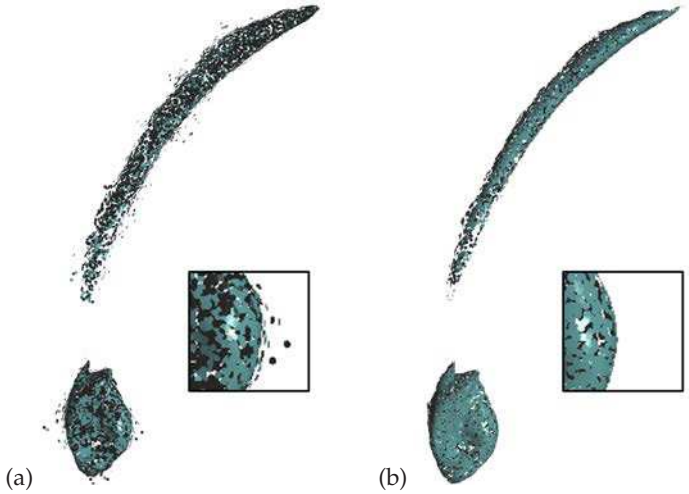


Fig. 3. Comparison between (a) direct isosurface extraction and (b) smooth isosurface extraction for an SPH simulation data set.

5. Acceleration of smooth isosurface extraction

5.1 Narrow-band processing

The global processing of the level-set function can be slow when dealing with unstructured point-based volume data sets containing several million data points. We developed an improved level-set approach that performs the process of the level-set function locally

(Rosenthal et al., 2010). Since for isosurface extraction we are only interested in the zero level set, values are only updated in regions close to the zero level set. In each iteration of the level-set process, the zero level set is extracted using direct isosurface extraction from unstructured point-based volume data and a narrow band around the zero level set is constructed. The band consists of two parts: an inner and an outer band. The inner band contains all data points within a small area around the zero level set. These points are updated when executing the level-set step. The outer band encloses the inner band providing all those neighbors of the points of the inner band that are necessary to approximate gradients and mean curvature. As before, neighborhood information is obtained using an efficient kd-tree scheme, gradients and mean curvature are estimated using a four-dimensional least-squares fitting approach.
The construction of the two-layer band around the zero level set is shown in Figure 4. The zero level set (colored blue) is extracted in form of a point cloud representation. Then, all sample points with distance to the zero level set less than a distance d_α are marked as belonging to the inner layer of the band (green). Thereafter, all additional sample points needed for the gradient computations within the level-set process are marked as belonging to the outer layer of the band (red points). All other sample points (grey) are not used for the current level-set step. The distance d_α can be estimated using the CFL condition that bounds the level-set step.

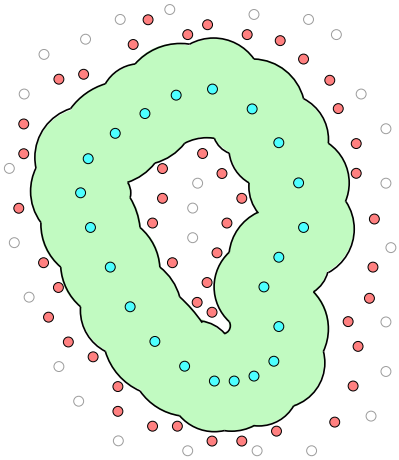


Fig. 4. Narrow-band construction for more efficient level-set processing.

How the level-set function is updated after having executed a level-set iteration on the narrow band with size d_α is shown in Figure 5: Points (green) with minimum distance to the zero-level-set points (blue) smaller than $\frac{d_\alpha}{4}$ have been in the α-band in the preceding time step and, thus, their level-set function values have been updated in the level-set iteration step. Points (red) in the outer band or with distance to the zero level set greater than $\frac{d_\alpha}{2}$ might have not be included in the computations of the last level-set iteration step. We assign their level-set function value to the signed distance to the zero-level-set points. For all points in the α-band with a distance to the zero level set in the range $\left[\frac{d_\alpha}{4}, \frac{d_\alpha}{2}\right]$, the new level-set function value is interpolated between the level-set function value from the preceding level-set step and their signed distance to the zero-level-set points.

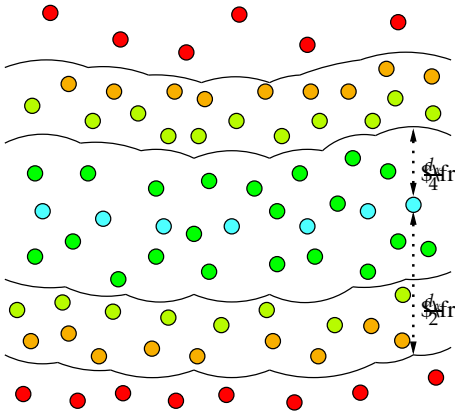


Fig. 5. Narrow-band update during level-set procedure (d_α denotes the size of the narrow band).

For performance analysis we applied it to an unstructured point-based volume data set with eight million randomly distributed samples. The data set was generated by resampling the regular Hydrogen data set of size $128 \times 128 \times 128$ to the random points. Illustrations of the evolution process for this data set are shown in Figure 6 (Data set courtesy of Peter Fassbinder and Wolfgang Schweizer, SFB 382 University Tübingen.). For each time step, a splat-based ray tracing of the zero level set is shown on the right-hand side. On the left-hand side, a point rendering of a slab of the data set is shown illustrating the narrow band. Extracted surface points of the zero level set are colored black, sample points in the α-band are colored green, and sample points in the outer band are colored red. Sample points not belonging to the narrow band are not rendered.
The whole local level-set process for extracting a smooth isosurface from the Hydrogen data set with eight million sample points and given nearest neighbors needed 24 steps and was performed in 6 minutes. For the four million version of the data set, the overall computation time for the entire level-set approach including pre-computations dropped to 84 seconds. This is a significant speed-up in comparison to the time of 13 minutes needed without the narrow-band processing. Still, it produces equivalent results in terms of quality and correctness.

5.2 Variational level-set detection of local isosurfaces

Another acceleration of the level-set approach can be achieved by only operating locally on a region of interest. We derived a variational formulation for smooth local isosurface extraction using an implicit surface representation in form of a level-set approach, deploying a moving least-squares (MLS) approximation, and operating on a kd-tree (Molchanov et al., 2011). The locality of our approach has two aspects: First, our algorithm extracts only those components of the isosurface, which intersect a subdomain of interest. Second, the action of the main term in the governing equation is concentrated near the current isosurface position. Both aspects reduce the computation times per level-set iteration. As for most level-set methods a reinitialization procedure is needed, but we also consider a modified algorithm where this step is eliminated. The final isosurface is extracted in form of a point cloud representation.

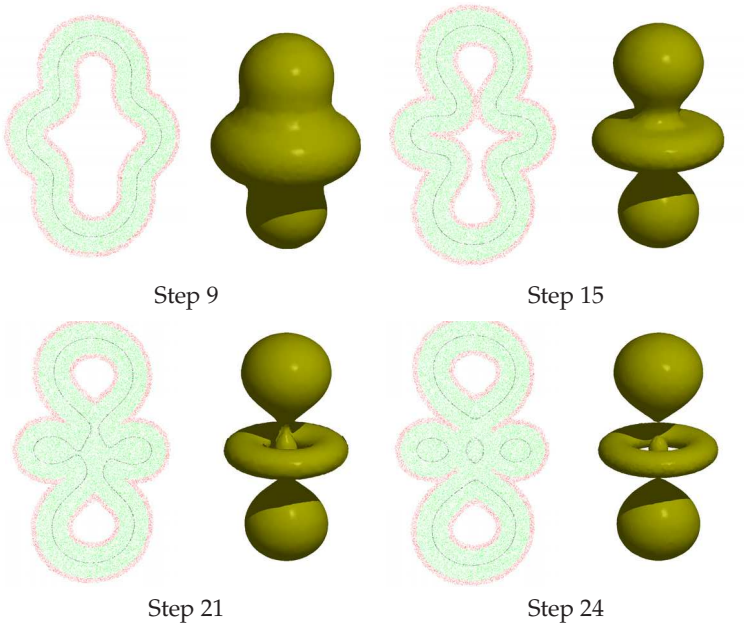


Fig. 6. Narrow-band level-set evolution.

We also presented a novel point completion scheme that allows us to handle highly adaptive point sample distributions.
A variational approach is used to derive the local level-set updates. We start with a construction of an error functional E, which is the target function that we want to minimize. The error depends on the given data f, the constant f_{iso} representing the isovalue, and a level-set function φ together with its derivatives of first order, i.e., $E = E(\varphi, \nabla\varphi; f, f_{\text{iso}})$. The total error consists of two weighted terms

$$E = E_1 + \lambda E_2, \tag{3}$$

measuring accuracy and smoothness of the solution, respectively. We propose to use

$$E_1 = \frac{1}{4}\int_D (\text{sgn}(\varphi(\mathbf{x})) - \text{sgn}(f(\mathbf{x}) - f_{\text{iso}}))^2 \, \mathrm{d}\mathbf{x}, \tag{4}$$

and

$$E_2 = \int_D \delta(\varphi(\mathbf{x})) \, |\nabla\varphi(\mathbf{x})| \, \mathrm{d}\mathbf{x}. \tag{5}$$

Here, we use the standard definitions of the sign function $\text{sgn}(x)$ and the Dirac function $\delta(x)$ and the derivative is used in the sense of distributions.
A function φ_∞ minimizing some functional of the form $\int L(\varphi, \nabla\varphi)\,\mathrm{d}\mathbf{x}$ should satisfy the Euler-Lagrange equation

$$\left(\frac{\partial L}{\partial \varphi} - \sum_i \frac{\partial}{\partial x_i}\frac{\partial L}{\partial \varphi_i}\right)\Bigg|_{\varphi=\varphi_\infty} = 0, \tag{6}$$

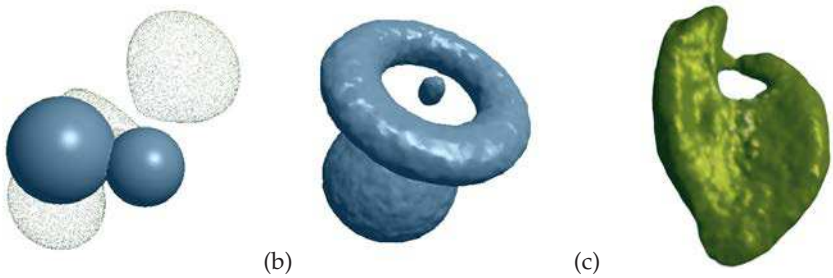
where φ_i is the i-th component of $\nabla\varphi$. We derive the Euler-Lagrange equations for functionals E_1 and E_2. The idea of a level-set approach is to detect φ_∞ as a fixed point of an evolution equation for $\varphi = \varphi(\mathbf{x}, t)$ minimizing E. Here, t is an artificial time parameterizing the minimization process $\varphi(\mathbf{x}, t) \rightarrow \varphi_\infty(\mathbf{x})$ as $t \rightarrow \infty$. To construct the PDE, one equates the left-hand side of Euler-Lagrange Equation with $-\partial\varphi/\partial t$. For the functional E, it reads

$$\frac{\partial\varphi}{\partial t} = \delta(\varphi)\,(\mathrm{sgn}(f - f_{\mathrm{iso}}) - \mathrm{sgn}(\varphi)) + \lambda\,\delta(\varphi)\,\nabla\cdot\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right). \tag{7}$$

Subsequently, this equation is regularized and discretized in space and time leading to an iterative process for the value of φ at each sample point. Using an explicit Euler time integration, we obtain the iteration step

$$\varphi^{k+1} = \varphi^k + \Delta t\left[\delta_\epsilon(\varphi^k)\,(\mathrm{sgn}(f - f_{\mathrm{iso}}) - \mathrm{sgn}_\epsilon(\varphi^k)) + \lambda\,\delta_{\epsilon'}(\varphi^k)\,\nabla\cdot\left(\frac{\nabla\varphi^k}{|\nabla\varphi^k|}\right)\right] \tag{8}$$

which is applied to all sample points $\mathbf{x}_i$, where the upper index k stands for the k-th layer in time. This update rule can be applied locally. Figure 7 shows how smooth isosurface components are extracted locally within a given region of interest.



(a) (b) (c)

Fig. 7. Local level-set evolution of isosurfaces within a region of interest for multi-component data sets: (a) selecting region of interest; (b) extracted isosurface components; (c) same procedure applied to extract one component of an isosurface from the density field of an SPH simulation of a white dwarf.

5.3 Non-iterative second-order approximation of signed distance function

Signed distance functions are an obligatory ingredient to the level-set methods. When assuming that the underlying function is a signed distance function, several simplifications and speed-ups of the level set approach can be achieved. Usual approaches for the construction of signed distance functions to a surface are either based on iterative solutions of a special partial differential equation or on marching algorithms involving a polygonization of the surface. We propose a novel method for a non-iterative approximation of a signed distance function and its derivatives in a vicinity of a manifold. We use a second-order scheme to ensure higher accuracy of the approximation (Molchanov et al., 2010).

The manifold is defined (explicitly or implicitly) as an isosurface of a given scalar field, which may be sampled at a set of irregular and unstructured points. We use a spatial subdivision in form of a fast kd-tree implementation to access the samples and perform transformations on

the data. We derive a novel moving least-squares (MLS) approach for a second-order algebraic fitting to locally reconstruct the isosurface. Stability and reliability of the algorithm is achieved by a proper scaling of the MLS weights, accurate choices of neighbors, and appropriate handling of degenerate cases. We obtain the solution in an explicit form, such that no iterative solving is necessary, which makes our approach fast. The accuracy relies on second-order algebraic fitting.

We propose to perform the following steps to construct a signed distance field around an implicitly given isosurface:

1. Given a scalar field $f(\mathbf{x})$ on samples $\mathbf{x}_i$, extract a set of isopoints $\mathbf{p}_j$ corresponding to the isovalue f_{iso}; estimate normals on isopoints $\mathbf{n}_j$ using kd-tree and MLS technique. Skip this step if the isosurface is given explicitly.
2. For a given $\alpha > 0$ mark all samples $\mathbf{x}_i$ lying in α-neighborhood of the isosurface; jointly establish lists of neighboring isopoints for the marked samples.
3. For each sample $\mathbf{y}_k$ in the band find its neighbors and check two angles to detect a layer sheet.
4. If the sample lies between isosurface components, compute the distances to both of them as in the next step, compare the values found and take the minimal one.
5. If the criterion for a layer is not fulfilled:
   - perform a local sphere fitting to reconstruct a part of the isosurface close to $\mathbf{y}_k$,
   - if the sphere degenerates to a plane, compute a distance between the sample and the surface (taking into consideration its orientation),
   - if the sphere is not degenerated, use the isopoint normals to analyse its convexity and compute the distance between the sample and the sphere.

An accurate computation of the distance between a sample and an isosurface is hard if the isosurface is represented as a sparse set of isopoints. Therefore a (local) reconstruction of the smooth surface is required. In our approach we find an implicit algebraic surface to fit the discrete data which includes isopoints positions and associated normals. We consider algebraic spheres of the form

$$s(\mathbf{x}) = a_0 + \mathbf{a} \cdot \mathbf{x} + a_4 \mathbf{x} \cdot \mathbf{x}, \tag{9}$$

where $\mathbf{a} = (a_1, a_2, a_3)$ and $\mathbf{x} = (x_1, x_2, x_3)$. The solution may naturally degenerate to a plane as a_4 vanishes and therefore is exact for flat surfaces. However, a direct enrichment of the class is problematic: there exist no analytic formula for distance to algebraic ellipsoids. We utilize an approach of algebraic sphere fitting using positional and derivative constraints. First, we find $m+1$ isopoints nearest to the point of interest $\mathbf{y}$. Let h be the distance from $\mathbf{y}$ to the farthest neighbor. This parameter will define the support size of the weighting function

$$\omega_{\mathbf{y}}(\mathbf{p}) = \max\left\{ \left(1 - \frac{\|\mathbf{y}-\mathbf{p}\|^2}{h^2}\right)^4,\ 0 \right\}. \tag{10}$$

Now we look for the optimal algebraic sphere, whose zero-isosurface $\{\mathbf{x} \in \mathbb{R}^3 : s(\mathbf{x}) = 0\}$ optimally fits the positions of the isopoints, i.e., $s(\mathbf{p}_j) = 0$, and their normals, i.e., $\nabla s(\mathbf{p}_j) = \mathbf{n}_j$. The best fit is defined by parameters $a_0, \ldots, a_4$ minimizing the cost function

$$E(a_0, \ldots, a_4) = \sum_{j=1}^{m} \omega_j \left[|s(\mathbf{p}_j)|^2 + \beta \|\nabla s(\mathbf{p}_j) - \mathbf{n}_j\|^2 \right] \tag{11}$$

with $\omega_j = \omega_{\mathbf{y}}(\mathbf{p}_j)$. The minimization problem

$$s_{\text{opt}}(\mathbf{x};\, a_0, \ldots, a_4) = \arg\min E \tag{12}$$

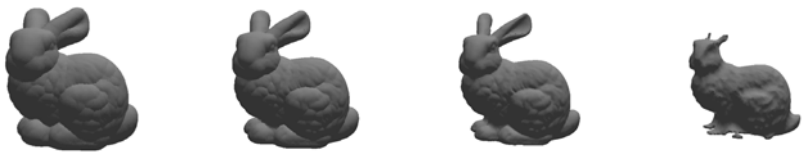
has the following explicit solution

$$a_4 = \frac{\beta}{2} \frac{\sum \omega_j \mathbf{p}_j \cdot \mathbf{n}_j - \sum \omega_j \mathbf{p}_j \cdot \sum \omega_j \mathbf{n}_j / \Omega}{\sum \omega_j \mathbf{p}_j \cdot \mathbf{p}_j - \sum \omega_j \mathbf{p}_j \cdot \sum \omega_j \mathbf{p}_j / \Omega}, \tag{13}$$

$$\mathbf{a} = \sum \omega_j \mathbf{n}_j / \Omega - 2a_4 \sum \omega_j \mathbf{p}_j / \Omega, \tag{14}$$

$$a_0 = \mathbf{a} \cdot \sum \omega_j \mathbf{p}_j / \Omega - a_4 \sum \omega_j \mathbf{p}_j \cdot \mathbf{p}_j / \Omega, \tag{15}$$

where $\Omega = \sum \omega_j$.
Figure 8 shows the result when extracting different isosurfaces from a signed distance field to a surface in explicit point cloud representation. Isosurfaces for different isovalues f_{iso} of the signed distance function field constructed for the bunny data set with $35k$ surface points, (Data set courtesy of the Stanford University Computer Graphics Laboratory.).



$f_{\text{iso}} = 4.0$ $f_{\text{iso}} = 2.0$ $f_{\text{iso}} = 0.0$ $f_{\text{iso}} = -3.5$

Fig. 8. Isosurfaces extracted from a non-iterative second-order approximation of the signed distance function to a surface in point cloud representation.

We proposed a novel method for the efficient computation of a signed distance function to a surface in point cloud representation. This allows us to develop a fast level-set approach for extracting smooth isosurfaces from point-based volume data, as we can use any point cloud surface as initial zero level set. Since for most applications a rough estimate of the desired surface can be obtained quickly, the overall level-set process can be shortened significantly. Additionally, we can avoid the computational overhead and numerical difficulties of PDE-based reinitialization.
In summary, putting all acceleration methods together we achieved to reduce the computation costs for the entire level-set approach including all components by about two orders of magnitude. For data sets with 16 million particles, the processing time dropped from tenths of minutes to tenths of seconds.

6. Image-space point cloud surface rendering

The extracted surfaces are given in point cloud surface representation, i.e., points on the surfaces with no structure or neighborhood known. Reconstructing a triangular mesh from

these point clouds can be very time consuming. Instead, point-based rendering approaches have gained a major interest in recent years, basically replacing global surface reconstruction with local surface estimations using, for example, splats or implicit functions. Crucial to their performance in terms of rendering quality and speed is the representation of the local surface patches. We presented a novel approach that goes back to the original ideas of Grossman & Dally (1998) to avoid any object-space operations and compute high-quality renderings by only applying image-space operations (Rosenthal & Linsen, 2008a).
Starting from a point cloud including normals (obtained from gradients of the underlying scalar field), we render a lit point cloud to a texture with color, depth, and normal information. Subsequently, we apply several filter operations. In a first step, we use a mask to fill background pixels with the color and normal of the adjacent pixel with smallest depth. The mask assures that only the desired pixels are filled. We use the eight masks shown in Figure 9, where the white pixels indicate background pixels and the dark pixels could be both background or non-background pixels. For each background pixel, we test whether the 3×3 neighborhood of that pixel matches any of the cases. In case it does, the pixel is not filled. Otherwise, it is filled with the color and depth information of the pixel with smallest depth out of the 3×3 neighborhood.

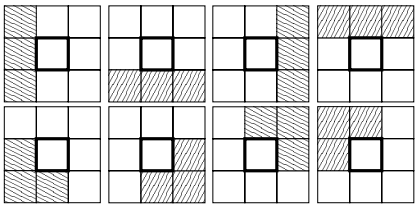


Fig. 9. Filters with size 3×3 for detecting whether a background pixel is beyond the projected silhouette of the object. If one of the eight masks matches the neighborhood of a background fragment, it is not filled. White cells indicate background pixels, dark cells may be background or non-background pixels.

Similarly, in a second pass, we fill the pixels that display occluded surface parts. The resulting piecewise constant surface representation does not exhibit holes anymore and is smoothed by a standard smoothing filter in a third step. The same three steps can also be applied to the depth channel and the normal map such that a subsequent edge detection and curvature filtering leads to a texture that exhibits silhouettes and feature lines. Anti-aliasing along the silhouettes and feature lines can be obtained by blending the textures. When highlighting the silhouette and feature lines during blending, one obtains illustrative renderings of the 3D objects. The GPU implementation of our approach achieves interactive rates for point cloud renderings without any pre-computation. Figure 10 shows the individual steps of the proposed pipeline including illustrative rendering. Figure 13(a) shows such another rendering result for a data set with 883k surface points. The rendering is performed at 52 frames per second (fps).
For a realistic visualization of the models, transparency and shadows are essential features. We propose extensions to our method for point cloud rendering with transparency and shadows at interactive rates (Dobrev et al., 2010a;b). Again, our approach does not require any global or local surface reconstruction method, but operates directly on the point cloud. All passes are executed in image space and no pre-computation steps are required. The

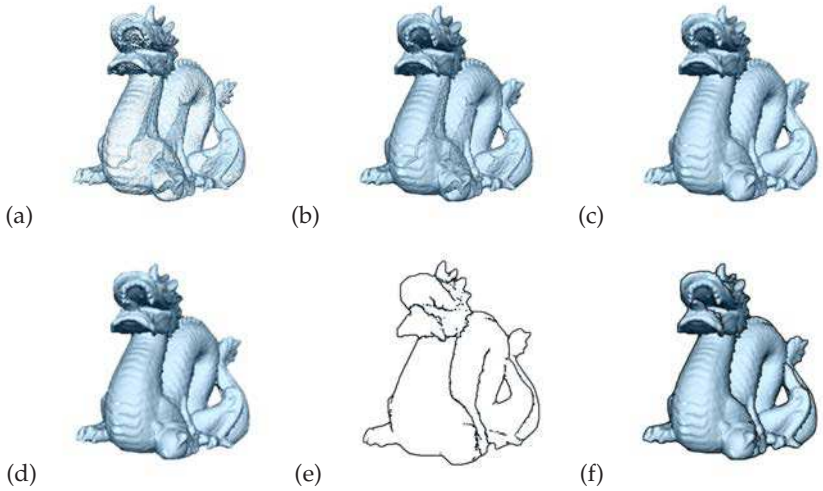


Fig. 10. Image-space point cloud surface rendering pipeline applied to the Dragon data set (Data set courtesy of Stanford University Computer Graphics Lab.): (a) Lit points; (b) after filling background pixel; (c) after filling occluded pixels; (d) after smoothing; (e) extracted feature lines; (f) illustrative rendering.

underlying technique for our approach is a depth peeling method for point cloud surface representations. The idea of depth peeling is to successively remove the front layer to extract hidden layers. Hence, one virtually renders the object and all visible parts represent the first layer. This is removed and the process is iterated to successively extract all hidden layers. For surfaces in point cloud representation, another challenge arises, as shown in Figure 11. When projecting the first layer (blue) in point cloud representation to the screen, the layer exhibits holes such that hidden layers (red) or the background (grey) become visible. To overcome the problem we use, again, the image-space masks presented above to produce layers without holes. Figure 12 shows four different layers of the Blade data set that are extracted using depth peeling.

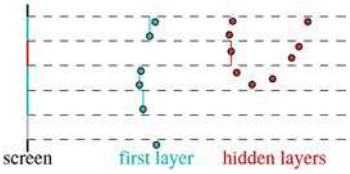


Fig. 11. Depth peeling for point cloud surfaces.

Having detected a sorted sequence of surface layers, they can be blended front to back with given opacity values to obtain renderings with transparency. These computation steps achieve interactive frame rates. Figure 13(b) shows a rendering with transparency. The rendering is performed at 17.5 fps.

To determine which parts of a surface are directly lit by a light source and which parts fall into the shadow of the light source, we determine and mark all points that are visible from the light

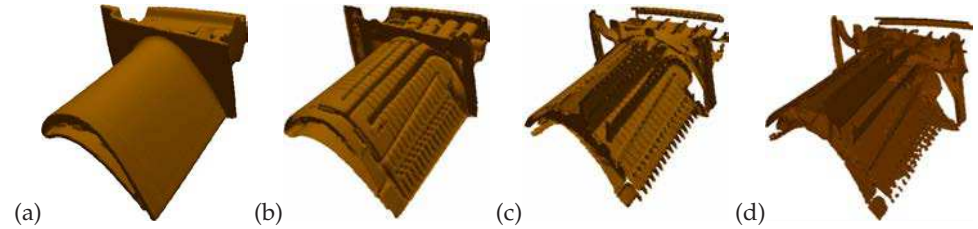


Fig. 12. Four successive layers extracted using depth peeling on point cloud surface representations.

source similar to "pre-baking" irradiance textures for polygonal mesh scenes. We use point cloud shadow textures, which are basically Boolean arrays that store which points are lit. Once the shadow texture is determined, lit points are drawn properly illuminated with ambient, diffuse, and specular reflection components using Phong's illumination model (Phong, 1975), while unlit points are only drawn using the ambient reflection component. This illumination creates the effect of shadows, as only those points are marked unlit where the light source is occluded by other surface parts.

To determine which points are visible from the light source, we render the scene with the light source's position being the viewpoint with depth testing enabled. All visible points are marked in an array. However, we observe that, due to the high point density, it is not unusual that several adjacent points of one surface layer project to the same fragment position. The suggested procedure would only mark the closest point for each fragment as lit, which would lead to an inconsistent shadow textures. Again, depth peeling is the key to solve this problem, but we apply it differently. While for transparent surface rendering our goal was to extract different surface layers, now we use it to find all the points that belong to a single surface layer, namely the closest one.

For the shadow texture computation, we also apply a Monte-Carlo integration method to approximate light from an area light source, leading to soft shadows. Shadow computations for point light sources are executed at interactive frame rates. Shadow computations for area light sources are performed at interactive or near-interactive frame rates depending on the approximation quality. Figure 13(c) shows a rendering with shadows using the Monte-Carlo approach. The rendering with 5 Monte Carlo samples is performed at 9 fps, while the rendering without Monte-Carlo sampling, i.e., with 1 sample, is performed at 25 fps.

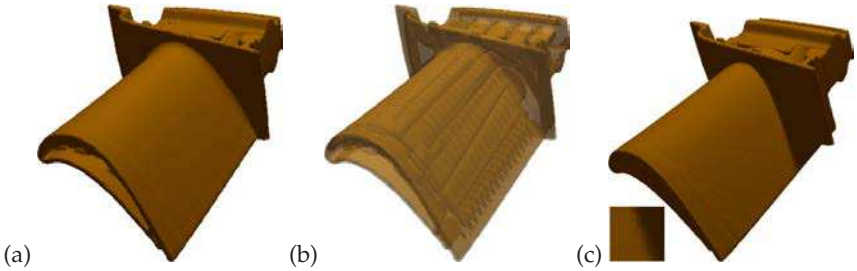


Fig. 13. (a) Image-space point cloud surface rendering applied to the Blade data set (courtesy of Visualization Toolkit). (b) Rendering with transparency using depth-peeling approach. (c) Rendering with shadows using Monte-Carlo integration.

We also investigated the use of ray tracing techniques for high-quality rendering based on splat representations, but the complexity of this approach impedes interactivity (Linsen et al., 2007).

7. Surface extraction from multiple fields

As the data sets resulting from SPH simulations typically contain a multitude of physical variables, it is desirable that visualization methods take into account the entire multi-field volume data rather than concentrating on one variable. We presented a visualization approach based on surface extraction from multi-field particle volume data (Linsen et al., 2008). The surfaces segment the data with respect to the underlying multi-variate function. Decisions on segmentation properties are based on the analysis of the multi-dimensional attribute space. The attribute space exploration is performed by an automated multi-dimensional hierarchical clustering method, whose resulting density clusters are shown in the form of density level sets in a 3D star coordinate layout (Long, 2010; Long & Linsen, 2011). In the star coordinate layout, the user can select clusters of interest. A selected cluster in attribute space corresponds to a segmenting surface in object space. Based on the segmentation property induced by the cluster membership, we extract a surface from the volume data. We directly extract our surfaces from the SPH data without prior resampling or grid generation. The surface extraction computes individual points on the surface, which is supported by an efficient neighborhood computation. The extracted surface points are, again, rendered using point-based rendering operations. Our approach combines methods in scientific visualization for object-space operations with methods in information visualization for attribute-space operations.

7.1 Attribute space visualization

Given the multi-dimensional attribute space with a large number of d-dimensional points lying in that attribute space, each point corresponds to one sample of the volumetric data field and each dimension represents one data attribute (typically one scalar value) stored at that sample. In order to understand the distribution of the points in attribute space, we propose to compute a density function and to determine the number of clusters as well as the high density region of each cluster. Given a multivariate density function $f(x)$ in d dimensions, modes of $f(x)$ are positions where $f(x)$ has local maxima. Thus, a mode of a given distribution is more dense than its surrounding area. We want to find the attraction regions of modes. To do so, we choose various values for constants λ $(0 < \lambda < \sup_x f(x))$ and consider regions of the particle space where values of $f(x)$ are greater than or equal to λ. The λ-level set of the density function $f(x)$ denotes a set $S(f,\lambda) = \{x \in \mathbb{R}^d : \ f(x) \geq \lambda\}$. The set $S(f,\lambda)$ consists of a number q of connected components $S_i(f,\lambda)$ that are pairwise disjoint. The subsets $S_i(f,\lambda)$ are called λ-density clusters (λ-clusters for short). A cluster can contain one or more modes of the respective density function. Let the domain of the data set be given in the form of a d-dimensional hypercube, i. e., a d-dimensional bounding box. To derive the density function, we spatially subdivide the domain of the data set into cells of equal shape and size. Thus, the spatial subdivision provides a binning into d-dimensional cells. For each cell we count the number of points lying inside. The multivariate density function $f(x)$ is given by the number of points per cell divided by the cell's area and the overall number of data points. As the area is equal for all cells, the density of each cell is proportional to the number of data points lying inside the cell. The cell should be small enough such that local changes of the density

function can be detected but also large enough to contain a large number of points such that averaging among points is effective. Because of the curse of dimensionality, there will be many empty cells. We do not need to store empty cells such that the amount of cells we are storing and dealing with is (significantly) smaller than the number of the d-dimensional points. The λ-clusters can be computed by detecting regions of connected cells with densities larger than λ. As we identify density with point counts, the densities are integer values. Hence, we start by computing density clusters for $\lambda = 1$. Subsequently, we process each detected λ-cluster individually by iteratively removing those cells with minimum density, where the minimum density increases in steps of 1. If this process causes a cluster to fall into two subclusters, the subclusters represent higher-density clusters within the original cluster. If a cluster does not fall into subclusters during the process, it is a mode cluster. This process generates a hierarchical structure, which is summarized by the high density cluster tree (short: cluster tree). The root of the cluster tree represents all points. Figure 14(a) shows a cluster tree with 4 mode clusters represented by the tree's leaves. Cluster tree visualization provides a method to understand the distribution of data by displaying the attraction regions of modes of the multivariate density function. Each cluster contains at least one mode.

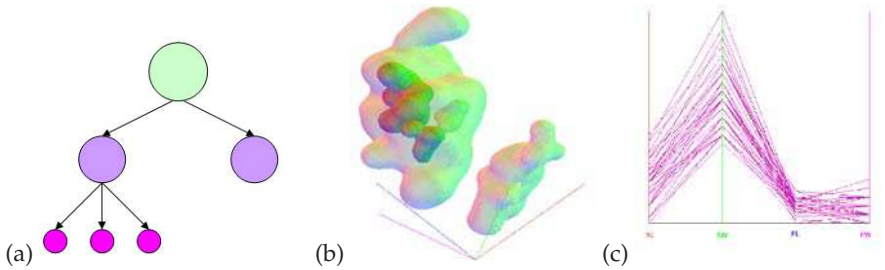


Fig. 14. (a) Cluster tree of density visualization with four modes shown as leaves of the tree. (b) Nested density cluster visualization based on cluster tree using 3D star coordinates. (c) Right-most cluster in (b) is selected and its homogeneity is evaluated using parallel coordinates.

Having computed the d-dimensional high density clusters, we need to project them into a three-dimensional space for visualization purposes. In order to visualize the high density clusters in a way that allows clusters to be correlated with the d dimensions, we need to use a coordinate system that incorporates all d dimensions. Such a coordinate system can be obtained by using star coordinates. When projecting the d-dimensional high density clusters into a three-dimensional star coordinate representation, clusters should remain clusters. Thus, points that are close to each other in the d-dimensional feature space should not be further apart after projection into the three-dimensional space. Let O be the origin of the $3D$ star coordinate system and $(a_1, \ldots, a_d)$ be a sequence of d three-dimensional vectors representing the axes. The mapping of a d-dimensional data point $x = (x_1, \ldots, x_d)$ to a three-dimensional data point $\Pi(x)$ is determined by the average sum of vectors a_k of the $3D$ star coordinate system multiplied with its attributes x_k for $k = 1, \ldots, d$, i.e.,

$$\Pi(x) = O + \frac{1}{d} \sum_{k=1}^{d} x_k a_k. \tag{16}$$

Since it can be shown that

$$||\Pi(x) - \Pi(y)||_1 \leq ||x - y||_1 \tag{17}$$

for any d-dimensional points x and y, the distance of the images of two d-dimensional points is lower than or equal to the distance of the points with respect to the L_1-norm. Therefore, two points in the multi-dimensional space are projected to $3D$ star coordinates preserving the similarity properties of clusters (at least with respect to the L_1-norm). In other words, the mapping of d-dimensional data to the 3D visual space does not break clusters. The second property that our projection from multi-dimensional feature space into three-dimensional star coordinate systems should fulfill is that separated clusters should not be projected into the same region. The projection into star coordinates may cause severe cluttering of clusters when not carefully choosing the axes $(a_1, \ldots, a_d)$. To alleviate the problem of overlapping clusters we introduce a method which chooses a "good" coordinate system. Assume that a hierarchy of high density clusters have q mode clusters, which do not contain any higher level densities. Let m_i be the barycenter of the points within the ith cluster, $i = 1, \ldots, q$. We want to choose a projection that maintains best the distances between clusters. Let $\{v_1, v_2, v_3\}$ be an orthonormal basis of the candidate three-dimensional space of projections. The desired choice of a $3D$ star coordinate layout is to maximize the distance of the q projected barycenters $V^T m_i$ with $V = [v_1, v_2, v_3]^T$, i.e. to maximize the objective function

$$\sum_{i<j} ||V^T m_i - V^T m_j||^2 = \text{trace}(V^T S V) \tag{18}$$

with

$$S = \sum_{i<j} (m_i - m_j)(m_i - m_j)^T . \tag{19}$$

Thus, the three vectors v_1, v_2, v_3 are the three unit eigenvectors corresponding to the three largest eigenvalues of matrix S. This step is a principal component analysis (PCA) applied to the barycenters of the clusters. As a result, we choose the d three-dimensional axes of the $3D$ star coordinate system as $a_i = (v_{1i}, v_{2i}, v_{3i})$, $i = 1, \ldots, d$.

Obviously, we can also project into 2D coordinates in the same way. However, when comparing and evaluating projections to 2D and 3D visual space (Poco et al., 2011), a quantitative analysis confirms that 3D projections outperform 2D projections in terms of precision. Moreover, a user study indicates that certain tasks can be more reliably and confidently answered with 3D projections. Nonetheless, as 3D projections are displayed on 2D screens, interaction is more difficult.

After having computed the projected clusters, we can display them using star coordinates by rendering a point primitive for each projected data point. A less cluttered and more beautiful display is to render the boundary of the clusters. Considering the cluster that is described by the set of points $\{p_i = (x_i, y_i, z_i) : i = 1, \ldots, m\}$ after being projected into the 3D space. In order to compute the boundary of this group of points,we need to have a continuous representation of the group. Therefore, we consider the function

$$f_h(p) = \sum_{i=1}^{m} K\left(\frac{p - p_i}{h}\right), \; p \in \mathbb{R}^3 , \tag{20}$$

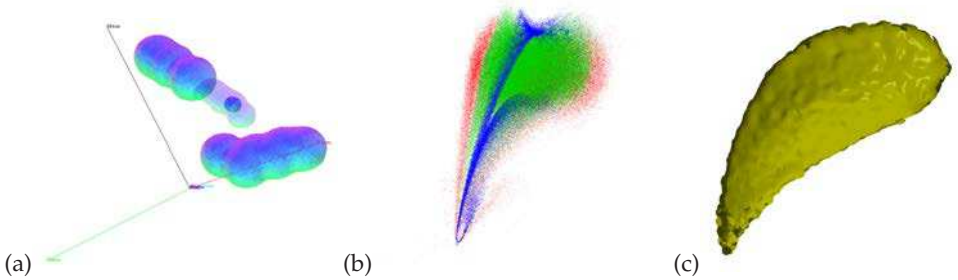
where K is a kernel function and h is the bandwidth. Then, we can reconstruct the field over a regular grid and render the boundary set of the points by using standard isosurface extraction

methods to extract the boundary surface of the set $S(h,c) = \{p \in \mathbb{R}^3 : f_h(p) \geq c\}$, where c is an isovalue. We choose parameter h and c to guarantee that $S(h,c)$ is connected and has a volume of minimum extension. The kernel function should be sufficiently smooth and have a small compact support. For example, we can choose $K(p) = (1 - ||p||^2)^2$ for $||p|| \leq 1$ and $K(p) = 0$ otherwise and the bandwidth h to be equal to the longest length of the minimum spanning tree of these m points. In Figure 14(b) we show the visualization of the clusters by rendering such boundary surfaces, where it can be shown that for the chosen kernel isovalue $c = \frac{9}{16}$ is appropriate. In order to visualize all clusters of the cluster tree, we render the surfaces in a semi-transparent fashion. The resulting visualization shows sequences of nested surfaces, where the inner surfaces represent higher density levels. Figure 14(b) shows the nested density cluster visualization with respect to the cluster tree in Figure 14(a).

7.2 Coordinated views

Generating all clusters and displaying them in star coordinates allows for further analysis of the detected clusters. The simplest interaction method is to select individual clusters by just clicking at the boundary surface. When a cluster is selected, intra-cluster variability is visualized using parallel coordinates, see Figure 14(b) and (c). In both pictures the relation between the selected cluster with the dimension can be observed.

Moreover, we visualize the coordinated view in physical space, which exhibits the spatial location of the selected feature. The rendering in physical space can be preformed by just plotting all particles that belong to the selected feature or by extracting a boundary surface of that feature, i.e., a surface that separates all particles that belong to the feature from all particles that do not belong to the feature. Figure 15 shows an attribute-space rendering of the detected clusters in 3D optimized star coordinates (a), a color-coded object-space rendering of the clustered particles (b), and a separation surface of clusters in object space (c). The underlying SPH simulation is that of tidal disruption and ignition of a white dwarf by a moderately massive black hole (Rosswog et al., 2009).



(a) (b) (c)

Fig. 15. (a) Seven-dimensional attribute space visualization of SPH data set using optimized 3D star coordinates. (b) Object space visualization of cluster distribution. (c) Object space visualization of a separating surface.

For the visualization of enclosing surfaces in attribute as well as in object space, we looked into an alternative approach of enclosing surfaces for point clusters using 3D discrete Voronoi diagrams (Rosenthal & Linsen, 2009). Our system provides three different types of enclosing surfaces. By generating a discrete distance field to the point cluster and extracting an isosurface from the field, an enclosing surface with any distance to the point cluster can be

generated. As a second type of enclosing surfaces, a hull of the point cluster is extracted. The generation of the hull uses a projection of the discrete Voronoi diagram of the point cluster to an isosurface to generate a polygonal surface. Generated hulls of non-convex clusters are also non-convex. The third type of enclosing surfaces can be created by computing a distance field to the hull and extracting an isosurface from the distance field. This method exhibits reduced bumpiness and can extract surfaces arbitrarily close to the point cluster without losing connectedness. Figure 16 shows the idea of the different approaches starting from an isosurface from the distance field to the point cluster (a), connecting the neighbors that contribute to the surface in (a) to form a non-convex hull (b), and computing surfaces that are equidistant to the computed non-convex hull (b). Figure 17 shows a comparison of the different enclosing surfaces when applied to a cluster of points when projected into optimized star coordinates.

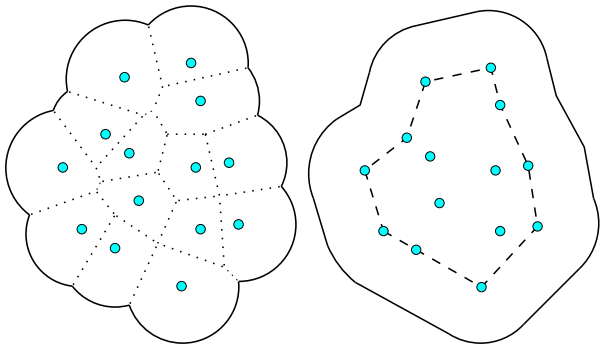


Fig. 16. (a) Extracting an isosurface from the distance field to the point cluster. Voronoi regions on the isosurface induce neighborhoods. (b) Neighbors are connected to form a hull. The image also shows an isosurface extracted from the distance field to the hull.

We extended our work on interactivity by explicitly encoding the cluster hierarchy in a tree that is visually encoded in a radial layout. Coordinated views between cluster tree visualization and parallel coordinates as well as object-space visualizations allow for an interactive analysis of multi-field SPH data (Linsen et al., 2009). The cluster tree allows for the selection of detected clusters, the parallel coordinate plots show the properties of the selected clusters, and object-space visualizations in form of extracted surfaces or particle distributions exhibit the location of the respective clusters in physical space. Figure 18 shows such a visual analysis set-up when applied to the IEEE Visualization Contest data (Rosenthal et al., 2008).
We also proposed a method to integrate the parallel coordinates into the cluster tree visualization. The MultiClusterTree approach (Long & Linsen, 2011) uses circular parallel coordinates for the embedding into the radial hierarchical cluster tree layout, which allows for the analysis of the overall cluster distribution. This visual representation supports the comprehension of the relations between clusters and the original attributes. The combination of the 2D radial layout and the circular parallel coordinates is used to overcome the overplotting problem of parallel coordinates when looking into data sets with many records. Figure 19 shows how integrated circular coordinates can provide a good overview of the cluster distribution.

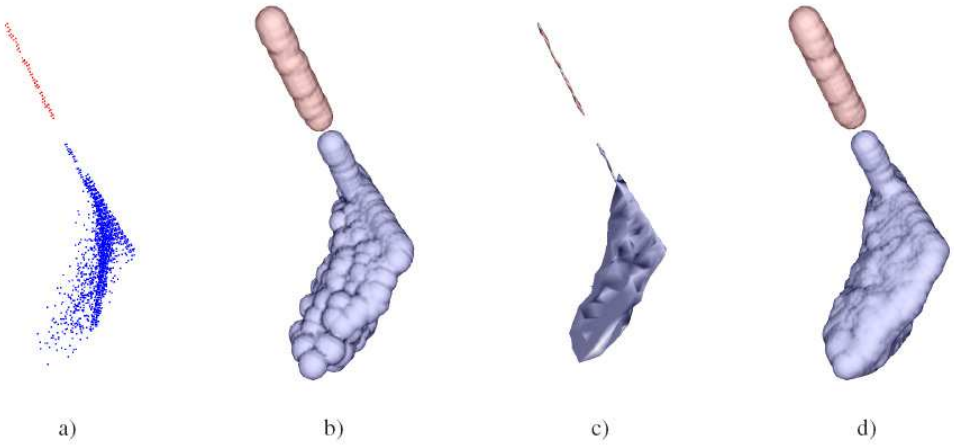


Fig. 17. Different visualizations of two point clusters (colored red and blue) from the 2008 IEEE Visualization Design Contest data. The clusters were found using density-based clustering of multidimensional feature space and were projected to a 3D visual space using a linear projection. Additionally to the cluster points (a), three types of enclosing surfaces are shown. (b) Isosurface extraction from distance field computed using a 3D discrete Voronoi diagram of resolution $256 \times 256 \times 256$. (c) Hull of the cluster computed from the isosurface of the distance field. (d) Isosurface extraction from distance field to hull.

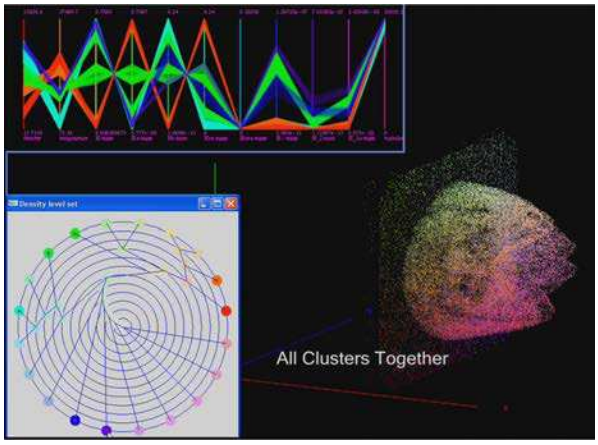


Fig. 18. Coordinated views allow for selecting clusters in cluster tree and investigating properties in attribute space (using parallel coordinates) as well as locations in physical space.

8. Interactive visual system for exploration of multiple scalar and flow fields

Our research results are combined in the SmoothViz software system that is offered to the SPH community via our website (http://vcgl.jacobs-university.de/software). Not all presented features are included yet. Currently, the system consists of three modules responsible

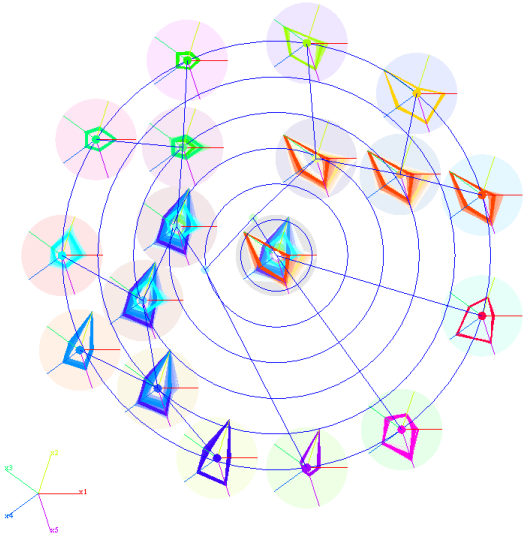


Fig. 19. Integrated circular parallel coordinates in clusters tree visualization for data set with hierarchical clusters.

for time-varying data manipulation, scalar field exploration, and flow field visualization. An intuitive graphical user interface (GUI) allows for easy processing and interaction. Additional functionalities and visualizations that are common in the SPH community have been included.

First, the user can load SPH data containing time-varying particle positions and time-varying multiple scalar and vector field values sampled at the particles. A 3D view of the particle distribution at a chosen time step allows the user to adjust the viewing parameters using arbitrary rotation and translation of camera. Loading of successive or preceding time steps from the time-varying series of data sets is as easy as play or rewind in a standard media player. Extracted pathlines can show evolution in time of an individual particle or sets of particles. Figure 20(a) shows the GUI and a particle distribution plot for a chosen time step.

There are two options to represent the structure of a selected scalar field: Maximal intensity projection plots can render any of the scalar fields using one of the build-in color maps and allowing for manually modifying the transfer function. Figure 20(b) shows the GUI for the transfer function modification and the respective maximum intensity plot of a chosen scalar field. Alternatively, isosurfaces can be extracted for interactively selected isovalues and shown using a point splatting technique or a dense point cloud rendering. Figure 20(c) shows a number of nested isosurfaces using point cloud renderings.

Finally, a specified number of streamlines can be computed with respect to the vector field chosen by the user. Combined views are possible to explore multiple fields simultaneously, e.g. multiple isosurfaces together with stream- or pathlines. Figure 20(d) shows an isosurface rendering using point splatting combined with a rendering of selected streamlines. For more details on the system, we refer to the user manual that comes with the SmoothViz software package.

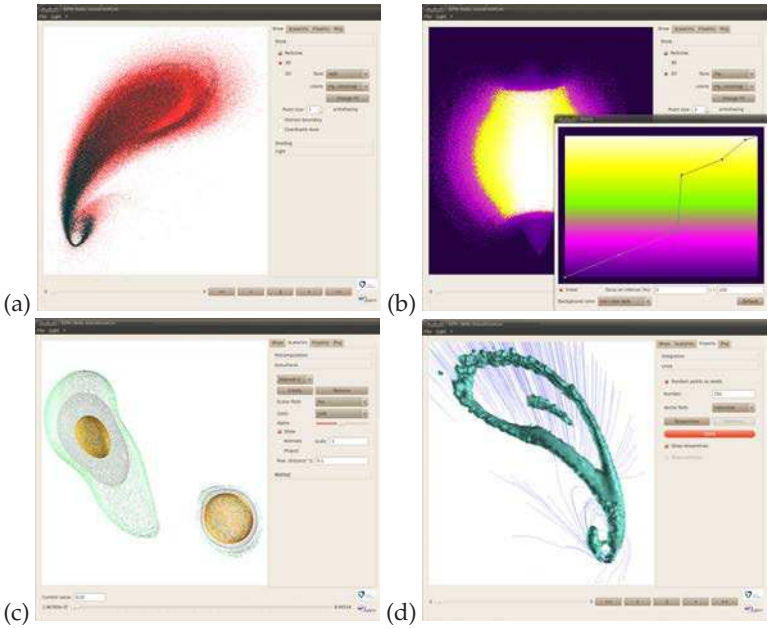


Fig. 20. Screenshots of SmoothViz software system for SPH data exploration: (a) Three-dimensional particle distribution modeling a White Dwarf passing close to a Black Hole. (b) Maximal intensity projection plot of the density field of a White Dwarf with user defined transfer function; (c) Several density isosurfaces of two White Dwarfs in point-based representation. (d) Interplay of a velocity field (shown with streamlines) and a temperature field (shown as splatted isosurface).

9. Conclusion

We have presented approaches for visualization of SPH data. All methods operate directly on the particles that are distributed in a highly adaptive and irregular manner and that do not have any connectivity. Operating on the particles avoids the introduction of errors that occur when resampling to a grid. Our visualizations focus on surface extractions from such data. We first presented an isosurface extraction from any scalar field of the SPH data. It exploits a fast navigation through a *k*d-tree via an indexing structure and allows for fast isosurface extraction of high quality. Because of approximations made during simulation, it is desirable to add a smoothing term to the isosurface extraction method. This is achieved by the use of level-set methods. Again, the method operates on the particles only. We have presented several ways on how to accelerate the computations including a narrow-band approach, a local variational approach, and a signed distance function computation to any isosurface representation. Extracted isosurfaces are given in form of point clouds. We presented how they can be rendered using an image-space point cloud rendering approach that avoids any pre-computation and thus can immediately applied to any extracted surface. Shadows and transparency are supported at interactive rates. We further extended the work to the extraction of boundary surfaces of features in multi-field data. The attribute space of the multi-field data is being explored using clustering and cluster visualization

methods. Coordinated or integrated views to parallel or circular coordinates, respectively, allow for further visual analysis of the properties of the extracted clusters. Coordinated views to object space allow for the investigation of the spatial distribution of detected features. Enclosing surfaces show the cluster boundaries. The presented functionality has partially been incorporated into the SmoothViz software package including further features such as geometric flow visualization. It allows for interactive exploration and integrated analysis of multiple fields of SPH data.

10. Acknowledgments

This work was supported by the German Research Foundation (DFG) under grant number LI 1530/6-1.

11. References

Biddiscombe, J., Graham, D. & Maruzewski, P. (2008). Visualization and analysis of SPH data, *ERCOFTAC Bulletin* 76(9-12).

Borouchaki, H., Hecht, F., Saltel, E. & George, P. (1995). Reasonably efficient delaunay based mesh generator in 3 dimensions, *4th International Meshing Roundtable*, Sandia National Laboratories, pp. 3–14.

CGAL (2011). Computational geometry algorithms library (CGAL), http://www.cgal.org/.

Cha, D., Son, S. & Ihm, I. (2009). Gpu-assisted high quality particle rendering, *Computer Graphics Forum* 28(4): 1247–1255.

Co, C. S. & Joy, K. I. (2005). Isosurface Generation for Large-Scale Scattered Data Visualization, *in* G. Greiner, J. Hornegger, H. Niemann & M. Stamminger (eds), *Proceedings of Vision, Modeling, and Visualization 2005*, Akademische Verlagsgesellschaft Aka GmbH, pp. 233–240.

Courant, R., Friedrichs, K. & Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik, *Mathematische Annalen* 100(1): 32 – 74.

Delaunay, B. N. (1934). Sur la sphere vide, *Bull. Acad. Sci. USSR* 7: 793–800.

Dobrev, P., Rosenthal, P. & Linsen, L. (2010a). An image-space approach to interactive point cloud rendering including shadows and transparency, *Computer Graphics and Geometry* 12(3): 2–25.

Dobrev, P., Rosenthal, P. & Linsen, L. (2010b). Interactive image-space point cloud rendering with transparency and shadows, *in* V. Skala (ed.), *Communication Papers Proceedings of WSCG, The 18th International Conference on Computer Graphics, Visualization and Computer Vision*, UNION Agency – Science Press, Plzen, Czech Republic, pp. 101–108.

Du, Q. & Wang, D. (2006). Recent progress in robust and quality delaunay mesh generation, *J. Comput. Appl. Math.* 195(1): 8–23.

Fraedrich, R., Auer, S. & Westermann, R. (2010). Efficient high-quality volume rendering of sph data, *IEEE Transactions on Visualization and Computer Graphics* 16: 1533–1540.

Fraedrich, R., Schneider, J. & Westermann, R. (2009). Exploring the "millennium run" - scalable rendering of large-scale cosmological datasets, *IEEE Transactions on Visualization and Computer Graphics* 15(6): 1251–1258.

George, P. L., Hecht, F. & Saltel, E. (1991). Automatic mesh generator with specified boundary, *Comput. Methods Appl. Mech. Eng.* 92(3): 269–288.

Gingold, R. A. & Monaghan, J. J. (1977). Smoothed particle hydrodynamics - Theory and application to non-spherical stars, *Monthly Notices of the Royal Astronomical Society* 181: 375–389.

Grossman, J. P. & Dally, W. J. (1998). Point sample rendering, *Proceedings of 9th Eurographics Workshop on Rendering*, pp. 181–192.

Hopf, M. & Ertl, T. (2003). Hierarchical splatting of scattered data, *Visualization Conference, IEEE* .

Hopf, M., Luttenberger, M. & Ertl, T. (2004). Hierarchical splatting of scattered 4d data, *IEEE Computer Graphics and Applications* 24: 64–72.

Linsen, L., Long, T. V. & Rosenthal, P. (2009). Linking multi-dimensional feature space cluster visualization to surface extraction from multi-field volume data, *IEEE Computer Graphics and Applications* 29(3): 85–89.

Linsen, L., Long, T. V., Rosenthal, P. & Rosswog, S. (2008). Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization, *IEEE Transactions on Visualization and Computer Graphics* 14(6): 1483–1490.

Linsen, L., Müller, K. & Rosenthal, P. (2007). Splat-based ray tracing of point clouds, *Journal of WSCG* 15(1–3): 51–58.

Long, T. V. (2010). *Visualizing High Density Clusters in Multidimensional Data*, PhD thesis, Jacobs University.

Long, T. V. & Linsen, L. (2011). Visualizing high density clusters in multidimensional data using optimized star coordinates, *Journal of Computational Statistics (to appear)* .

Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis, *Astronomical Journal* 82: 1013–1024.

Maur, P. & Kolingerová, I. (2001). Post-optimization of delaunay tetrahedrization, *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, IEEE Computer Society, Washington, DC, USA, p. 31.

Molchanov, V., Rosenthal, P. & Linsen, L. (2010). Non-iterative second-order approximation of signed distance function for any isosurface representation, *Computer Graphics Forum* 29(3): 1211–1220.

Molchanov, V., Rosenthal, P. & Linsen, L. (2011). Variational level-set detection of local isosurfaces from unstructured point-based volume data, *Schloss Dagstuhl Scientific Visualization Workshop 2009 Follow-up, to appear*.

Navratil, P. A., Johnson, J. L. & Bromm, V. (2007). Visualization of cosmological particle-based datasets, 13(6): 1712–1718.

Pav, S. E. & Walkington, N. J. (2004). Robust three dimensional delaunay refinement, *13th International Meshing Roundtable*, Sandia National Laboratories, SAND 2004-3765C, pp. 145–156.

Phong, B. T. (1975). Illumination for computer generated pictures, *Commun. ACM* 18: 311–317.

Poco, J., Etemadpour, R., Paulovich, F. V., Long, T. V., Rosenthal, P., de Oliveira, M. C. F., Linsen, L. & Minghim, R. (2011). A framework for exploring multidimensional data with 3d projections, *Computer Graphics Forum* 30(3): 1111–1120.

Price, D. (2007). SPLASH: An interactive visualisation tool for smoothed particle hydrodynamics simulations, *Publications of the Astronomical Society of Australia* 24: 159–173.

Rosenberg, I. D. & Birdwell, K. (2008). Real-time particle isosurface extraction, *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, ACM, New York, NY, USA, pp. 35–43.

Rosenthal, P. (2009). *Direct Surface Extraction from Unstructured Point-based Volume Data*, Shaker Verlag, Aachen, Germany (Ph.D. Thesis Jacobs University, Bremen, Germany).

Rosenthal, P. & Linsen, L. (2006). Direct isosurface extraction from scattered volume data, *in* B. S. Santos, T. Ertl & K. I. Joy (eds), *Eurographics / IEEE VGTC Symposium on Visualization - EuroVis 2006*, pp. 99–106,367.

Rosenthal, P. & Linsen, L. (2008a). Image-space point cloud rendering, *Proceedings of Computer Graphics International*, pp. 136–143.

Rosenthal, P. & Linsen, L. (2008b). Smooth surface extraction from unstructured point-based volume data using PDEs, *IEEE Transactions on Visualization and Computer Graphics* 14(6): 1531–1546.

Rosenthal, P. & Linsen, L. (2009). Enclosing surfaces for point clusters using 3d discrete voronoi diagrams, *Computer Graphics Forum* 28(3): 999–1006.

Rosenthal, P., Long, T. V. & Linsen, L. (2008). "Shadow Clustering": Surface extraction from non-equidistantly sampled multi-field 3D scalar data using multi-dimensional cluster visualization, VisWeek 08 Conference Compendium, Winner of IEEE Visualization Design Contest.

Rosenthal, P., Molchanov, V. & Linsen, L. (2010). A narrow band level set method for surface extraction from unstructured point-based volume data, *in* V. Skala (ed.), *Proceedings of WSCG, The 18th International Conference on Computer Graphics, Visualization and Computer Vision*, UNION Agency – Science Press, Plzen, Czech Republic, pp. 73–80.

Rosenthal, P., Rosswog, S. & Linsen, L. (2007). Direct surface extraction from smoothed particle hydrodynamics simulation data, *Proceedings of the 4th High-End Visualization Workshop*, Lehmanns Media - LOB, pp. 50–61.

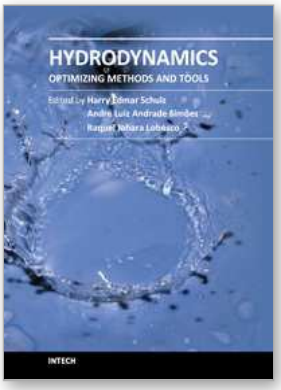
Rosswog, S. (2009). Astrophysical smooth particle hydrodynamics, *New Astronomy Reviews* 53(4-6): 78 – 104.

Rosswog, S., Ramirez-Ruiz, E. & Hix, W. R. (2009). Tidal Disruption and Ignition of White Dwarfs by Moderately Massive Black Holes, *Astrophysical Journal* 695: 404–419.

Sapidis, N. S. & Perucchio, R. (1991). Domain delaunay tetrahedrization of arbitrarily shaped curved polyhedra defined in a solid modeling system, *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, ACM Press, New York, NY, USA, pp. 465–480.

Schindler, B., Fuchs, R., Biddiscombe, J. & Peikert, R. (2009). Predictor-corrector schemes for visualization ofsmoothed particle hydrodynamics data, *IEEE Transactions on Visualization and Computer Graphics* 15: 1243–1250.

Walker, R., Kenny, P. & Miao, J. (2005). Visualization of Smoothed Particle Hydrodynamics for Astrophysics, *in* L. Lever & M. McDerby (eds), *Theory and Practice of Computer Graphics 2005*, Eurographics Association, University of Kent, UK, pp. 133–138. (Electronic version http://diglib.eg.org).
URL: *http://www.cs.kent.ac.uk/pubs/2005/2230*



Hydrodynamics - Optimizing Methods and Tools
Edited by Prof. Harry Schulz

ISBN 978-953-307-712-3
Hard cover, 420 pages
Publisher InTech
Published online 26, October, 2011
Published in print edition October, 2011

The constant evolution of the calculation capacity of the modern computers implies in a permanent effort to adjust the existing numerical codes, or to create new codes following new points of view, aiming to adequately simulate fluid flows and the related transport of physical properties. Additionally, the continuous improving of laboratory devices and equipment, which allow to record and measure fluid flows with a higher degree of details, induces to elaborate specific experiments, in order to shed light in unsolved aspects of the phenomena related to these flows. This volume presents conclusions about different aspects of calculated and observed flows, discussing the tools used in the analyses. It contains eighteen chapters, organized in four sections: 1) Smoothed Spheres, 2) Models and Codes in Fluid Dynamics, 3) Complex Hydraulic Engineering Applications, 4) Hydrodynamics and Heat/Mass Transfer. The chapters present results directed to the optimization of the methods and tools of Hydrodynamics.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821