Direct Surface Extraction from Smoothed Particle Hydrodynamics Simulation Data

Paul Rosenthal[1] and Stephan Rosswog[2] and Lars Linsen[1]

[1]Computational Science and Computer Science,
School of Engineering and Science,
Jacobs University Bremen, Germany

[2]Astrophysics,
School of Engineering and Science,
Jacobs University Bremen, Germany

{p.rosenthal,s.rosswog,l.linsen}@iu-bremen.de

Abstract

Smoothed particle hydrodynamics is a completely mesh-free method to simulate fluid flow. 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. Being completely Lagrangian and following the motion of the flow, these particles represent an unstructured data set at each point in time, i.e. the particles do not exhibit a spatial arrangement nor a fixed connectivity. To visualize the simulated particle data at a certain point in time, we propose a method that extracts surfaces segmenting the domain of the particles with respect to some scalar field.

For scalar volume data, isosurface extraction is a standard visualization method and has been subject to research for decades. We propose a method that directly extracts surfaces from smoothed particle hydrodynamics simulation data without 3D mesh generation or

reconstruction over a structured grid. It is based on spatial domain partitioning using a kd-tree and an indexing scheme for efficient neighbor search.

A geometry extraction step computes points on the surface by linearly interpolating between neighbored pairs of sample points. Its output is a point cloud representation of the surface. The final rendering step uses point-based rendering techniques to visualize the point cloud. A level-set approach can be applied for smoother segmentation results when extracting the geometry of the zero level set.

1 Introduction

Surface extraction is the most commonly used visualization technique for scalar volume data beside direct volume rendering. Common ways to deal with unstructured point-based volume data are to resample the data using scattered data interpolation techniques [Franke & Nielson, 1991], or to compute a grid that connects the unstructured data points [Co & Joy, 2005]. With a recently presented new approach [Rosenthal & Linsen, 2006], it is now possible to extract surfaces directly from unstructured point-based volume data without any resampling or mesh generation.

In smoothed particle hydrodynamics each particle is moving according to the flow equations. Therefore, at each point in time, the simulation data are represented on an unstructured point-based data set. Standard visualization techniques include rendering of the color-coded sample points, e.g. by the commercial software AVS or IDL, extraction of the point set's bounding surface [Vesterlund, 2004], or direct volume rendering, e.g. by a free software package developed by Daniel Price from the University of Exeter, UK. We propose a method that directly extracts a surface that segments the domain of the particles with respect to a given isovalue. For noisy and sparse sampled data sets we introduce a level-set method smoothing the data set to get a better segmentation result.

The isosurface-extraction pipeline is described in Section 3. In Section 4 an introduction to the used level-set method is given. Finally, Section 5 provides a case study applying our methods to smoothed particle hydrodynamics.

2 Smoothed Particle Hydrodynamics

Invented in the astrophysical context [Lucy, 1977, Gingold & Monaghan, 1977], smoothed particle hydrodynamics (SPH) has by now found its way in many different modeling areas, see [Monaghan, 2005] for a recent comprehensive review. SPH is a completely Lagrangian, mesh-free method to solve the equations of fluid dynamics and therefore particularly well suited for simulations in which large deformations occur.

The fluid continuum is represented by moving interpolation centers ("particles"), each of which possesses an interaction range, the so-called smoothing length. The local density is calculated via a kernel weighted sum over neighboring particles within its interaction range, the calculation of pressure gradients involves sums over kernel gradients rather than finite differences. To arrive at the discretized fluid equations nothing more than a Lagrangian, a summation prescription for the density, and the first law of thermodynamics are needed. Therefore, even in their discretized form the equations conserve all physically conserved quantities *by construction*. The particles are advanced in time according to the equation which expresses momentum conservation.

3 Isosurface Extraction

In every point in time, the result of a smoothed particle hydrodynamics simulation is an unstructured point-based volume data set. More precisely, it is a set of trivariate scalar fields $f : \mathbb{R}^3 \to \mathbb{R}$, whose values are given for a large, finite set of sample points $(\mathbf{x}_i, f(\mathbf{x}_i))$, 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. This isosurface extraction is performed in two main steps. First, a set of isopoints $\mathbf{p}_k \in \mathbb{R}^3$ on the isosurface is computed, i.e. $f(\mathbf{p}_k) = v_{\text{iso}}$. In a second step, some kind of neighborhood information for the isopoints is generated. This is subsequently used for rendering the isosurface.

Our idea for the computation of isopoints from the smoothed particle hydrodynamics simulation data is based on linear interpolation between pairs of samples with close positions $\mathbf{x}_i$ and $\mathbf{x}_j$. The inspiration for this

approach is given by isopoint computation using the marching tetrahedra algorithm [Treece et al., 1999] after partitioning the domain via Delaunay tetrahedrization. In this case, the Delaunay triangulation connects natural neighbors defined by the Voronoi diagram.

We want to adopt this isopoint computation method for our purposes while avoiding the expensive computation of the Delaunay tetrahedrization. Thus, we need an approximation of the natural neighbors for each sample position $\mathbf{x}_i$, which will be obtained by using a spatial space decomposition. Simply replacing the natural neighbors by the nearest neighbors would fail in our case of smoothed particle hydrodynamics simulation data due to varying density distributions of the sample point locations.

The kd-tree data structure is known to be a data structure with well-balanced trade-off between flexibility and efficiency [Bentley, 1975] and, what is essentially in our application, with robustness against varying density distribution of sample point positions. To perform a fast exploration of the kd-tree, we introduce an indexing scheme that, beside saving storage space, allows us to determine neighbors using bitwise operations on the index. We determine a small number of potential candidates for our neighbors and reject some of them using an angle and maximum distance criterion.

We compute isopoints respectively by linearly interpolating between two neighboring sample points on different sides of the isosurface. In a final step, we use a recently presented point-based rendering technique [Linsen et al., 2007] using splats to render the isosurface in point cloud representation.

3.1 Data Storage

The n SPH-simulation data points are stored in a three-dimensional kd-tree. Besides the Cartesian coordinates of the points also the function values from the SPH-simulation have to be stored. The tree is stored as a vector of points. Thereon the kd-tree is build recursively splitting the sample point sets of each cell at the median, while cycling through the coordinates. The recursion stops if every cell contains exactly one sample point.

The height of the tree is $\lceil \log_2(n+1) \rceil$. In worst-case ($n = 2^j$), $j \in \mathbb{N}_+$, the number of nodes in the kd-tree is $2n-1$. The nodes of the kd-tree are stored in the vector in breadth-first order, i.e. the root is in position 1 and the children of the node in position j are in positions $2j$ and $2j+1$.

To have fast access to the nodes of the kd-tree a binary indexing scheme is introduced. It allows fast navigation through the tree by using only binary

operations on the binary representation of the nodes positions. Moreover, qualitative propositions about the locations of cells can be made. Thus many informations are implicitly saved in the indexing scheme and will be used for speeding up runtime.

3.2 Neighborhood Computation

The search for all neighbors will be done for all sample points that lie inside a cell of the kd-tree. The neighborhood of a sample point $\mathbf{x}$ contains all cells or splitting planes that have at least one point in common with the cell of $\mathbf{x}$. A two-dimensional illustration of the neighborhood is shown in Figure 1.

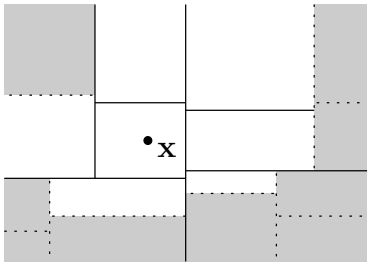

Figure 1: Neighborhood of the sample point $\mathbf{x}$. The bright areas denote the neighboring cells and the solid lines represent the neighboring splitting planes.

As mentioned in Section 3.1 the way of storing the nodes of the kd-tree in the vector constitutes that the position of every node and leaf in the tree is clearly determined by the binary representation of its index. Thus, cells and splitting planes can be identified with their index in the vector.

The computation of the neighborhood for a cell is divided in two main steps. First we compute direct neighbors, i.e. cells and planes that resulted from the last three steps of the tree building process. These neighbors can be directly determined by the index of the observed sample point $\mathbf{x}$.

In the second step we want to get neighbors for the three other faces of the cell of $\mathbf{x}$. The maximum number of these indirect neighbors is in contrast to the direct neighbors not constant. More precisely it is $O(\sqrt[3]{n^2})$ in our case of a three-dimensional kd-tree. But even in the worst case only a small number of cells reach this number of indirect neighbors. Averaged, a cell has at most nine indirect neighbors for each face.

To compute the indirect neighbors, the three splitting planes covering the remaining faces of the cell of $\mathbf{x}$ are directly obtained from the cells index. Subsequently for every found splitting plane a binary search on the opposite side of $\mathbf{x}$ is done and delivers the remaining indirect neighbors.

All in all, the neighborhood computation uses with small exceptions only informations provided by the indexing scheme. Thus, the search for all neighbors uses mostly binary operations on the cells indices and has consequentially a performance that is similar to nearest neighbor search in kd-trees.

3.3 Angle and Maximum Distance Criterion

For our intention, i.e. interpolating isopoints between neighboring sample points, it is very important having neighbors not lying behind other neighbors. One problem occurring in this case is shown in a two-dimensional example in Figure 2.

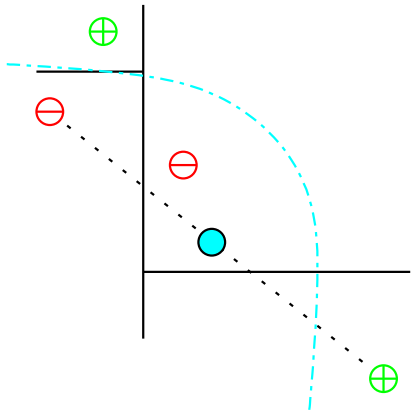

Figure 2: Example of a bad isopoint interpolation caused by the small angle between the two negative sample points and the positive sample point in the lower right cell. The desired isocontour is drawn in light blue. Note that the interpolated isopoint is farther away from the isoline than each sample point.

To avoid such situations we establish an angle criterion extending the angle criterion method Linsen and Prautzsch [Linsen & Prautzsch, 2001] used for point-based surface representations to volume data. It limits the maximum angle between neighbors to one sample point. If the criterion is violated by a pair of neighbors, the farther neighbor is omitted from the neighborhood.

A second criterion limits the maximum distance between neighbors. This criterion arises from the fact, that errors of linear interpolation can grow enormously with increasing distance. If a neighbor's distance to the observed point exceeds the distance threshold it will also be omitted from the neighborhood.

4 Level-Set Segmentation

Due to varying particle number densities of the SPH data the direct isosurface extraction step can produce rough isosurfaces in sparse areas. To generate smoother segmentations, we propose the application of a level-set segmentation method to the data prior to isosurface extraction.

The basic idea of level-set methods goes back to Osher and Sethian [Osher & Sethian, 1988], who first described the evolution of a closed hypersurface. In general a so called level-set function φ is first initialized on the sample points. Subsequently this level-set function is successively adapted to the data set f. For this process a variety of approaches exist. A detailed overview of this field of research is given by Osher and Fedkiw [Osher & Fedkiw, 2003].

For our purposes, i.e. the smooth segmentation of SPH data, we used the level-set process equation

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

which models hyperbolic normal advection, weighted with factor $a > 0$, and mean curvature flow, weighted with factor $b > 0$. This means that the level-set function moves towards the data set with the factor a and the mean curvature of the resulting isosurface is smoothed with the factor b.

This level-set process is done using a forward Euler time integration until the level-set function does not move any more. The needed derivatives are approximated using a least-squares approach. Subsequently the zero level set is extracted using direct isosurface extraction, as described in Section 3.

5 Case Study: White Dwarf Black Hole Encounters

Our Galaxy, the Milky Way, is surrounded by a halo of more than 150 globular clusters, spherical agglomerations of typically 100.000 stars. The very large stellar number densities in their centers are believed to produce via stellar collisions black holes between several hundred and several thousand solar masses, so-called "intermediate mass black holes" (intermediate between stellar mass black holes that result from a supernova explosion and the supermassive black holes that are observed in the centers of galaxies).

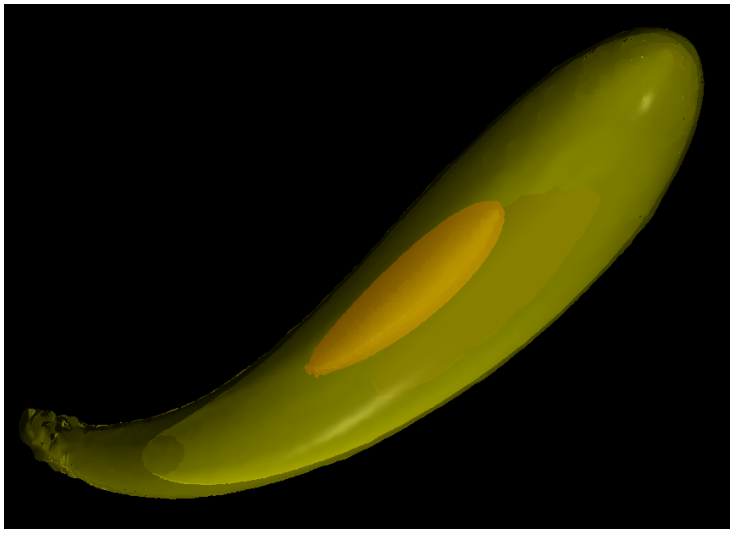

Figure 3: Rendering of two surfaces segmenting the white dwarf data set with respect to the density. The transparent yellow surface represents the isosurface to the isovalue $2{\cdot}10^4 \mathrm{g/cm}^3$, whereas the red surface is the isosurface to the isovalue $2 \cdot 10^5 \mathrm{g/cm}^3$.

Being old, globular clusters also contain large numbers of the remnants of solar-type stars, so-called white dwarfs. We simulated the fly-by of white dwarfs close to such an intermediate mass black hole [Rosswog and Ramirez-Ruiz 2007, to be submitted]. In these simulations the white dwarf was modeled with up to 7.000.000 SPH particles. During the fly-by the white dwarfs become very strongly compressed and the resulting temperature increase triggers the thermonuclear ignition of the white dwarf material. Thus for each

SPH particle the evolution of the nuclear abundances and the resulting energy release is calculated via a nuclear reaction network [Hix et al., 1998]. Since during the compression the required nuclear reaction time steps are smaller than the hydrodynamic time steps by several orders of magnitude, the hydrodynamics and the nuclear reactions are calculated by two different time-integration schemes in an operator splitting fashion.

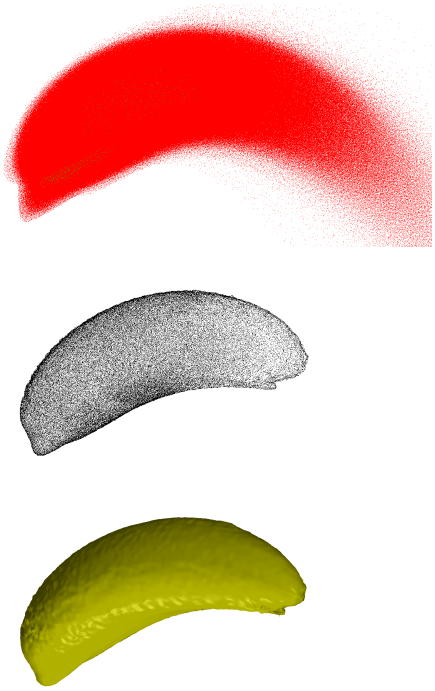

Figure 4: Illustration of the surface extraction pipeline on a later point in time of the white dwarf simulation. In the topmost picture, the seven million SPH data points are shown. The middle picture shows the extracted isopoints. The extracted isosurface is rendered in the lowermost picture.

Such a simulation usually produces of the order one thousand data dumps, where each dump contains more than one Gbyte of physical data represented on the SPH particles. To understand and analyze such simulations it is of uttermost importance to be able to visualize the physical variables in a 3D rendering like the one shown in Figure 3. As the outcome of such simulations is a priori unknown, there are no "standard tasks" that can be routinely performed on the data sets. Therefore, the key to an efficient

scientific production process is a very rapid and flexible 3D visualization tool that allows to interactively explore the large data sets.

Figure 5: Comparison between direct isosurface extraction on the left side and level-set segmentation on the right side for a white dwarf simulation with 500.000 particles. To illustrate the significant advantage of the level-set approach, we show a rendering of the surface points with very small splats.

For every data set with seven million particles, the presented method takes ten seconds to build the kd-tree and another eight seconds to generate the neighborhood informations. Subsequently every isosurface-point-cloud extraction lasts eight seconds, i.e. no recalculation of the kd-tree or neighborhood is needed. The extracted 3D point cloud can be explored interactively. If a relevant isosurface is found, it can be rendered in at most one minute. An illustration of the surface-extraction pipeline is shown in Figure 4.

A comparison between isosurface extraction and the level-set approach on a sparse sampled 500.000 particles white dwarf data set is shown in Figure 5.

6 Conclusion

We presented a visualization method for smoothed particle hydrodynamics simulation data. To visualize a scalar field associated to the particles, we directly extract a surface segmenting the domain of the particles with respect to a given isovalue without any 3D mesh generation or reconstruction over a structured grid. For this purpose we use a spatial domain partitioning using a kd-tree and an indexing scheme for efficient neighbor search. Between appropriate neighbors, surface points are computed using linear interpolation

with respect to the isovalue. To improve the smoothness of the extracted surface a level-set method is applied to the data prior to surface extraction. The resulting point cloud is visualized using point-based rendering techniques.

The presented application pipeline was tested on a case study dealing with the simulation of a white dwarf passing a black hole. After a short preprocessing the extracted isopoint clouds can be explored interactively, whereas the isovalue or the physical quantity can be changed in nearly interactive time. This fast and interactive data visualization approach allows for an efficient extraction of the physical information contained in the simulation data and substantially speeds up the otherwise cumbersome analysis process.

References

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9), 509–517.

[Co & Joy, 2005] 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* (pp. 233–240).: Akademische Verlagsgesellschaft Aka GmbH.

[Franke & Nielson, 1991] Franke, R. & Nielson, G. M. (1991). *Geometric Modeling: Methods and Applications*, chapter Scattered Data Interpolation: A Tutorial and Survey, (pp. 131–160). Springer Verlag, New York.

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

[Hix et al., 1998] Hix, W. R., Khokhlov, A. M., Wheeler, J. C., & Thielemann, F.-K. (1998). The Quasi-Equilibrium-reduced alpha -Network. *Astrophysical Journal*, 503, 332–+.

[Linsen et al., 2007] Linsen, L., Müller, K., & Rosenthal, P. (2007). Splat-based ray tracing of point clouds. *Journal of WSCG*, 15(1–3).

[Linsen & Prautzsch, 2001] Linsen, L. & Prautzsch, H. (2001). Global versus local triangulations. In J. Roberts (Ed.), *Proceedings of Eurographics 2001, Short Presentations*.

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

[Monaghan, 2005] Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports of Progress in Physics*, 68, 1703–1759.

[Osher & Fedkiw, 2003] Osher, S. & Fedkiw, R. (2003). *Level set methods and dynamic implicit surfaces*. Springer.

[Osher & Sethian, 1988] Osher, S. & Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formualtions. *Journal of Computational Physics*, (79), 12–49.

[Rosenthal & Linsen, 2006] Rosenthal, P. & Linsen, L. (2006). Direct isosurface extraction from scattered volume data. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization* (pp. 99–106).

[Treece et al., 1999] Treece, G. M., Prager, R. W., & Gee, A. H. (1999). Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4), 583–598.

[Vesterlund, 2004] Vesterlund, M. (2004). Simulation and rendering of a viscous fluid using smoothed particle hydrodynamics. Master's thesis, Umea University, Sweden.

Index

angle criterion, 6

cell, 4

direct neighbors, 5

indexing scheme, 4
indirect neighbors, 5
isopoint, 3
isosurface, 3

kd-tree, 4
kernel, 3

level-set segmentation, 7

maximum distance criterion, 6

neighborhood, 5

particle, 3
point-based rendering, 4

smoothed particle hydrodynamics, 3
smoothing length, 3
splats, 4

unstructured point-based volume data, 3