Smooth Surface Extraction from Unstructured Point-based Volume Data Using PDEs

Paul Rosenthal and Lars Linsen

Abstract—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 propose an approach that directly extracts smooth surfaces from unstructured point-based volume data without prior resampling or mesh generation. When operating on unstructured data one needs to quickly derive neighborhood information. The respective information is retrieved by partitioning the 3D domain into cells using a kd-tree and operating on its cells. We exploit neighborhood information to estimate gradients and mean curvature at every sample point using a four-dimensional least-squares fitting approach. Gradients and mean curvature are required for applying the chosen PDE-based method that combines hyperbolic advection to an isovalue of a given scalar field and mean curvature flow. Since we are using an explicit time-integration scheme, time steps and neighbor locations are bounded to ensure convergence of the process. To avoid small global time steps, one can use asynchronous local integration.
We extract a smooth surface by successively fitting a smooth auxiliary function to the data set. This auxiliary 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 these informations the auxiliary 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 auxiliary function gradient in a sample exceeds a given threshold at some time, the auxiliary function is reinitialized to a signed distance function. After convergence of the evolvution, the resulting smooth surface is obtained by extracting the zero isosurface from the auxiliary function using direct isosurface extraction from unstructured point-based volume data and rendering the extracted surface using point-based rendering methods.

Index Terms—PDEs, surface extraction, level sets, point-based visualization.

1 Introduction

With the permanently improving digital technologies, the amount of generated, collected, and stored data increases steadily. While costs for computation power, data storage, and data collection decline, more and more data has to be evaluated. When operating on cluster or shared-memory machines, numerical simulations of physical phenomena can produce huge data sets. For flexibility, some of these simulations are preferred to be carried out on an unstructured point-based data structure rather than a grid. For example, in astrophysics particle simulations are quite frequently used with the number of particles ranging up to several millions, each storing multiple scalar values. Another example of large unstructured point-based volume data generation are the deployment of sensor systems with large numbers of sensors.

An additional attractive property of unstructured point-based data approaches is that they naturally include all grid-based configurations. An unstructured point-based data visualization approach can always be applied to a gridded data set by neglecting the grid connectivity.

Surface extraction methods based on partial differential equations (PDEs) have a large variety of applications and, in particular, are a well-known technique for segmentation of scalar volume data. Many algorithms and approaches with different modifications of the main idea exist. Most of them address a specific problem or a specific type of gridded data sets. Typically the algorithms operate on hexahedral cells and a given initial surface is modified to explicitly or implicitly minimize a given energy functional.

Even though all these known algorithms cover a wide area of problems, to our knowledge no algorithm exists that directly operates on unstructured point-based volume data, where scalar function values are given at points in a three-dimensional domain that have an arbitrary distribution and no grid connectivity. If this type of data has to be processed, it is typically resampled over a regular structured grid using scattered data interpolation techniques. Such an interpolation technique introduces resampling inaccuracies that increase the uncertainty or error in the resulting visualization.

We propose a PDE-based surface extraction method that directly operates on a large unstructured point-based volume data set, i. e. we are neither resampling the data over a structured grid nor generating a global or local polyhedrization. Instead, we determine some neighborhood information for the sample points, initialize an auxiliary function on the sample points, and process this function according to its approximated and interpolated properties like gradient or mean curvature. For our surface extraction method we use hyperbolic advection to an isovalue of a given scalar field and mean curvature flow. In the context of level-set methods, the auxiliary function is typically referred to as level-set function.

To ensure numerical stability of the chosen explicit time integration scheme and thus convergence of the overall evolution process we choose appropriate neighbors for the derivative calculation and small enough time steps. The drawback of small global time steps can be circumvented by using asynchronous local integration. After convergence of the process, the zero isosurface to the auxiliary function is extracted in form of a point cloud surface representation and visualized via point-based rendering techniques using splats.

The main ideas of our smooth surface extraction method and the structure of the entire pipeline are introduced in Section 3. For storing the sample points we use a spatial decomposition based on a three-dimensional kd-tree, as described in Section 4. The approximation of gradient and mean curvature requires the computation of nearest neighbor informations for each sample point. These are computed in a preprocessing step and stored during the whole evolution process. In every step and for every sample point the gradient of the auxiliary function and the mean curvature is approximated using a four-dimensional least-squares method, as described in Section 5.

- *Paul Rosenthal is with Jacobs University Bremen, E-mail: p.rosenthal@jacobs-university.de.*
- *Lars Linsen is with Jacobs University Bremen, E-mail: l.linsen@jacobs-university.de.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

While the reinitialization of the auxiliary function is described in Section 6, the evolution approach is explained in detail in Section 7. Considerations about stability and time integration are presented in Section 8. Section 9 explains the last step in our pipeline, the zero isosurface extraction. Theoretical and practical results, including analysis of error and computation times, are shown in Section 10.

2 Related Work

Common ways to deal with unstructured point-based volume data are to resample the data using scattered data interpolation techniques or to compute a polyhedral grid that connects the unstructured data points. Afterwards segmentation methods like isosurface extraction, region-growing methods, and also PDE-based methods like level-sets or finite elements can be applied to the gridded data to generate the desired visualizations. Level-set methods, in particular, tend to operate on regular hexahedral grids what facilitates discrete derivative computations.

Scattered data interpolation is a well-studied field. We refer to one of many surveys on this topic for further details [10]. Recently, Park et al. [25, 26] have shown that scattered data reconstruction for large data sets can be achieved at interactive or near-interactive rates when discretizing the approach and resampling over a regular grid. Unfortunately, such resampling steps always introduce inaccuracies. In our case, when observing data sets with highly varying point density, the interpolation error would be enormous when using regular grids that fit today's memory constraints. Adaptive grids can reduce the error, but the more adaptive it gets the more complicated the processing becomes, which raises the desire to directly operate on the unstructured point-based volume data.

Level-set methods go back to Sethian and Osher [22, 23, 30], who first described the evolution of a closed hypersurface. As one can see, the general approach is younger than classical surface extraction and scalar volume data processing techniques. Nevertheless, due to their flexibility already a large community is using and developing level-set methods. Many different approaches exist and the range of application areas is wide. Still, to our knowledge, all existing level-set algorithms require some kind of underlying grid and connectivity information among the sample points.

Breen et al. [2] presented a general framework for level-set segmentation of a large variety of regular data sets. Museth et al. [20] use a level-set method to segment non-uniform data sets, and Enright et al. [5] apply a level-set approach to an octree-based adaptive mesh. Beside these, many other approaches, e.g. [18, 31, 33], exist that target a huge variety of level-set segmentation tasks on different structured data sets. We want to explicitly mention the particle level-set methods [13], as they use free particles during the level-set computations. However, in contrast to our approach, particle level-set methods still require an underlying grid or mesh. We present an approach that operates directly on unstructured point-based volume data.

Some level-set approaches such as active contours [19] require an explicit representation of the initial surface, which is deformed until it converges to the desired surface. Our approach, instead, changes the scalar values in the sample points over time such that the surfaces are giving implicitly. In particular, we do not need to reconstruct the scalar field at any positions other than the sample points.

3 General Approach

Given a set of unstructured sample points $\mathbf{x}_i \in D \subset \mathbb{R}^3$ in space and a volumetric scalar field $f : D \to \mathbb{R}$ with bounded domain D that is given at these sample locations, we want to extract a smooth isosurface $\Gamma_{\text{iso}} \subset D$ with respect to a given isovalue f_{iso}.

We first need to build a data structure that stores and handles neighborhood information. This is done using a three-dimensional kd-tree. In a preprocessing step, we apply a standard algorithm [11] on kd-trees for computing the n nearest neighbors for each sample. We chose $n = 26$ inspired by the regular case on a structured equidistant hexahedral grid, where every sample $\mathbf{x}$ has 26 nearest neighbors in the L_∞-metric.

Afterwards an auxiliary function φ is initialized as a signed distance function for every sample. We choose a radial function as initial function such that the initial isosurfaces are spheres. The auxiliary function is adapted in an iterative process steered by the PDE. If the norms of the auxiliary function gradients exceed a given threshold at any time in the process, a reinitialization step is applied to this function. This keeps the auxiliary function close to a signed distance function, where distances are measured with respect to the isosurface, and assures good numerical behavior of the function during the PDE-based process.

During the adapting process, the gradient $\nabla\varphi$ of the auxiliary function φ and the mean curvature κ_φ are approximated only for the sample points. The approximation is computed using a four-dimensional least-squares approach. Having all necessary informations collected, a time step is performed with respect to the equation

$$\frac{\partial \varphi}{\partial t} = \left(a\,(f - f_{\text{iso}} - \varphi) + b\kappa_\varphi\right)|\nabla\varphi| \;, \tag{1}$$

which models hyperbolic normal advection [22], weighted with factor $a > 0$, and mean curvature flow, weighted with factor $b > 0$.

Since we are using an explicit Euler time discretization for updating the function at the sample points, the time steps are bounded by the Courant-Friedrichs-Lewy condition [3] to permit numerical stability. One can follow a global or a local strategy for updating the level-set function values.

If one chooses one global time step for all sample points, the step is bounded by the most restrictive stability condition of the sample points. The required function properties are fast to compute, but for most samples the time steps are smaller than required.

Instead, one can use adaptive time integration [21, 24]. Here, one has a local time step for every sample. Thus, the stability condition of a sample does not affect the time steps of other sample points. Since calculations per sample point are more complicated and time consuming this method only pays off for data sets with highly varying point density.

In both cases, the auxiliary function is deformed at the sample points over time until it reaches steady state, i. e., until the function values do not change more than a given threshold from one time step to the subsequent one. After convergence the isosurface to the isovalue zero is extracted from the generated scalar field over the sample points. This surface has then the desired properties of Γ_{iso}. Note that the surface does not need to be extracted during the PDE computations. It is given implicitly.

4 Scattered Data Storage

For fast calculations and access to the sample points some preprocessing steps are required. For each sample we calculate the 26 nearest neighbors with respect to the Euclidean distance. This neighbor information is used later on to approximate geometric properties of the auxiliary function.

To store the scattered data points including space coordinates, data function value, and auxiliary function value we use a three-dimensional kd-tree T. This data structure helps saving computation time. In order to save storage space the samples are not directly stored in the kd-tree but in a vector of sample point locations P and in a vector of values V. Vector V stores all the function values. The nodes of the kd-tree only link to P and V.

Using vector P, the kd-tree T is build recursively. We start with depth 0 and $T = \emptyset$ and corresponding vector P. Then, for every depth i in the kd-tree and vector P', which is a subvector of P, we sort P' in $x^{i \bmod 3}$-direction, where x_0, x_1 and x_2 denote the three dimensions. A node is inserted into T, P' is split in two half-sized vectors P'_1 and P'_2 and the pivot-value of the splitting is stored in the inserted node. If P' is odd-sized, a link to the midpoint is also stored in the node. While proceeding recursively with the subvectors P'_1 and P'_2, we get two children for the just inserted knot. If a subvector is empty the recursion stops.

5 Calculating Function Properties

To process the auxiliary function following hyperbolic normal advection and mean curvature flow, as modeled in Equation (1), we have to

calculate the gradient and mean curvature in each sample point and for every time step.

Because of the unstructured distribution of the samples, we are not able to use any grid-based approach for calculating function gradients or mean curvature. Instead we use a four-dimensional least-squares approach to approximate the gradient $\nabla\varphi$ of the auxiliary function φ and the mean curvature κ_φ. During the calculations only a nearest-neighbors computation of the sample points is needed. Furthermore, no information about the scalar field other than at the sample points is needed.

5.1 Gradient Calculation

Let $\varphi : \mathbb{R}^3 \supset D \to \mathbb{R}$ be a differentiable function, then the graph of φ is the submanifold $\text{graph}(\varphi) \subset \mathbb{R}^4$ defined as

$$\text{graph}(\varphi) := \{(\mathbf{x}, \varphi(\mathbf{x})) \,:\, \mathbf{x} \in D\} \,.$$

The gradient $\nabla\varphi(\mathbf{x})$ in every point $\mathbf{x}$ can be derived by projecting the normal $\mathbf{n}(\mathbf{x}, \varphi(\mathbf{x}))$ of the tangent hyperplane to $\text{graph}(\varphi)$ in $\mathbf{x}$ to the volumetric domain. We obtain that

$$\nabla\varphi(\mathbf{x}) = -\text{pr}_{\mathbb{R}^3}(\mathbf{n}(\mathbf{x}, \varphi(\mathbf{x}))) \,,$$

where $\text{pr}_{\mathbb{R}^3} : \mathbb{R}^4 \to \mathbb{R}^3$ denotes the orthogonal projection to the three first coordinates. One- and two-dimensional illustrations of this geometrical consideration are presented in Figures 1 and 2.

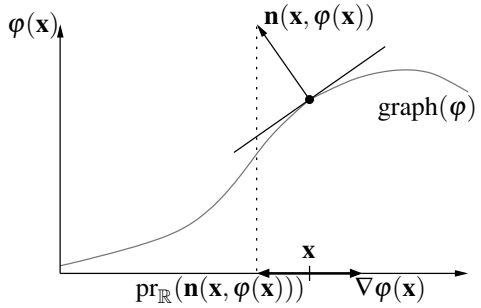

Fig. 1. Relation between the normal to the graph and gradient of a one-dimensional scalar function φ.

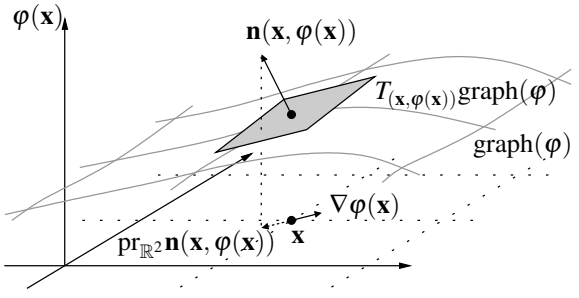

Fig. 2. Graph of a two-dimensional function φ with tangent plane $T_{(\mathbf{x},\varphi(\mathbf{x}))}\text{graph}(\varphi)$, normal $\mathbf{n}(\mathbf{x}, \varphi(\mathbf{x}))$ to the graph, and gradient $\nabla\varphi(\mathbf{x})$.

Considering the nearest neighbors of $\mathbf{x}$ in $\mathbb{R}^3$, the tangent hyperplane $T_{(\mathbf{x},\varphi(\mathbf{x}))}\text{graph}(\varphi) \subset \mathbb{R}^4$ is approximated using a four-dimensional least-squares fitting through the neighboring samples with associated function values. The normal of the resulting hyperplane is projected to $\mathbb{R}^3$ to get the function gradient.

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

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

This formula is a generalization of several well-known gradient approximation methods. For example, using points (x, φ_1) and $(x+h, \varphi_2)$ leads to the standard forward differencing

$$\frac{\mathrm{d}\varphi}{\mathrm{d}x} = \frac{\varphi_2 - \varphi_1}{h} \,.$$

Also central differencing is a special case of Equation (2). Using points (x, φ_1), $(x+h, \varphi_2)$, and $(x-h, \varphi_0)$ leads to

$$\frac{\mathrm{d}\varphi}{\mathrm{d}x} = \frac{\varphi_2 - \varphi_0}{2h} \,.$$

Such a closed formula can also be derived for approximations of gradients in higher dimensions. It turns out, that the chosen least-squares approach also generalizes the known forward and central differencing schemes on grids in higher dimensions. The formulae with the closed forms for our application in 3D are given in the appendix.

This least-squares approach results in a consistent gradient approximation: If the distance to the used neighbors converges to 0, the computed hyperplane converges to the tangent hyperplane to the graph of the function. Hence, the approximated gradient converges towards the exact gradient.

5.2 Mean Curvature Calculation

The calculation of the mean curvature κ_φ requires some more considerations. We follow the ideas described by Osher and Fedkiw [22]. First, we note that

$$\kappa_\varphi = \nabla \cdot \frac{\nabla\varphi}{|\nabla\varphi|} \,.$$

More precisely, we have

$$\kappa_\varphi = \left(\nabla\frac{(\nabla\varphi)_1}{|\nabla\varphi|}\right)_1 + \left(\nabla\frac{(\nabla\varphi)_2}{|\nabla\varphi|}\right)_2 + \left(\nabla\frac{(\nabla\varphi)_3}{|\nabla\varphi|}\right)_3$$

with

$$\frac{\nabla\varphi}{|\nabla\varphi|} = \left(\frac{(\nabla\varphi)_1}{|\nabla\varphi|}, \frac{(\nabla\varphi)_2}{|\nabla\varphi|}, \frac{(\nabla\varphi)_3}{|\nabla\varphi|}\right) \,.$$

Thus, we can reduce the problem of mean curvature calculation to the problem of three gradient calculations for the three dimensions of the normalized gradient of φ.

The normalized gradient of φ can be approximated for every sample, as described in Section 5.1. Then, the gradient to every function

$$\frac{(\nabla\varphi)_i}{|\nabla\varphi|} : \mathbb{R}^3 \to \mathbb{R} \,, \quad i = 1,2,3 \,,$$

is approximated using again a four-dimensional least-squares approach. Afterwards the ith components of gradients $\nabla\frac{(\nabla\varphi)_i}{|\nabla\varphi|}$, $i = 1,2,3$, are taken and summed up to get an approximation to the mean curvature κ_φ. Due to using only multiple consistent gradient calculations, the consistency of this approach for mean curvature approximation is obvious.

6 Reinitialization

As described by Peng et al. [28], the quality of the PDE-process significantly degrades if the auxiliary function φ is not close to a signed distance function of the isosurface. In our approach the function is initialized as a signed distance function to the isosurface. Unfortunately the PDE process cannot maintain this property.

Hence, we have to reinitialize the auxiliary function to a signed distance function to the new isosurface if the norms of the gradients of the function exceed a certain threshold. For this reinitialization we choose a PDE approach solving the special Eikonal equation

$$\frac{\partial \varphi}{\partial t} = \text{sign}(\varphi)(1 - |\nabla \varphi|) \,. \tag{3}$$

This approach is much faster than exact calculation of the signed distance function. As we use an explicit Euler time integration for the reinitialization process, Equation (3) leads to the time development equation

$$\varphi^{i+1} = \varphi^i + \Delta t \cdot \text{sign}(\varphi^i)(1 - |\nabla \varphi^i|) \,.$$

7 Smooth Isosurface Extraction

As described in Section 3, our goal is to extract a smooth isosurface $\Gamma_{\text{iso}} \subset D \subset \mathbb{R}^3$ with respect to an isovalue f_{iso} of $f : \mathbb{R}^3 \supset D \to \mathbb{R}$.

For solving this problem we choose a PDE formulation which is a combination of two well-known approaches. The hyperbolic advection [22]

$$\frac{\partial \varphi}{\partial t} + \alpha |\nabla \varphi| = 0$$

models the transport of the interface in normal direction with speed α. We want to extract an isosurface from the given data set, so our goal is to minimize

$$E = \int_D |\varphi - (f - f_{\text{iso}})| \mathrm{d}\mathbf{x} \,.$$

This leads to the evolution equation

$$\frac{\partial \varphi}{\partial t} + (\varphi - (f - f_{\text{iso}}))|\nabla \varphi| = 0 \,. \tag{4}$$

A second property we want to consider is the smoothness of the resulting surface Γ_{iso}. Thus, our second goal should be the minimization of the surface area of Γ, i. e., we want to minimize

$$|\Gamma| = \int_D \delta(\varphi) |\nabla \varphi| \mathrm{d}\mathbf{x} \,,$$

where δ denotes the one-dimensional Dirac δ-function with

$$\delta(x) = \frac{\mathrm{d}}{\mathrm{d}x} H(x)$$

and H being the one-dimensional Heaviside function, defined as

$$H(x) = \begin{cases} 1 & \text{for} \quad x \geq 0 \\ 0 & \text{for} \quad x < 0 \end{cases} \,.$$

These considerations lead to the model of mean curvature flow [6, 7, 8, 9], characterized by

$$\frac{\partial \varphi}{\partial t} = \kappa_\varphi |\nabla \varphi| \,, \tag{5}$$

where κ_φ denotes the mean curvature to the level set.

Combining hyperbolic normal advection (4) and mean curvature flow (5), we get the evolution equation

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

with scaling parameters $a, b > 0$, where gradient $\nabla \varphi$ and mean curvature κ_φ are computed as described in Section 5. This equation is solved to steady state using

$$\varphi^{i+1} = \varphi^i + \Delta t \cdot \frac{\partial \varphi^i}{\partial t} \,, \tag{6}$$

i. e. an explicit time discretization of order one.

8 Stability and Time Integration

8.1 Stability

According to the Lax-Richtmyer equivalence theorem [32] convergence of a finite difference scheme is equivalent to consistency and stability. As mentioned in Section 5.1 our gradient and mean curvature approximations are consistent. The stability of our differential schemes and following constraints to the time step sizes will be now observed.

For simplicity, we describe our analysis for the two-dimensional hyperbolic normal advection case, i. e.

$$\frac{\partial \varphi}{\partial t} = a|\nabla \varphi| \,, \quad a > 0 \,.$$

Applying our finite difference scheme, derived in Section 5.1, and explicit Euler time integration we get an evolution equation

$$\varphi^{i+1}(\mathbf{x}) = (\mathbf{E}(\Delta t)\varphi^i)(\mathbf{x}) \,, \tag{7}$$

for each sample $\mathbf{x} \in \mathbb{R}^2$ with discrete solution operator $\mathbf{E}$. We use the von Neumann stability analysis [14, 34], which makes use of spatial Fourier transforms [1, 4], to verify stability of the time evolution Equation 7. Von Neumann's theorem states that the evolution process is stable with respect to the maximum norm, iff

$$\left|\tilde{\mathbf{E}}(\xi)\right| \leq 1 \,, \quad \text{for all} \quad \xi \in \mathbb{R} \,,$$

where $\tilde{\mathbf{E}}$ denotes the Fourier transform of $\mathbf{E}$.

For observing two neighbors per sample point (x, y) for the gradient approximation we can assume the situation (x, y, f_1), $(x+1, y, f_2)$, $(x + \Delta x, y + \Delta y, f_3)$ without loss of generality. In this case, von Neumann stability analysis leads to the following conditions assuring stability

$$\lambda_y - \Delta y \frac{\lambda_x}{\Delta x} \leq \frac{1}{\Delta t} \tag{8}$$

$$\left((1 - \Delta y)\frac{\lambda_x}{\Delta x} + \lambda_y\right) \leq \frac{1}{\Delta t} \tag{9}$$

$$\Delta y \frac{\lambda_x}{\Delta x} \leq \lambda_y \tag{10}$$

$$\frac{\lambda_x}{\Delta x} \geq 0 \,, \tag{11}$$

where $\lambda_x = \frac{a}{|\nabla f|}\frac{\partial \varphi}{\partial x}$ and $\lambda_y = \frac{a}{|\nabla f|}\frac{\partial \varphi}{\partial y}$. One easily sees, that Equations (8) and (9) can be maintained by choosing a small enough time step Δt. Contrary to this, the last two conditions affect the relation between partial derivatives of φ and Δx, Δy. From a descriptive point of view, they assure the upwinding of the process and describe two sectors in which the second neighbor is allowed to be to assure stability.

In our approach, where the domain is $\mathbb{R}^3$, the same considerations lead to similar conditions and restrictions for the time step and sample locations. Therefore, we are able to process the auxiliary function assuring stability by choosing appropriate neighbors and a small enough time step.

In practice, we observed that when choosing the 26 nearest neighbors and small enough time steps Δt in Equation 6, we never ran into stability problems. To derive an estimate for a time step Δt we transferred the considerations by Osher and Fedkiw [22] to our case. The condition for the hyperbolic normal advection becomes

$$\Delta t \frac{a|f - f_{\text{iso}} - \varphi|}{d_{\min}|\nabla \varphi|} \left(\left|\frac{\partial \varphi}{\partial x_1}\right| + \left|\frac{\partial \varphi}{\partial x_2}\right| + \left|\frac{\partial \varphi}{\partial x_3}\right|\right) < 1 \,,$$

where $d_{\min}$ denotes the Euclidian distance to the nearest neighbor, i. e. the radius of the minimal domain of dependence. The condition for the whole evolution becomes

$$\Delta t \left(\frac{a|f - f_{\text{iso}} - \varphi|}{d_{\min}|\nabla \varphi|} \left(\left|\frac{\partial \varphi}{\partial x_1}\right| + \left|\frac{\partial \varphi}{\partial x_2}\right| + \left|\frac{\partial \varphi}{\partial x_3}\right|\right) + \frac{6b}{d_{\min}^2}\right) < 1 \,.$$

Although there is no evidence that this criterion ensures stability, it worked out well for all the practical cases we considered. While from a theoretical point of view, we can ensure stability by choosing appropriate neighbors and time steps with respect to Equations (8) to (11), for practical purposes it is beneficial to use a larger number of neighbors, as larger number of neighbors (26 in our examples) ensure better gradient approximations.

8.2 Synchronous Time Integration

Time steps have to be small enough to ensure stability, i. e., the physical domain of dependence is required to lie in the domain of dependence of the finite difference scheme. All these considerations are done in a fixed sample point at a certain point in time. If we want to apply a global time step for all sample points, it is bounded by the most restrictive stability condition when considering all points. Thus, if the sample points have a highly varying distribution or if the underlying scalar field has big local variations, time steps for all points may be bounded by the rather restrictive stability condition of a few points.

8.3 Asynchronous Time Integration

To alleviate this potential drawback, one can use asynchronous time integration. Here, like in global time integration, all sample points start at the same point in time. Then for each sample point one time step is computed, only bounded by the local stability condition.

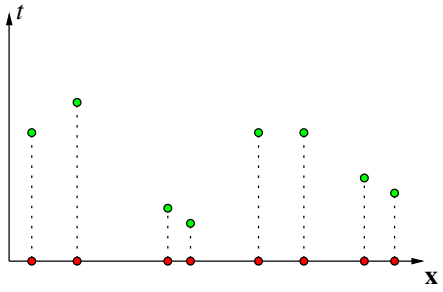

Fig. 3. Sample point time positions before the time integration (marked in red) and after the first time step (marked in green).

As shown in Figure 3 most of the sample points are now asynchronous. To compute the next step for any of the sample points, one would have to evaluate the function values at other sample points at the same point in time.

Fortunately, using the linear Euler time integration we can circumvent this time consuming step. If we save for every sample point not only its current function value and point in time but also the previous function value and point in time, we are able to reconstruct all function values between the previous two points in time by linear interpolation.

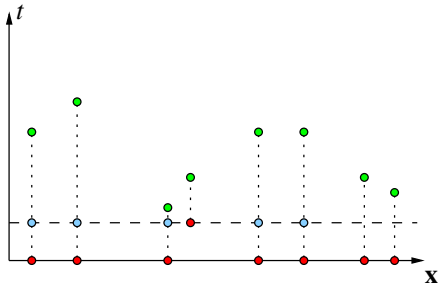

Fig. 4. Progress of time line through the sample points. The earliest sample point is processed further using the linearly interpolated function values (marked in blue) of its neighbors.

In Figure 4, the asynchronous time integration process is visualized. After having computed the first time step, the time line is set to the earliest subsequent point in time. Afterwards the function values of the neighbors are linearly interpolated at this point in time. This interpolation can be performed for all neighbors, since we are always updating the nearest point in time, i. e., we are processing the earliest sample point. Hence, for all other sample points we can access stored values for a future point in time and a past point in time with respect to the current time line. With the interpolated properties, the auxiliary function at the current sample point is integrated in time with the local time-step restriction and the time line proceeds to the next point in time.

If at any time the norm of the auxiliary function gradient exceeds the given threshold, the time line is stopped, all sample points are interpolated to the current point in time, and the reinitialization process is started. During reinitialization the time steps are also bounded by a stability condition and asynchronous time integration is applied in the same way as described before.

The whole evolution process stops, if the function values of the sample points change no more with respect to a certain tolerance, i. e. when the process reaches steady state. In this case, the function values of the sample points are interpolated to the current time line and we get a synchronous auxiliary function.

9 Isosurface Extraction and Rendering

To obtain the smooth isosurface to the initial data set, the zero isosurface of the auxiliary function after convergence is extracted. This is done using an isosurface extraction approach for unstructured point-based volume data presented by Rosenthal and Linsen [29]. An appropriate neighborhood is generated for each sample out of the *k*d-tree structure. For fast access to the nodes of the *k*d-tree and efficient calculation of the neighborhood, an efficient indexing scheme has been introduced.

Exploiting the indexing scheme for fast neighbor computation, isopoints are linearly interpolated between neighboring samples with different auxiliary function signs, where the neighborhood approximates a natural neighborhood. The surface normals to the isopoints are interpolated using a four-dimensional least-squares approach. As a result we get the isosurface represented as a point cloud including surface normals.

Since no connectivity of the generated points of the point cloud is known, a point-based rendering is favorable. The only information we have, are the interpolated function values and normals of the surface points as well as the nearest neighbors. From this information we can generate a splat-based representation of the isosurface using a least-squares approach. The resulting splats are rendered using a point-based raytracing technique [16].

10 Results and Discussion

We applied the presented approach to a variety of data sets to verify our method. For performance analysis we applied it to a resampled unstructured point-based data set of 4M randomly distributed samples, generated from the regular Hydrogen data set of size $128 \times 128 \times 128$. An illustration of the evolution process for this data set is shown in Figure 5.

To compare the computation times, we downsampled the data set to different sizes. These and the following computation times were measured on a 2.66 GHZ XEON processor. The runtime analysis for the preprocessing is presented in Table 1.

# samples	*k*d-tree generation	NN-calculation
500k	1 sec	23 sec
1M	2 sec	50 sec
2M	4 sec	111 sec
4M	9 sec	239 sec

Table 1. Computation times for the preprocessing of the Hydrogen data set with different sample quantities. The preprocessing includes the generation of the *k*d-tree and nearest neighbor calculation.

The runtimes for the isopoint extraction were also analyzed regarding the Hydrogen data set. A summary of the results including the

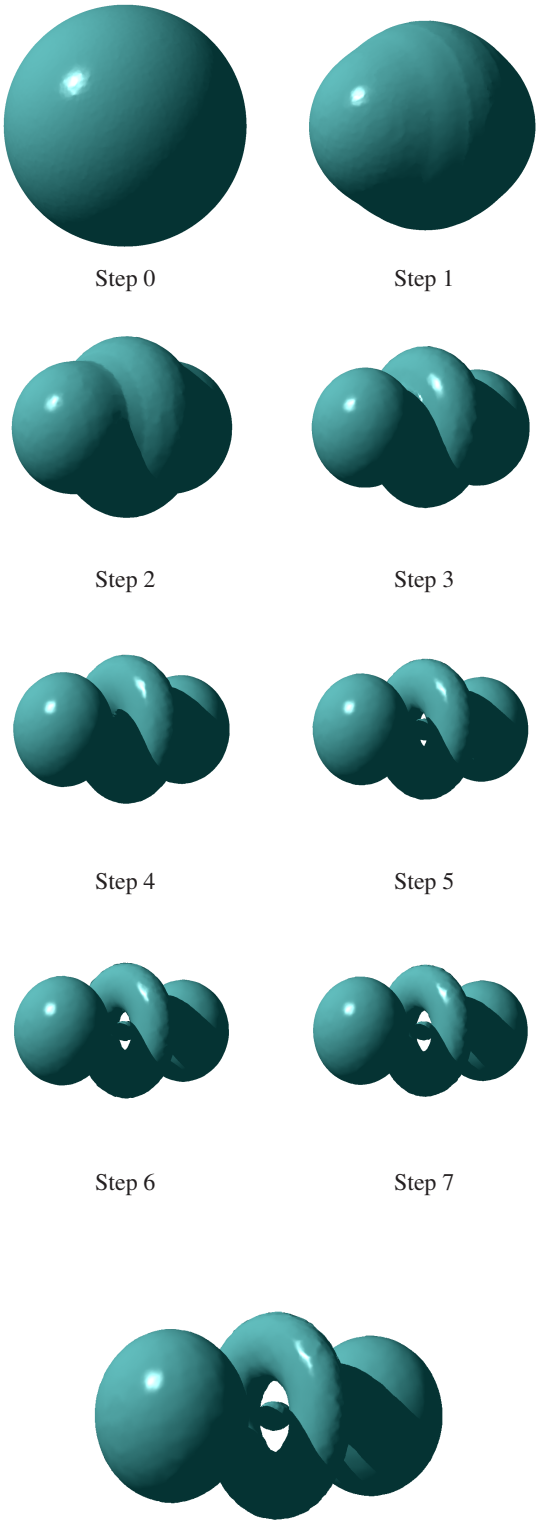

Fig. 5. Point-based raytracing of the smooth isosurface extraction process on the Hydrogen data set with 4M randomly distributed sample points. (Data set courtesy of Peter Fassbinder and Wolfgang Schweizer, SFB 382 University Tübingen.)

number of samples, number of extracted zero-level-set points, calculation time for the neighborhood information, and calculation time for the point extraction is shown in Table 2. Because of some tuning in the implementation, we achieved significantly faster results than those stated by Rosenthal and Linsen [29].

# samples	# points	neighborhood	point extr.
500k	7k	0.8 sec	0.5 sec
1M	12k	1.7 sec	1.1 sec
2M	18k	4.1 sec	1.9 sec
4M	29k	10.1 sec	2.6 sec

Table 2. Isopoint extraction times for the hydrogen data set. The number of extracted surface points and extraction times refer to the zero isosurface to the auxiliary function after convergence of the evolution process.

The whole method applied to the 4M hydrogen data set, including preprocessing, evolution of the auxiliary function with synchronous time integration, and isopoint extraction lasted 22 minutes. A detailed analysis of the computation times of the evolution process and the reinitialization is given in Table 3. We observed that an asynchronous update step is significantly slower than a synchronous one. Thus, using an adaptive time integration scheme only pays off in case of heavily varying sample density.

	aux. function evolution	reinitialization
synchr. int.	59k samp./sec	88k samp./sec
asynchr. int.	2k samp./sec	46k samp./sec

Table 3. Computation times comparison for the evolution and the reinitialization process with synchronous and asynchronous time integration. The times are specified in thousand processed sample points per second.

Finally we applied our method to a real-world example of an unstructured point-based volume data sets, provided by astrophysical particle simulations of Stephan Rosswog, Jacobs University, Bremen, Germany. In the simulation, a set of particles representing a White Dwarf passes a black hole and is torn apart by the strong gravity.

The data sets represent a snapshot of this simulation at a certain point in time, where several physical properties are given. Two smooth isosurfaces to different isovalues were extracted from a density data set with 500k sample points. They are shown in Figure 6. The whole process with asynchronous time integration lasted 68 minutes.

To compare our PDE-based surface extraction approach to direct isosurface extraction in terms of quality of the extracted surface, we extracted an isosurface from the 500k White Dwarf data set with both algorithms. A visualization of the extracted point clouds is presented in Figure 7. The output generated by the PDE-based approach does not exhibit any outliers and results in a much smoother surface.

11 Conclusion and Future Work

We have presented a smooth surface extraction method that can directly be applied to unstructured point-based volume data sets. No global or local three-dimensional mesh generation or reconstruction over a regular grid is applied. The presented approach is able to extract surfaces with respect to normal advection to a scalar field and mean curvature flow.

In a preprocessing step, the samples are stored in a *k*d-tree and for every sample the nearest neighbors are calculated. Afterwards, an auxiliary function is initialized as a signed distance function. Subsequently, the evolution of this function begins.

The needed properties of the auxiliary function, like gradient or mean curvature, are approximated using four-dimensional least-squares approaches and the function is processed according to hyperbolic normal advection and mean curvature flow. If the auxiliary function departs from a signed-distance function, a reinitialization is performed. For time integration of the evolutions, a synchronous or

Fig. 6. Isosurfaces of the 500k White Dwarf simulation data set. The underlying scalar field represents the density in space. We segmented the data set regarding $40\mathrm{g/cm^3}$ (surface on the left side) and $100\mathrm{g/cm^3}$ (surface on the right side). (Data set courtesy of Stephan Rosswog, Jacobs University, Bremen, Germany.)

an asynchronous approach are applied assuring stable time steps. The auxiliary function is processed until it reaches a steady state.

To visualize the computed surface, a point cloud representing of the zero isosurface to the auxiliary function is extracted. A point-based rendering technique using splats is executed on this point cloud. To analyze our method, we applied it to several data sets.

In terms of future work a localization of the algorithm, similar to the ideas by Peng et al. [28], as well as an integration of the reinitialization step into the processing of the auxiliary function, as proposed by Lefohn et al. [15], could be considered to speed up the calculations. Also, the nearest neighbor calculation can be improved using faster algorithms [17]. A general question would be, if it is reasonable to use higher-order time discretization schemes for the evolution process like the ones proposed by Gottlieb et al. [12]. Finally, the rendering engine should be improved, since the splatting approach used has problems with handling sharp edges [27].

A Gradient Calculation in $\mathbb{R}^3$

For the partial derivative of the function $\varphi : \mathbb{R}^3 \to \mathbb{R}$, represented through the points $(x_i, y_i, z_i, \varphi_i)$, $i = 1, \ldots, n$, in y-direction we get

$$\frac{\partial \varphi}{\partial y} = \frac{X_1 \sum_{i=1}^{n} x_i \varphi_i + X_2 \sum_{i=1}^{n} y_i \varphi_i + X_3 \sum_{i=1}^{n} z_i \varphi_i + X_4 \sum_{i=1}^{n} \varphi_i}{Y},$$

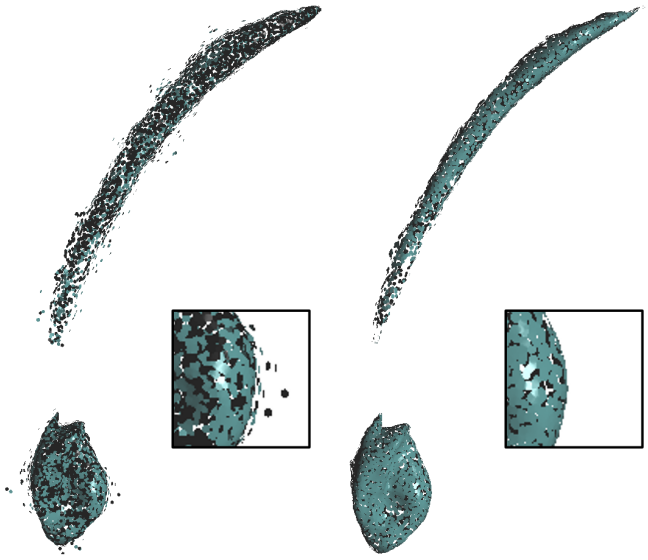

Fig. 7. Comparison between direct isosurface extraction on the left side and smooth isosurface extraction on the right side for the 500K White Dwarf simulation data set. To illustrate the significant advantage of the PDE-based approach over direct isosurface extraction, we include a close-up view on the surface and show a rendering of the surface points with very small splats.

where

$$\begin{aligned}
X_1 = &\sum_{i=1}^{n} x_i y_i \left(n \sum_{i=1}^{n} z_i^2 - \left(\sum_{i=1}^{n} z_i \right)^2 \right) \\
&+ \sum_{i=1}^{n} x_i z_i \left(\sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i - n \sum_{i=1}^{n} y_i z_i \right) \\
&+ \sum_{i=1}^{n} x_i \left(\sum_{i=1}^{n} y_i z_i \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i^2 \right) \\
X_2 = &\sum_{i=1}^{n} x_i^2 \left(\left(\sum_{i=1}^{n} z_i \right)^2 - n \sum_{i=1}^{n} z_i^2 \right) \\
&+ \sum_{i=1}^{n} x_i z_i \left(n \sum_{i=1}^{n} x_i z_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i \right) \\
&+ \sum_{i=1}^{n} x_i \left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i^2 - \sum_{i=1}^{n} x_i z_i \sum_{i=1}^{n} z_i \right) \\
X_3 = &\sum_{i=1}^{n} x_i^2 \left(n \sum_{i=1}^{n} y_i z_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i \right) \\
&+ \sum_{i=1}^{n} x_i y_i \left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i - n \sum_{i=1}^{n} x_i z_i \right) \\
&+ \sum_{i=1}^{n} x_i \left(\sum_{i=1}^{n} x_i z_i \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i z_i \right) \\
X_4 = &\sum_{i=1}^{n} x_i^2 \left(\sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i^2 - \sum_{i=1}^{n} y_i z_i \sum_{i=1}^{n} z_i \right) \\
&+ \sum_{i=1}^{n} x_i y_i \left(\sum_{i=1}^{n} x_i z_i \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i^2 \right) \\
&+ \sum_{i=1}^{n} x_i z_i \left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i z_i - \sum_{i=1}^{n} x_i z_i \sum_{i=1}^{n} y_i \right)
\end{aligned}$$

and

$$
\begin{aligned}
Y = &\left(\sum_{i=1}^{n} y_i z_i\right)^2 \left(n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right) \\
&+ \left(\sum_{i=1}^{n} x_i z_i\right)^2 \left(n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right) \\
&+ \left(\left(\sum_{i=1}^{n} x_i y_i\right)^2 - \sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2\right) \left(n\sum_{i=1}^{n} z_i^2 - \left(\sum_{i=1}^{n} z_i\right)^2\right) \\
&+ \sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i^2 \left(\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i\right) \\
&+ \sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i^2 \left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} y_i\right) \\
&+ 2\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i z_i \left(\sum_{i=1}^{n} y_i \sum_{i=1}^{n} y_i z_i - \sum_{i=1}^{n} y_i^2 \sum_{i=1}^{n} z_i\right) \\
&+ 2\sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} y_i z_i \left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} z_i - n\sum_{i=1}^{n} x_i z_i\right) \\
&+ 2\sum_{i=1}^{n} y_i \sum_{i=1}^{n} z_i \left(\sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i z_i - \sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i z_i\right) .
\end{aligned}
$$

Analogously, we obtain the partial derivatives in x- and z-direction.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under project grant LI-1530/6-1.

References

[1] R. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Science Engineering, 3 edition, 1999.

[2] D. Breen, R. Whitaker, K. Museth, and L. Zhukov. Level set segmentation of biological volume data sets. In *Handbook of Medical Image Analysis, Volume I: Segmentation Part A*, pages 415–478, New York, 2005. Kluwer.

[3] R. Courant, K. Friedrichs, and H. Lewy. On partial difference equations of mathematical physics. *IBM J.*, 11:215–222, 1967.

[4] H. Dym and H. McKean. *Fourier Series and Integrals*. Academic Press Inc., 1972.

[5] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures*, 83(6-7):479–490, 2004.

[6] J. Escher and G. Simonett. The volume preserving mean curvature flow near spheres. In *Proceedings of the American Mathematical Society*, volume 126, pages 2789–2796, 1998.

[7] L. C. Evans and J. Spruck. Motion of level sets by mean curvature I. *J. Diff. Geometry*, 33:635–681, 1991.

[8] L. C. Evans and J. Spruck. Motion of level sets by mean curvature II. *Trans. American Math. Soc.*, 330(1), 1992.

[9] L. C. Evans and J. Spruck. Motion of level sets by mean curvature III. *J. Geometric Analysis*, 2(2), 1992.

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

[11] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[12] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43(1):89–112, 2001.

[13] S. E. Hieber and P. Koumoutsakos. A lagrangian particle level set method. *J. Comput. Phys.*, 210(1):342–367, 2005.

[14] S. Larsson and V. Thome. *Partial differential equations with numerical methods*. Springer, 2005.

[15] A. E. Lefohn, J. M. Kniss, C. D. Hansen, and R. T. Whitaker. Interactive deformation and visualization of level set surfaces using graphics hardware. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 11, Washington, DC, USA, 2003. IEEE Computer Society.

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

[17] J. McNames. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001.

[18] R. B. Milne. *An Adaptive Level-Set Method*. PhD thesis, University of California, Berkeley, 1995.

[19] B. S. Morse, W. Liu, T. S. Yoo, and K. Subramanian. Active contours using a constraint-based implicit representation. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 285–292, Washington, DC, USA, 2005. IEEE Computer Society.

[20] K. Museth, D. E. Breen, L. Zhukov, and R. T. Whitaker. Level set segmentation from multiple non-uniform volume datasets. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 179–186, Washington, DC, USA, 2002. IEEE Computer Society.

[21] Y. A. Omelchenko and H. Karimabadi. Self-adaptive time integration of flux-conservative equations with sources. *J. Comput. Phys.*, 216(1):179–194, 2006.

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

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

[24] N. F. Otani. Computer modeling in cardiac electrophysiology. *J. Comput. Phys.*, 161(1):21–34, 2000.

[25] S. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann. A framework for real-time volume visualization of streaming scattered data. In M. Stamminger and J. Hornegger, editors, *Proceedings of Tenth International Fall Workshop on Vision, Modeling, and Visualization 2005*, pages 225–232,507. DFG Collaborative Research Center, 2005.

[26] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann. Discrete Sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):243–253, 2006.

[27] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[28] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A pde-based fast local level set method. *J. Comput. Phys.*, 155(2):410–438, 1999.

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

[30] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, second edition edition, 1999.

[31] J. Strain. Tree methods for moving interfaces. *J. Comput. Phys.*, 151(2):616–648, 1999.

[32] J. C. Strikwerda. *Finite difference schemes and partial differential equations*. Wadsworth Publ. Co., Belmont, CA, USA, 1989.

[33] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comput. Phys.*, 148:81–124, 1999.

[34] J. W. Thomas. *Numerical Partial Differential Equations*. Springer, 1998.