Motion Visualization in Large Particle Simulations

Roland Fraedrich[a] and Rüdiger Westermann[a]

[a]Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

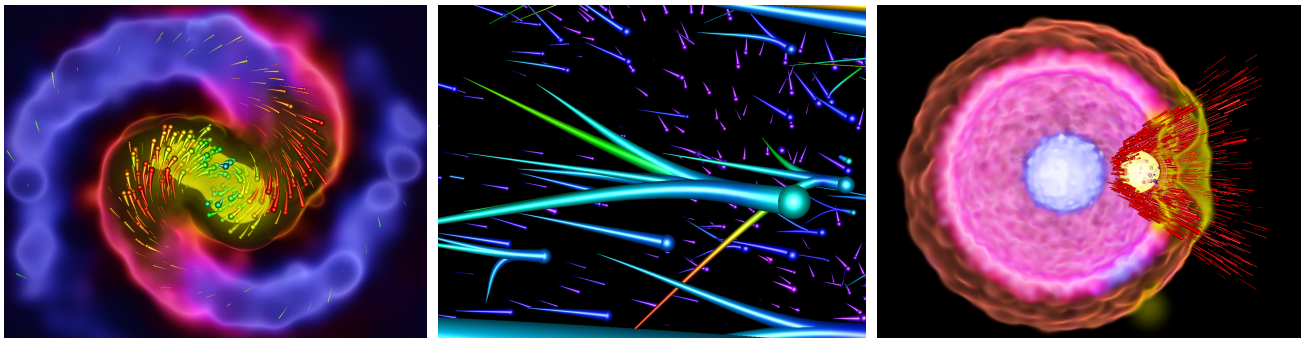

Figure 1. We present a novel space-time hierarchy for large particle sets, which allows for the analysis of formation processes in astrophysical particle simulations, such as the merging of particles into clusters (middle image). Our approach can show the motion of mass at multiple levels of detail, and it allows classifying the motion dynamics based on directional information.

ABSTRACT

Interactive visualization of large particle sets is required to analyze the complicated structures and formation processes in astrophysical particle simulations. While some research has been done on the development of visualization techniques for steady particle fields, only very few approaches have been proposed to interactively visualize large time-varying fields and their dynamics. Particle trajectories are known to visualize dynamic processes over time, but due to occlusion and visual cluttering such techniques have only been reported for very small particle sets so far. In this paper we present a novel technique to solve these problems, and we demonstrate the potential of our approach for the visual exploration of large astrophysical particle sequences. We present a new hierarchical space-time data structure for particle sets which allows for a scale-space analysis of trajectories in the simulated fields. In combination with visualization techniques that adapt to the respective scales, clusters of particles with homogeneous motion as well as separation and merging regions can be identified effectively. The additional use of mapping functions to modulate the color and size of trajectories allows emphasizing various particle properties like direction, speed, or particle-specific attributes like temperature. Furthermore, tracking of interactively selected particle subsets permits the user to focus on structures of interest.

Keywords: Particle visualization, pathlines, multi-resolution representation, space-time hierarchy.

1. INTRODUCTION

For the visual analysis of unsteady flow fields, particle-based approaches, such as interactively seeding and tracking massless particles and constructing pathlines showing their motion, are a well-established class of techniques. In contrast to pure particle visualization, where particles are rendered as individual graphical primitives, these characteristic lines allow for an improved understanding of complicated motion patterns. This is because pathlines can depict the time history of the position of a particle in one image, at the same time providing additional exploration possibilities. For instance, geometric properties of the graphical objects used to render these lines can be modified according to the intrinsic flow properties. In this way local structures in the domain can be revealed quite effectively.

Further author information: (Send correspondence to Roland Fraedrich)
Roland Fraedrich: E-mail: fraedrich@tum.de
Rüdiger Westermann: E-mail: westermann@tum.de

Characteristic flow lines are usually computed by seeding massless particles into the flow domain and using numerical techniques to integrate these particles through space based on the underlying flow field. For the visualization of grid-based Eulerian simulations, where the fluid velocities are observed at locations that are fixed in space, numerical integration requires the interpolation of velocities from these locations to progress a particle in space and time. In Lagrangian particle simulations, for instance Smoothed Particle Hydrodynamics (SPH), where individual fluid material volumes are carried with the flow, the quantities given at the discrete set of particles are smoothed using some kind of point spread function to obtain a continuum. Mathematically this is written as

$$A(\mathbf{r}) = \sum_j \mathbf{m}_j \cdot \frac{A_j}{\rho_j} \cdot W(|\mathbf{r} - \mathbf{r}_j|, h_j),$$

where $A(\mathbf{r})$ is the smoothed quantity at position $\mathbf{r}$, and m_j, ρ_j, $\mathbf{r}_j$, and A_j are the mass, density, position, and the carried quantity, respectively, of the particles having an influence on $\mathbf{r}$. W is the point spread or kernel function with a support h_j that can vary from particle to particle.

In Lagrangian particle simulations of fluids, the features of the entire physical system can also be obtained by tracing the motion of the simulated material volumes itself. This motion is defined by the underlying physical model, and it describes the dynamics of particles that carry a mass, which is smoothed over a certain volume around the particle. Accordingly, the mass transport within the simulated domain is completely described by the motion of the entire particle set, which is not guaranteed for the tracing of massless particles. It is worth noting, however, that the simulated particles behave slightly different than massless particles, since they interact with neighboring particles through a repelling force that represents the material pressure and gravitational effects. Mathematically this means that the smooth function $A(\mathbf{r})$ is usually not equal to the particle quantities at the particle positions, yet this function will generally not be too far from these values. Consequently, the trajectories of SPH particle will be close to the trajectories of massless particles seeded into the continuous vector field given by the discrete particle set, but they will not exactly coincide with these trajectories.

In this work we address the problem of interactively exploring the motion paths of particles in astrophysical simulations via visualization. The time-dependent data sequences we explore are comprised of sets of particles that were simulated using cosmological N-body/SPH simulations. In these simulation, the motion of a large number of particles under their own self-gravity is followed using Newton's equations of motion. Each sequence shows the evolution of a set of particles over time, with the particles being represented by their position as well as the physical quantities they carry, such as mass, temperature, and volume.

At first glance, the given problem seems to be easy. By simply connecting corresponding particles in adjacent time steps via line segments or higher order curves, and by rendering the so constructed pathlines via computer graphics techniques, the motion paths of each particle over time can be visualized. However, in practice this approach is of limited use due to the following reasons: Firstly, the visualization of pathlines does not allow for a (relative) analysis of the particles' velocities. Secondly, by solely following individual particles on the simulation level, it is very difficult to analyze cluster formations and splits in the particle distributions. Thirdly, since the simulations we address are comprised of many millions of particles, occlusion effects and visual clutter thereof prohibit an effective visualization in terms of revealing the relevant motion structures. Furthermore, the massive amount of geometry that has to be rendered for visualizing the pathlines of all particles can easily exceed the capacities of even the most powerful graphics accelerators.

To overcome these limitations we propose a novel space-time hierarchy for representing time-dependent particle distributions at different scales. Based on this hierarchy, we present interactive visualization techniques for depicting the motion paths of single particles as well as coherent clusters of particles. In addition to direction, we can show the motion velocity along these paths via animated color maps or geometry structures that move according to the velocity. In particular, our paper makes the following specific contributions:

- A space-time hierarchy for particle sets.
- A level-of-detail approach for the visualization of particle simulations via pathlines.
- Enhancement of regions of interest via user-defined mapping functions to modulate the appearance of trajectories according to cluster properties or via interactive selection and tracking of particle subsets.

Once the space-time hierarchy has been built on the CPU in a pre-process, the visualization runs entirely on the GPU and provides interactive rates even for high resolution particle sequences. In combination with in-core and out-of-core strategies to stream particle sets that are too large to fit into GPU memory, rates of more than 40 frames per second can be achieved for time-varying data sets consisting of millions of particles.

The space-time hierarchy can be used for representing the mass flow on different levels of detail, giving rise to an effective analysis of physical processes such as colliding and merging mass structures. Furthermore, it enables level-of-detail visualizations of particle trajectories to reduce the amount of graphical primitives to be rendered. Notably the performed clustering can more generally be used to create a time-dependent multi-resolution hierarchy of the particle field, by merging particles according to the resampling operators presented for adaptive SPH simulations.[1,2] This hierarchy can then be used to combine visualizations of particle trajectories with multi-resolution visualizations of the particle density field, or any other quantity carried by the particles.

The remainder of this paper is organized as follows. In the next section we review previous work that is related to ours. The proposed space-time hierarchy for particle sets is introduced in Section 3. Section 4 presents the scale-dependent mapping techniques we propose to effectively visualize motion clusters in these sets. In Section 5 we evaluate the effectiveness of our approach for the visual analysis of particle simulations, and we analyze the performance of our approach. We conclude the paper with an outline of future research in the field.

2. RELATED WORK

Particle-based visualization techniques have been studied extensively over the last years, mainly in the context of flow visualization. There is a vast body of literature related to this field and a comprehensive review is beyond the scope of this paper. However, Post et al.,[3] Laramee and Hauser,[4] and McLoughlin et al.[5] discuss the basic principles underlying such techniques and provide many useful algorithmic and implementation specific details.

Only very little work has been reported on the application of flow visualization techniques for visualizing time-varying particle data. Most commonly, such sequences are rendered by visualizing the continuous scalar fields represented by the particle quantities in succession and revealing the motion dynamics via the evolution of the so constructed fields. Techniques to efficiently resample these fields to grid structures that can be rendered efficiently have been proposed in Ref. 6–10. Splatting of transparent particle sprites was presented by Ref. 11–13. Dedicated visualization systems for SPH data, including rendering options like particle splatting or slicing, and providing specific mechanisms for handling and analyzing large particle data were discussed in Ref. 14–18.

To the best of our knowledge, only very few approaches have explicitly addressed the efficient visualization of particle pathways in particle based simulations. In the work of Schindler et al.[19] the focus was on the visualization of vortex core lines in flow fields given by SPH simulations. One major contribution was that all computations were performed directly on the SPH representation to achieve high quality and consistency with this representation.

Our work is inspired by multi-scale techniques for time-varying scalar fields, as proposed, for instance, in Ref. 20–23. Underlying these approaches is some form of multi-resolution representation to classify distribution characteristics in space as well as temporal motion characteristics. Even though we do not explicitly encode the particles' motion trajectories at different time scales, by performing motion-based clustering in combination with trajectory visualization a very similar analysis of temporal activities in particle data is achieved.

3. SPACE-TIME HIERARCHY

Our construction of a space-time hierarchy for time-dependent particle data is related to clustering methods for vector fields[?,24–27] in that it considers similar merging criteria for clustering particles. In contrast to these techniques, however, our objective is not to approximate a given flow field by a sparse set of vector samples, but to hierarchically describe the distribution and flow of mass over time that is represented by the motion of individual particles. For this purpose, we introduce an estimate for the error in this transport when approximating a particle by a cluster. This error estimate is used as a metric to guide the hierarchical clustering of the particles in the initial time step, and to track splits and mergers of clusters over time (see Figure 2). The constructed space-time structure encodes the particle sets of all time steps hierarchically. This hierarchy is stored in a spatially-adaptive octree data structure, attributed by additional information to keep track of the clusters' evolutions over time.

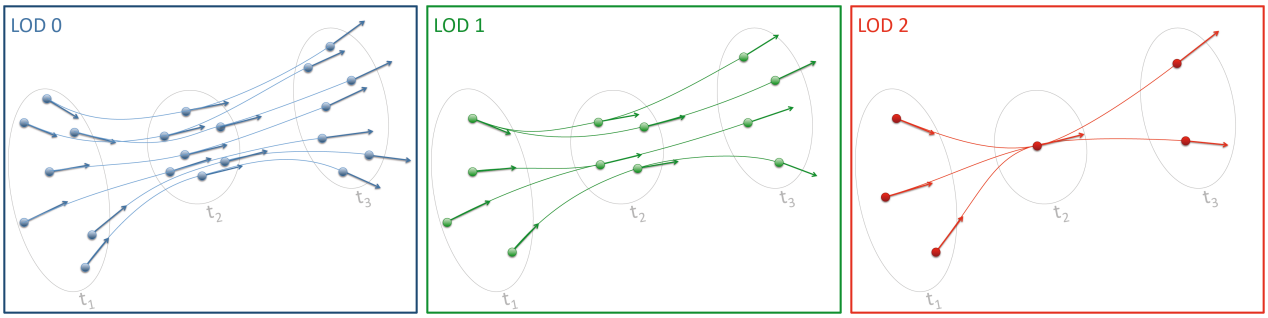

Figure 2. Illustration of our space-time hierarchy for particle trajectories with 3 time steps and for 3 levels of detail.

3.1 Error Estimation

Since at the core of our method is the merging of particles into clusters and the splitting of clusters once their particles start to diverge, a criterion is required to steer these operations. The criterion we propose is based on the error between the trajectory of a particle and the trajectory of the cluster into which this particle should be merged, or which contains this particle and should be split.

To generate a smooth trajectory from one particle's position at time step i to its position at time step $i+1$ we interpolate smoothly between these two positions using a cubic Hermite spline:

$$\mathbf{r}(t) \quad = \quad (2t^3 - 3t^2 + 1) \cdot \mathbf{r}_p^{(i)} + (t^3 - 2t^2 + t) \cdot \mathbf{v}_p^{(i)} + (-2t^3 + 3t^2) \cdot \mathbf{r}_p^{(i+1)} + (t^3 - t^2) \cdot \mathbf{v}_p^{(i+1)}.$$

Here, $\mathbf{r}_p^{(i)}$ and $\mathbf{r}_p^{(i+1)}$ are the particle's positions at the current and the next time step, respectively, $\mathbf{v}_p^{(i)}$ and $\mathbf{v}_p^{(i+1)}$ are the corresponding tangents, $t \in [0,1]$ is the interpolation factor, and $\mathbf{r}$ is the interpolated position. The tangents are given by the particle velocities.

When we merge a particle into a cluster, we essentially replace the particle trajectory by the trajectory of that cluster. The trajectory of a cluster is also computed as a Hermite spline from adjacent cluster positions. Let us assume that a cluster has control points $\mathbf{r}_C^{(i)}$ and $\mathbf{r}_C^{(i+1)}$ and tangents $\mathbf{v}_C^{(i)}$ and $\mathbf{v}_C^{(i+1)}$. Then, the error function expressing the spatial deviation between the particle trajectory and the cluster trajectory is given by

$$\begin{aligned} \epsilon(t) \quad = \quad & \big|\big| \, (2t^3 - 3t^2 + 1)(\mathbf{r}_C^{(i)} - \mathbf{r}_p^{(i)}) + (t^3 - 2t^2 + t)(\mathbf{v}_C^{(i)} - \mathbf{v}_p^{(i)}) + \\ & + (-2t^3 + 3t^2)(\mathbf{r}_C^{(i+1)} - \mathbf{r}_p^{(i+1)}) + (t^3 - t^2)(\mathbf{v}_C^{(i+1)} - \mathbf{v}_p^{(i+1)}) \, \big|\big|. \end{aligned}$$

The maximum of this function is used to decide whether merging a particle yields an acceptable approximation of its trajectory or not. To avoid calculating the maximum, we derive an upper bound $\tilde{\epsilon}$ for the maximum and use this bound as a reasonable estimate. Based on the observations that the sum of the weights of the control point positions always evaluates to 1 and the maximum weight of each tangent is $4/27$, the following bound is obtained:

$$\tilde{\epsilon} = \max\big(||\mathbf{r}_C^{(i)} - \mathbf{r}_p^{(i)}||, ||\mathbf{r}_C^{(i+1)} - \mathbf{r}_p^{(i+1)}||\big) + \frac{4}{27}\big(||\mathbf{v}_C^{(i)} - \mathbf{v}_p^{(i)}|| + ||\mathbf{v}_C^{(i+1)} - \mathbf{v}_p^{(i+1)}||\big). \tag{1}$$

3.2 Particle Merging

Based on the error estimate that has been derived for quantifying the "distance" between trajectories, we now describe how this estimate is used to guide our clustering process, and how the properties of a cluster based on the particles contained in this cluster are obtained. For the latter, we apply the resampling operators for adaptive SPH simulations,[1,2] which require the volume of the cluster to be the sum of the volumes of the merged particles, while any other vector or scalar quantity is averaged according to the particles' mass contributions.

Due to this definition, a cluster's position $\mathbf{r}_C^{(i)}$ and velocity $\mathbf{v}_C^{(i)}$ are not constant over time, but they have to be updated whenever a new particle is integrated. Accordingly, in a merging operation it is not sufficient to test whether a particle can be merged into a cluster, but it also has to be verified that after merging the

particle and updating the cluster all particles contained in it are still well represented with respect to the error estimate. Instead of computing the error estimate separately for every particle of a cluster, we use a conservative estimation of the maximum error of the cluster members. The basic idea is to memorize a former position and velocity of the cluster along with the maximum error of the particles with respect to these values. As long as the difference between the former representation and the new cluster properties, plus the memorized maximum error does not exceed the permitted error threshold, the merging of the particle is valid.

More formally, let $\hat{\mathbf{r}}_C^{(i)}$ and $\hat{\mathbf{v}}_C^{(i)}$ be some former position and velocity of a cluster C. In correspondence to the error estimate $\hat{\epsilon}$, we memorize the maximal errors of all particles in the cluster that result from considering either the positional error of the previous time step $(i-1)$ or that of current time step (i):

$$\theta_C^{(i)} = \max\left(||\mathbf{r}_{C'}^{(i-1)} - \mathbf{r}_p^{(i-1)}|| + \frac{4}{27}\left(||\mathbf{v}_{C'}^{(i-1)} - \mathbf{v}_p^{(i-1)}|| + ||\hat{\mathbf{v}}_C^{(i)} - \mathbf{v}_p^{(i)}|| \right) \,\middle|\, p \in C \right), \tag{2}$$

$$\delta_C^{(i)} = \max\left(||\hat{\mathbf{r}}_C^{(i)} - \mathbf{r}_p^{(i)}|| + \frac{4}{27}\left(||\mathbf{v}_{C'}^{(i-1)} - \mathbf{v}_p^{(i-1)}|| + ||\hat{\mathbf{v}}_C^{(i)} - \mathbf{v}_p^{(i)}|| \right) \,\middle|\, p \in C \right). \tag{3}$$

Here C' denotes the cluster that contains particle p at time $(i-1)$. Given a new cluster center $\mathbf{r}_C^{(i)}$ and velocity $\mathbf{v}_C^{(i)}$ that would result from merging an additional particle, the new maximum error ϵ' within the cluster is estimated by adding the differences of the positions and velocities to the memorized error terms:

$$\epsilon' = \max\big(\theta_C^{(i)}, \delta_C^{(i)} + ||\mathbf{r}_C^{(i)} - \hat{\mathbf{r}}_C^{(i)}||\big) + \frac{4}{27}||\mathbf{v}_C^{(i)} - \hat{\mathbf{v}}_C^{(i)}||$$

If this estimate is below the maximum permitted error, the particle can be merged with the cluster. Otherwise, we reset $\hat{\mathbf{r}}_C^{(i)}$ and $\hat{\mathbf{v}}_C^{(i)}$ to $\mathbf{r}_C^{(i)}$ and $\mathbf{v}_C^{(i)}$, and determine the new maximum error terms $\theta_C^{(i)}$ and $\delta_C^{(i)}$ to recalculate ϵ'. If the error still exceeds the threshold, no merging is performed. Otherwise the particle is integrated into the cluster. For performance reasons, the update of $\theta_C^{(i)}$ and $\delta_C^{(i)}$ are skipped when the values are close to the maximum permitted error. Nevertheless, particles may still be inserted if the update of $\mathbf{r}_C^{(i)}$ and $\mathbf{v}_C^{(i)}$ is small enough so that ϵ' does not exceed the threshold.

If clusters are built by merging clusters which have already been formed at a finer level of detail, and which already introduce errors with respect to position and direction, we adapt the calculation of $\theta_C^{(i)}$ and $\delta_C^{(i)}$ such that we add the maximal errors within such a cluster $\theta_p^{(i)}$ and $\delta_p^{(i)}$ to the respective error estimates.

3.3 Hierarchy Generation

Based on the proposed particle merging strategy, the particle space-time hierarchy can now be constructed. The hierarchy is comprised of one adaptive octree per time step, each of which is created in a bottom-up fashion. In the hierarchy of each time step, the particles are clustered consecutively up to a maximum error for the given level. The maximum permitted error for the root level is defined by the user, and according to the octree data structure it is halved for each additional level of detail. Additionally, we store the particle trajectories between successive time steps as the mapping from the clusters of one time step to the clusters of the next time step.

3.3.1 Initial Time Step

At the first time step, we start by storing the original particles at the leaves of the octree. Afterwards, we successively pass all particles to the next coarser level. For each particle that is inserted we look for potential merging candidates and add each pair of particles that can be merged into a priority queue that is ordered according to our error estimate. To accelerate the neighbor search, we organize the particles within each octree node in a uniform grid with a spacing that corresponds to the error threshold of the specific level. Thus, at most 27 of such bins have to be searched to find the candidate pairs. When all particles are inserted into the current level, we successively merge the particle pair with the smallest distance according to our error estimate. For each merger, the octree as well as the priority queue are updated accordingly. This process is repeated using the next candidate pair until the priority queue is empty. Note that in the special case of the initial time step there are no trajectory to be represented, but just the starting points of trajectory in space and time. Accordingly, the error terms regarding the position and velocity of the previous time step are set to zero.

3.3.2 Subsequent Time Steps

For all successive time steps, we again begin by storing the particles into the leaf nodes of the octree. Then, however, we do not proceed by passing particles from one level to the next coarser one, but we try to merge particles into the clusters of the previous time step for all levels of detail. If due to a merging operation the error estimate is violated for a cluster, it is iteratively split into sub-clusters until the merging criterion is met. These sub-clusters are then inserted into the respective level of the octree. As before, we search for merging candidates within the neighborhood and add each mergeable pair into the priority queue. After insertion of all clusters, we successively merge the clusters as before until the priority queue is empty. Note that by propagating the cluster information to the next time step, we preserve continuity over time and accelerate the merging process by reducing the number of neighboring particles that have to be tested.

The splitting of clusters needs some further explanations. As mentioned before, we start by calculating the error estimate for all particles a cluster is comprised of, in order to verify whether the cluster will remain unchanged. If a cluster has to be split, it is separated into sub-clusters by iteratively applying a PCA-based split algorithm.[28,29] That is, we determine the optimal splitting plane by computing the centroid of the particles' positions and the largest eigenvector of the auto-covariance matrix as surface normal (see Figure 3). Since we aim at minimizing the error estimate at the current time step, the PCA split is performed in the 6-dimensional error space that is given by the particle positions and velocities, where the latter vector is scaled by $4/27$ according to Equation 1. If the merging criterion is still violated within a sub-cluster, it is split as well.

In addition to the clusters that are build for the distinct time steps, we store the trajectories that result from the propagation of the clusters over time including cluster splits and mergers. Each trajectory consists of the cluster indices that define its start and end points within the successive time steps as well as the mass and radius of the represented cluster.

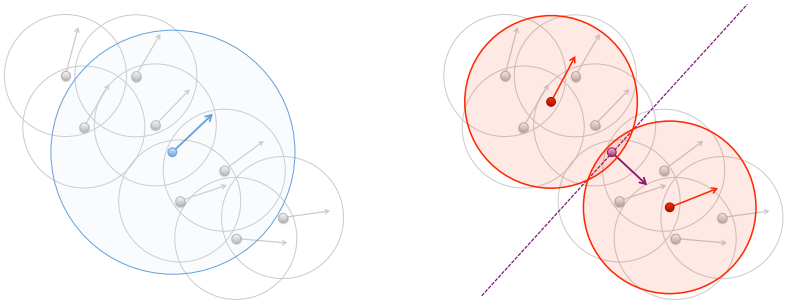

Figure 3. For the splitting of a cluster that violates the merging rule in the successive time step (left), we apply a PCA split to build subclusters (right).

3.4 Data Structures

The described preprocess results in one adaptive octree per time step, including particle and cluster data, as well as the cluster trajectories representing the particle motion between successive time steps. The octrees serve as data structures to hierarchically represent the spatial distribution of particles at the discrete set of time steps. We now augment the octrees with additional information to efficiently determine for a node the spatial region that is traversed by the particles stored at this node between the current and the next time step, yielding a continuous space-time partitioning data structure. Accordingly, successive nodes must contain the data that is needed for the interpolation of particle trajectories within the enclosed space and time. This is usually true for most clusters, but not for those that have left the domain of a node within the represented time. To allow for the interpolation between successive nodes, we insert a *cluster copy* of the end point of such particles into the successor of the starting point's node and adapt the bounding box of the node accordingly. Such an update is also required if a particle or cluster leaves the domain temporarily.

For each node of the space-time hierarchy, we finally store three data structures. The *cluster list* contains the clusters built at the distinct time steps and stores any quantity that is interpolated between two time steps, including the control points for the spline interpolation. The *trajectory list*, on the other hand, represents the

information of the actual flow of mass between successive time steps, including the amount of mass itself, the radius of the cluster and, most importantly, the control points of the trajectory. These are given by the indices in the cluster lists of the octree node and its successor (see Figure 4). The trajectory list is sorted by the cluster index of the first control point to allow for an efficient tracking of particles (see Section 4.1.2). To further support the tracking of particles, for each cluster copy we store the node ID and cluster list index of the original cluster in a *copy list*.

On the finest level of detail, which contains the individual particles of the simulation, merging and splitting of particles does not occur and the trajectory list has the same entries as the cluster list of the same time step. Thus, the index within the trajectory list into this cluster list is redundant and does not need to be stored.

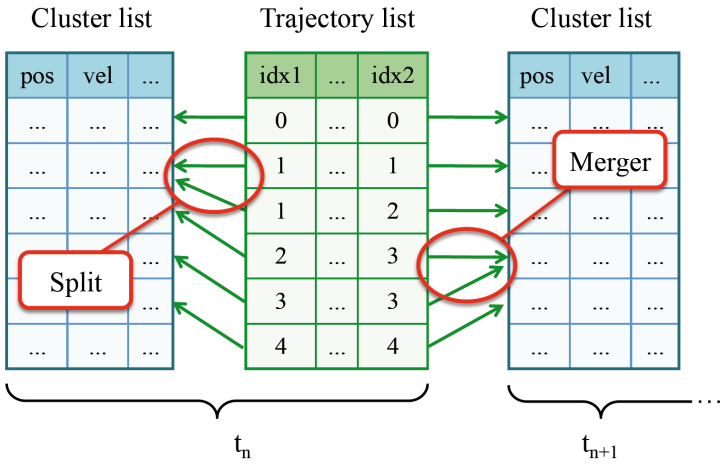

Figure 4. Illustration of the data structures of our space-time hierarchy. The clusters of each time step are stored in a cluster list. The trajectory list contains the propagation of clusters over time with *idx1* and *idx2* referencing the end points for interpolation. The copy list helps to track clusters that left the domain of the octree node at this time step.

4. RENDERING

The interpolation of clusters between successive time steps and the rendering of cluster trajectories is entirely performed on the GPU. Therefore, the data of all visible octree nodes are transferred to the GPUs local memory. The trajectory list of each octree node is bound as a vertex buffer and the cluster lists of this and the successive time step are bound as buffer resources. Based on the indices given per vertex, the position and tangents of the trajectories' end points can be accessed, and they are then used to interpolate the cluster's position and velocity by cubic Hermite splines. The mass and (if needed) the radius of the cluster are given per vertex. Particle trajectories are approximated by piecewise linear segments that are adaptively refined in the geometry shader by interpolating intermediate spline positions. To give a better impression of the shape and direction of the trajectory, the line can additionally be extruded to a tube or arrow.

4.1 User Interaction

Especially in large SPH simulations with many millions of particles, a scientist needs to be able to efficiently guide the exploration process and focus on the structures of interest. For this purpose, the visualization of the time-space hierarchy can be enriched with additional features to graphically emphasize certain regions or properties.

4.1.1 LOD Selection

Rendering is performed for all visible octree nodes and for all successive time steps that are covered by the current trajectory, i.e., all time step that are required for rendering the trajectory. Since this data contains clusters and trajectories at different resolution levels, a mechanism is provided that allows the user to manually select a specific level of detail for the entire data set. By interactively switching between different levels of detail while navigating through space and time, a quite effective exploration of particle trajectories is already possible.

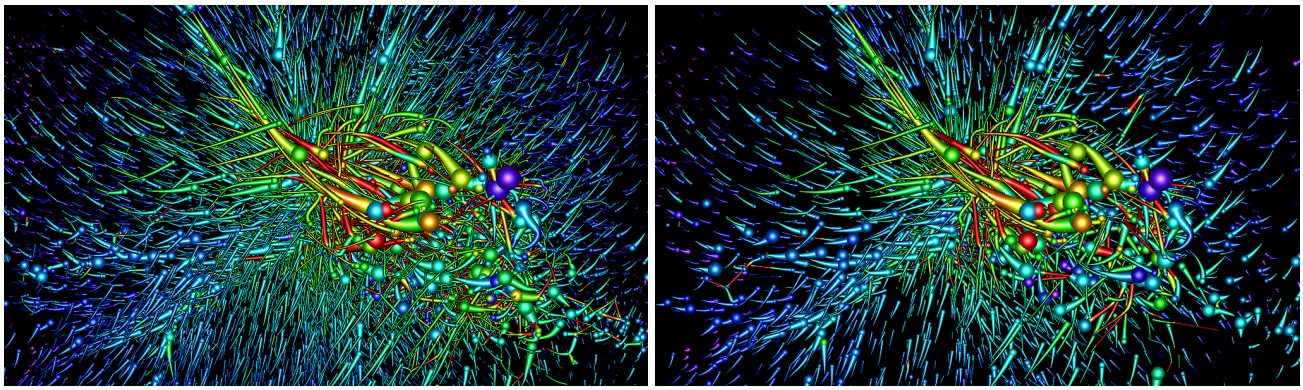

Figure 5. Fixed LOD (46 fps / left) in comparison to a view-dependent LOD (65 fps / right) on a 1440×900 viewport.

A drawback of this approach is that choosing a fine level of detail for a close-up view into a certain region can lead to very unpleasant visual clutter further away from the viewer (see left image of Figure 5). As an alternative, the octree nodes can be rendered automatically at the coarsest level of detail that maintains a given screen space error. This error is given by the error that is caused by replacing particles by the cluster in the current node (see Section 3.1), and the distance of this node to the viewer. Thus, the particle trajectories near the camera are automatically visualized in more detail, whereas the particle motions far away from the camera are represented via few, but larger clusters. This leads to good perception of the particle flow in the entire view frustum (see right image of Figure 5).

One drawback of an adaptive LOD is that discontinuities might be visible for particle trajectories that run through neighboring octree nodes with different levels of detail, and which correspondingly do not share control points between these levels. In practice, however, we measured that less than 0.2 % visible trajectories showed such discontinuities, and since they appear at the borders between different levels of detail, usually they show up only in the background.

4.1.2 Tracking of Particles

An important feature requested by astrophysicists is the possibility to define a specific set of particles and to track the position of only these particles over time. The selection itself can be performed via picking of one or more trajectories on the screen, based on a user-defined region of interest (e.g. a bounding box), or the cluster of interest can be retrieved from a pre-computed cluster catalogue. For the picking of clusters, we write the time step, the ID of the octree node as well as the index of the trajectory to an off-screen render target in order to identify the corresponding cluster of each trajectory on the screen. In case of an interactive spatial selection, we render the particles of all octree nodes that intersect the region of interest. For all particles within this region, the octree's node ID as well as the trajectory list index is written to a stream output buffer and read back to main memory.

The indices of the selected trajectories are stored in *tracking lists* of the corresponding octree nodes. To propagate the cluster selection over time, we determine the cluster list indices of the trajectory end points in the successive time step. Based on these cluster indices, we determine the corresponding trajectory list indices by a binary search and add the selected indices to the node's *tracking list*. Duplicates that result from the merging of particles can be discarded by using a set data structure for the tracking list. The tracking of clusters which leave the domain of an octree node is propagated to the corresponding octree node within the same time step by using the copy list (see Section 3.4).

The trajectories of the selected particles can be exclusively visualized by using the tracking list as index list for rendering (see Figure 6). As an alternative, the particles and their trajectories can be rendered together with the unselected data, but highlighted, e.g. by color. We realize this feature by tagging the selected entries of the trajectory lists using the sign bit of the mass property.

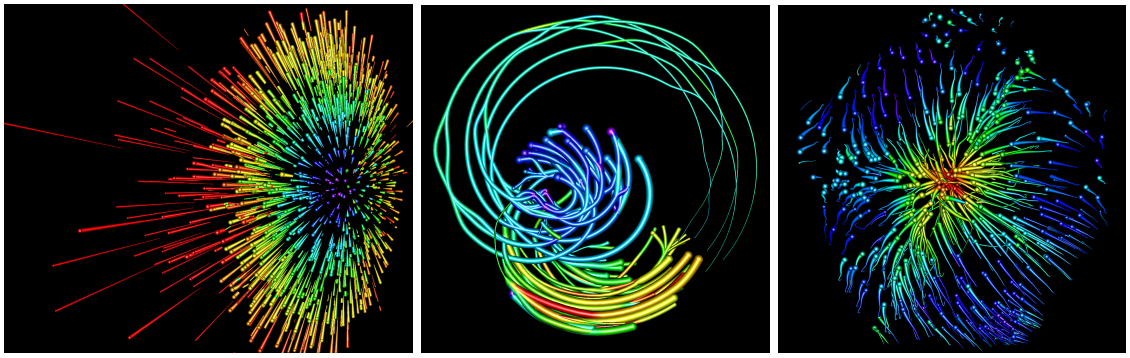

Figure 6. Examples for user defined tracking of particles showing a cross section of the SNIaEjecta data set (left), user-selected clusters over 30 time steps in the WDMerger data set (middle) and a subset of the aquarius dataset (right). The colors of the trajectories represent the cluster's velocities (left) and temperature (middle/right).

4.1.3 Trajectory Appearance

Another possibility to highlight a certain set of sub-clusters is to modulate the appearance of the trajectories according to certain cluster properties. In our standard setting, the size of the trajectory is set such that the area of the cross-section represents the mass of the according cluster. To avoid extreme differences in size for coarser LODs, the user can set a maximum mass threshold to clamp the radius of the trajectories. On the other hand, the trajectories of small clusters can disappear when their diameter falls below the pixel size. To preserve these trajectories the user can define a minimum mass threshold for which cluster trajectories are automatically enlarged to pixel size. The color of the trajectories can be determined by using arbitrary cluster properties, such as direction, velocity, mass, or data-specific attributes like temperature as input for a color map. Thus, the visualization can be enhanced by emphasizing the differences of cluster trajectories (see Figure 6).

4.2 Implementation Details

We have implemented the presented technique using C++ and DirectX. To optimize the data management on the CPU, we have used asynchronous I/O in combination with prefetching of octree nodes in space and time. When nodes are streamed into main memory, they are doubly linked along the time domain. By this, the octrees of the required time steps can be traversed collectively in order to gather the octree nodes needed for rendering. View frustum culling is used to omit invisible parts of the data set. On the GPU, we have implemented buddy memory allocation[30] to avoid time-consuming creation and deletion of resources per node. For the rendering of trajectories as spheres with tails, we have adapted the approach of Merhof et al.[31] to approximate tubes by a hybrid rendering of triangle stripes and point sprites. Furthermore, we added black halos with a size of 1 pixel around the trajectories to enable a better differentiation of the particles. Volume rendering of the SPH data for the hybrid visualization was implemented using perspective grid ray casting as proposed by Fraedrich et al.[9]

5. EXPERIMENTAL RESULTS

Our experiments have been carried out on a standard PC equipped with an Intel Core 2 Quad Q9450 2.66 GHz processor, 8 GB of RAM, and a NVIDIA GeForce GTX 480 with 1.5 GB video memory and PCI Express 2.0 x16. We demonstrate the efficiency of our approach for three different astrophysical particle simulations. *WDMerger* simulates the merger of a white dwarf binary system. *SNIaEjecta* simulates the impact of a supernova ejecta on a companion star. The *Aquarius* data set simulates the evolution of a MilkyWay-sized dark matter halo from shortly after the big bang to the present. In all of these simulations, each particle has a size of 40 Byte containing a unique ID, position and velocity, as well as additional floating point attributes, such as smoothing length, mass, and temperature. Table 1 gives detailed information about the data sets.

For each of the three data sets, we created the proposed space-time hierarchy with 6 levels of detail. This preprocess took 34 minutes for WDMerger, 101 minutes for SNIaEjecta and 115 minutes for the Aquarius data set. Table 2 gives information about the number of particles and cluster copies per level, averaged over all time steps, and shows the amount of data required to store each level of detail and the entire hierarchical data

Table 1. Data sets used in our experiments

Data set	Time steps	# Particles	Data size
WDMerger	84	2,000,000	6.3 GB
SNIaEjecta	49	8,745,571	16.0 GB
Aquarius	25	18,535,972	17.3 GB

Table 2. Number of clusters and copy clusters per level as averages over all time steps as well as the total amount of data for the distinct levels and for the entire space-time hierarchy.

	WDMerger (8,7 GB)			SNIaEjecta (22.3 GB)			Aquarius (24.1 GB)		
Level	# Clusters	# Copy Cl.	Total size	# Clusters	# Copy Cl.	Total size	# Clusters	# Copy Cl.	Total size
5	2,029	0	7 MB	1,495	0	4 MB	533	0	2 MB
4	4,627	378	18 MB	11,443	782	28 MB	2,549	95	3 MB
3	14,698	801	58 MB	64,606	3,532	155 MB	18,233	1087	20 MB
2	56,527	2,570	225 MB	282,963	24,529	685 MB	135,009	17,649	162 MB
1	201,310	10,876	796 MB	844,435	195,037	2,185 MB	846,592	215,892	1,120 MB
0	2,000,000	356,804	7,850 MB	8,745,571	1,375,428	19,782 MB	18,535,972	5,936,997	23,408 MB

representation. Compared to the original data, the data size increases by about 39% in average and the memory overhead of all low-resolution levels did not exceed 3 GB. Accordingly, without the finest level it is possible to load the entire hierarchy including all time steps into main memory. Thus, no expensive disk I/O is needed when navigating through space and time, or between different levels of detail.

Figure 7 shows the trajectories of the WDMerger data set at three levels of detail. As can be seen, the motion within the data is well preserved while the number of trajectories is significantly reduced. Moreover, the original particle trajectories result in a high degree of occlusion that obscures the flow of mass further inside the data set. Our scale-space approach permits the user to analyze the dynamics within the data by selecting an appropriate abstraction levels. Thus, flows that are marginal within the data set are attenuated and the primary motion structure is revealed.

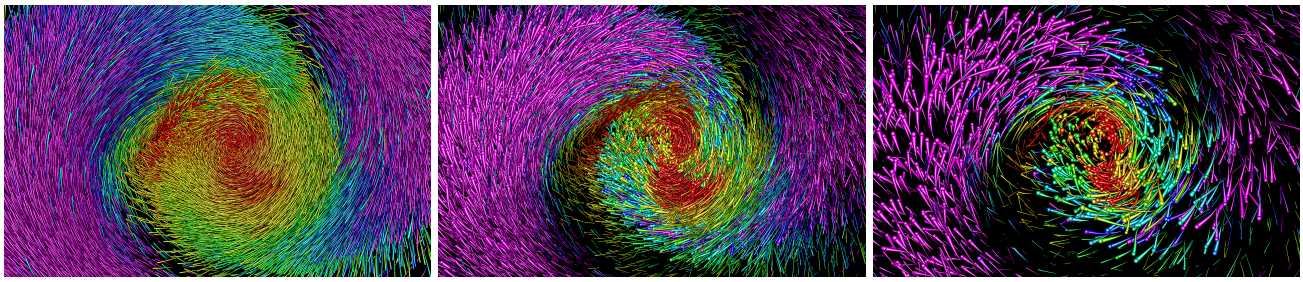

Figure 7. Three levels of detail from our space-time hierarchy, color coded according to the cluster's temperature: 2M particles rendered on a 1440×900 viewport at 3.1 fps (left), 5.6K @ 563 fps (middle), and 1.1K @ 704 fps (right).

The movement of longer time intervals is usually visualized for smaller groups of clusters since the problem of occlusion and visual cluttering naturally increases with the length of the rendered trajectories (compare Figure 6 (middle)). Note, however, that this reduction is only motivated by a better visual impression. Regarding rendering performance and video memory requirements, the worst case of our experiments (except the counter-example for occlusion and visual cluttering in Figure 7 (left)) was 46 fps and 9 MB (Figure 5 (left)), respectively, which is far away from the theoretical throughput of modern GPUs. It is self-evident, that for large data sets occlusions and visual cluttering will impose more severe limitations than rendering throughput. Accordingly, our visualization approach should scale very well with the size of the data set given that a reasonable number and length of trajectories is chosen.

Especially when focusing on the major particle directions at a coarse resolution levels of the proposed space-time hierarchy, detailed information about the underlying mass distribution gets lost (see Figure 8 (middle)). Ray casting the density field, on the other hand, can effectively reveal this distribution in one single time step, but has problems in showing the dynamics with respect to the directional transport (see Figure 8 (left)). By combining volume rendering of scalar particle quantities with the rendering of space-time trajectories, both drawbacks can be addressed (see Figure 8 (right)).

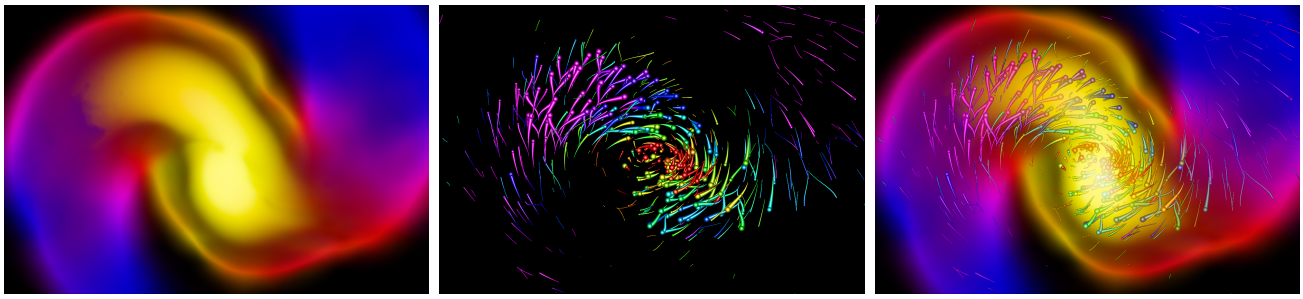

Figure 8. Comparison of volume rendering of the density field (left / 1.5 fps), motion visualization (middle / 900 fps), and a hybrid rendering using both techniques (right / 1.5 fps). All images were rendered on a 1440×900 viewport.

6. CONCLUSION AND FUTURE WORK

We have introduced a hierarchical space-time data structure for time-dependent particle sets that allows to analyze the flow of mass by rendering cluster trajectories at different scales. By choosing an appropriate level of detail, occlusion and visual clutter can be effectively avoided and the primary motion within the simulated data field can be revealed. To focus on regions of interest, arbitrary mapping function can be specified to modulate the appearance of the trajectories based on any particle quantity. Furthermore, sets of clusters can be interactively selected for exclusive rendering or visual highlighting. Combining the rendering of dynamic trajectories with other rendering techniques such as volume ray casting enables to analyze both, the shape of the density field and the underlying flow dynamics at a glance.

To the best of our knowledge, for the first time a scale-space analysis can be applied to the flow of mass within astrophysical data sets containing millions of particles. We offer a new opportunity to visually explore the result of numerical particle simulations and to effectively visualize the dynamics of particle sets in static images. In the future we are interested in applying our space-time hierarchy to particle sets from other fields of fluid dynamics and in adapting our approach for the hierarchical motion analysis of time-dependent vector fields. Furthermore, we plan to integrate focus-and-context techniques to locally adapt the visualization of the trajectories.

ACKNOWLEDGMENTS

The authors wish to thank Rüdiger Pakmor and Fritz Röpke from the Max-Planck-Institute for Astrophysics as well as Volker Springel from the Heidelberg Institute for Theoretical Studies for providing the data sets.

REFERENCES

[1] Desbrun, M. and Cani, M.-P., "Space-time adaptive simulation of highly deformable substances," Tech. Rep. 3829, INRIA, BP 105 - 78153 Le Chesnay Cedex - France (1999).

[2] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J., "Adaptively sampled particle fluids," *ACM Trans. Graph.* **26**(3), 48 (2007).

[3] Post, F. H., Vrolijk, B., Hauser, H., Laramee, R. S., and Doleisch, H., "Feature extraction and visualisation of flow fields," in [*Eurographics 2002 STAR*], Fellner, D. and Scopigno, R., eds., 69–100, Eurographics Association, Saarbrücken Germany (2002).

[4] Laramee, R. S. and Hauser, H., "Geometric flow visualization techniques for cfd simulation data," in [*Proc. 21st spring conference on Computer graphics*], *SCCG '05*, 221–224, ACM, New York, NY, USA (2005).

[5] McLoughlin, T., Laramee, R. S., Peikert, R., Post, F. H., and Chen, M., "Over Two Decades of Integration-Based, Geometric Flow Visualization," *Computer Graphics Forum* **29**(6), 1807–1829 (2010).

[6] Cha, D., Son, S., and Ihm, I., "GPU-assisted high quality particle rendering," *Computer Graphics Forum* **28**(4), 1247 – 1255 (2009).

[7] Navrátil, P. A., Johnson, J. L., and Bromm, V., "Visualization of cosmological particle-based datasets," in [*IEEE Trans. Vis. Comp. Graph. (Proc. IEEE Visualization 2007)*], (2007).

[8] Zhang, Y., Solenthaler, B., and Pajarola, R., "Adaptive sampling and rendering of fluids on the GPU," in [*Symposium on Point-Based Graphics*], 137–146 (2008).

[9] Fraedrich, R., Auer, S., and Westermann, R., "Efficient high-quality volume rendering of sph data," *IEEE Trans. Vis. Comp. Graph. (Proc. Visualization 2010)* **16**(6), 1533–1540 (2010).

[10] Jang, Y., Fuchs, R., Schindler, B., and Peikert, R., "Volumetric evaluation of meshless data from smoothed particle hydrodynamics simulations," in [*Proc. Volume Graphics 2010*], 45–52 (2010).

[11] Hopf, M. and Ertl, T., "Hierarchical splatting of scattered data," in [*Proc. IEEE Visualization 2003*], 443–440 (2003).

[12] Hopf, M., Luttenberger, M., and Ertl, T., "Hierarchical splatting of scattered 4D data," *IEEE Computer Graphics and Applications* **24**(4), 64–72 (2004).

[13] Fraedrich, R., Schneider, J., and Westermann, R., "Exploring the Millenium Run - scalable rendering of large-scale cosmological datasets," *IEEE Trans. Vis. Comp. Graph.* **15**(6), 1251–1258 (2009).

[14] Biddiscombe, J., Geveci, B., Martin, K., Moreland, K., and Thompson, D., "Time dependent processing in a parallel pipeline architecture," *IEEE Trans. Vis. Comp. Graph.* **13**(6), 1376–1383 (2007).

[15] Biddiscombe, J., Graham, D., and Maruzewski, P., "Visualization and analysis of SPH data," *ERCOFTAC Bulletin* **76**, 9–12 (2008).

[16] Ellsworth, D., Green, B., and Moran, P., "Interactive terascale particle visualization," in [*Proc. IEEE Visualization 2004*], 353–360, IEEE Computer Society, Washington, DC, USA (2004).

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

[18] Walker, R., Kenny, P., and Miao, J., "Visualization of smoothed particle hydrodynamics for astrophysics," in [*Theory and Practice of Computer Graphics 2005*], Lever, L. and McDerby, M., eds., 133–138 (2005).

[19] Schindler, B., Fuchs, R., Biddiscombe, J., and Peikert, R., "Predictor-corrector schemes for visualization ofsmoothed particle hydrodynamics data," *IEEE Trans. Vis. Comp. Graph.* **15**, 1243–1250 (2009).

[20] Westermann, R., "Compression domain rendering of time-resolved volume data," in [*Proc. IEEE Visualization 1995*], 168–175 (1995).

[21] Linsen, L., Pascucci, V., Duchaineau, M. A., Hamann, B., and Joy, K. I., "Hierarchical representation of time-varying volume data with "4th-root-of-2" subdivision and quadrilinear b-spline wavelets," in [*Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*], 346–355 (2002).

[22] Woodring, J. and Shen, H.-W., "Semi-automatic time-series transfer functions via temporal clustering and sequencing," *Comput. Graph. Forum* **28**(3), 791–798 (2009).

[23] Woodring, J. and Shen, H.-W., "Multiscale time activity data exploration via temporal clustering visualization spreadsheet," *IEEE Trans. Vis. Comput. Graph.* **15**(1), 123–137 (2009).

[24] Griebel, M., Preusser, T., Rumpf, M., Schweitzer, M. A., and Telea, A., "Flow field clustering via algebraic multigrid," in [*Proc. IEEE Visualization 2004*], 35–42 (2004).

[25] Telea, A. and van Wijk, J. J., "Simplified representation of vector fields," in [*Proc. IEEE Visualization 1999*], (1999).

[26] Garcke, H., Preusser, T., Rumpf, M., Telea, A., Weikard, U., and van Wijk, J., "A continuous clustering method for vector fields," in [*Proc. IEEE Visualization 2000*], (2000).

[27] Heckel, B., Weber, G., Hamann, B., and Joy, K. I., "Construction of vector field hierarchies," in [*Proc. IEEE Visualization 1999*], 19–25 (1999).

[28] Pauly, M., Gross, M., and Kobbelt, L. P., "Efficient simplification of point-sampled surfaces," in [*Proc. IEEE Visualization 2002*], *VIS '02*, 163–170, IEEE Computer Society, Washington, DC, USA (2002).

[29] Lensch, H. P. A., Goesele, M., Kautz, J., Heidrich, W., and Seidel, H.-P., "Image-based reconstruction of spatially varying materials," in [*Proc. 12th EG Workshop on Rendering Techniques*], 103–114 (2001).

[30] Knuth, D. E., [*The Art of Computer Programming. Volume 1 (Second ed.)*], ch. Fundamental Algorithms, 435–455, Addison-Wesley (1997).

[31] Merhof, D., Sonntag, M., Enders, F., Nimsky, C., Hastreiter, P., and Greiner, G., "Hybrid visualization for white matter tracts using triangle strips and point sprites," *IEEE Trans. Vis. Comp. Graph.* **12**, 1181–1188 (2006).