A Novel Approach to Visualizing Dark Matter Simulations

Ralf Kaehler, Oliver Hahn, and Tom Abel

Abstract—In the last decades cosmological N-body dark matter simulations have enabled ab initio studies of the formation of structure in the Universe. Gravity amplified small density fluctuations generated shortly after the Big Bang, leading to the formation of galaxies in the cosmic web. These calculations have led to a growing demand for methods to analyze time-dependent particle based simulations. Rendering methods for such N-body simulation data usually employ some kind of splatting approach via point based rendering primitives and approximate the spatial distributions of physical quantities using kernel interpolation techniques, common in SPH (Smoothed Particle Hydrodynamics)-codes. This paper proposes three GPU-assisted rendering approaches, based on a new, more accurate method to compute the physical densities of dark matter simulation data. It uses full phase-space information to generate a tetrahedral tessellation of the computational domain, with mesh vertices defined by the simulation’s dark matter particle positions. Over time the mesh is deformed by gravitational forces, causing the tetrahedral cells to warp and overlap. The new methods are well suited to visualize the cosmic web. In particular they preserve caustics, regions of high density that emerge, when several streams of dark matter particles share the same location in space, indicating the formation of structures like sheets, filaments and halos. We demonstrate the superior image quality of the new approaches in a comparison with three standard rendering techniques for N-body simulation data.

Index Terms—Astrophysics, dark matter, n-body simulations, tetrahedral grids.

1 INTRODUCTION

Starting with studies of the dynamics of clusters of galaxies by Zwicky in the early 30’s of the last century [35], lots of observational evidence has been gathered, suggesting that the luminous matter in the Universe, including objects like gas clouds and stars, comprises only a tiny fraction of its total mass. Most of the mass in the Universe is thought to be cold dark matter. “Cold” because it moves at non–relativistic speeds and dark because it does not interact with photons, and thus does not emit or absorb light, so that its presence can only be measured through its gravitational influence on ordinary matter. Some promising candidates to explain its nature are provided by particle physics. The most popular is a light neutralino suggested by super–symmetric extensions of the standard model of particle physics.

Dark matter is the key ingredient in the formation of the large-scale structure in the Universe, which arise from small density fluctuations. These are thought to have originated from quantum fluctuations and were stretched to macroscopic scales during an early inflationary epoch shortly after the Big Bang. Dark matter can then be thought of as a gas in which the particles do not collide. To study its evolution, N-body simulations, originally developed in plasma physics and for stellar dynamics, are being used [6, 26]. The outcome of such simulations allow for comparison with observational data of the large–scale distribution of galaxies, as for example the Sloan Digital Sky Survey [1]. Indeed, comparing such simulations with observational data dominates how the standard model of structure formation is being tested. The simulation codes usually treat dark matter as a collisionless gas sampled by a discrete number of tracer particles of equal mass. These are propagated over time by the aggregated gravitational forces acting on each particle. Different numerical methods predominantly differ in how they compute the overall gravitational forces in the computation domain.

Most previous visualizations of such simulations projected each particle separately into screen space, using different kernel profiles and methods to scale the splat sizes, usually based on certain local interpolation schemes for the physical quantities. One method that is particularly popular is based on gathering the nearest n-neighbors for each particle and use adaptive kernel smoothing to obtain a mass den-
sity at each particle position, see e.g. Monaghan [18]. This approach necessarily introduces significant smoothing, especially in regions of low dark matter densities, so-called voids.

We present a rendering approach that is based on an improved way to compute the dark matter density distribution [2, 24]. Instead of operating solely on positional information, it uses the full phase-space information available in N-body dark matter simulations. The computational domain is tessellated using tetrahedral cells that contain equal amounts of mass. The vertices of this mesh are defined by the dark matter particles evolved by the N-body code. The connectivity of the cells is generated once and is kept constant over time. Due to gravitational forces each cell will be warped and at later times many cells will overlap. However, the total mass per cell will stay constant, only its density will change, due to the change of the spatial volume of the cell. Accumulating the density contributions from all cells that fall onto a certain location in the computational domain provides an accurate density estimate for this region. The contributions of this paper can be summarized as follows:

- A data storage and access method that is tailored to the specific properties of the underlying tessellation derived from the tracer particles of N-body dark matter simulations. It allows to generate the complete tessellation, including all connectivity information and derived quantities, like mass density per cell, on-the-fly on the GPU during the rendering pass and thus minimizes the amount of data transferred between CPU and GPU. The method directly extends to datasets that exceed the available graphics memory.

- Three GPU-based rendering methods that exploit this data storage and access scheme, namely (1) a splatting approach that optimally places the splats at the mass centroids of fluid elements and locally scales the kernel sizes based on the correct mass densities at these locations, (2) a mass conserving resampling approach, that does not suffer from problems of slice-based resampling approaches which might miss parts or complete cells that fall between slices and (3) an efficient cell-projection approach that does not require any view-dependent decomposition of the tessellation.

- A comparison of the image quality of these new approaches to the standard rendering methods for N-body dark matter simulations, namely constant and adaptive kernel smoothing, as well as a Voronoi tessellation of the particle distribution.

- A demonstration of the effectiveness of the new approaches to visualize important features of the so-called cosmic web, in particular voids, filaments and dark matter halos.

The remainder of this paper is organized as follows. In Section 2 we will summarize related work in the field of visualization of N-body simulations and direct volume rendering of data on unstructured meshes. Section 3 will review the physical motivation for the tessellation approach, whereas Section 4 focusses on the rendering algorithm and an efficient implementation on current graphics hardware. Section 5 compares the new approaches with standard rendering methods for N-body simulations and we end with concluding remarks and directions for future work in Section 7.

2 Related Work

In the last years, numerous publications have studied the visualization and analysis of point-based datasets from N-body and SPH simulations and the approaches can be divided into two major categories. The first one comprises approaches that operate directly on the points, e.g. by projecting kernel profiles centered at the point locations into the frame-buffer. A GPU-assisted hierarchical splatting of point-based datasets via a PCA clustering procedure has been presented and applied to various N-body and SPH datasets by Hopft et al. [12]. Via compression and out-of-core techniques, this work has been extended to time-dependent N-body datasets [13]. An interactive rendering approach for very large N-body datasets has been presented by Fraedrich et al. [9]. The authors employ a continuous level-of-detail particle representation and a hierarchical quantization scheme to compress the particle coordinates and data attributes. A high performance parallelized algorithm for large-scale astrophysical data sets from particle-based simulations for multicore CPUs and CUDA-enabled GPUs has been presented [14]. Popov et al. [21] employ the Cloud-in-Cell method of PM (Particle-Mesh) schemes to resample point data onto a regular grid to analyze so-called multistream events, which characterize large-scale features like voids, halo and filaments. Haroz et al. [10] apply multidimensional visualization techniques to explore uncertainties of time-dependent cosmological particle datasets. They further present a hardware-accelerated approach for smooth temporal interpolation of the particle data in real-time. Various other point based rendering approaches have been presented [27, 22, 33].

The splatting approach proposed in this paper differs from these in the way the positions and sizes of the splats are computed. The locations are inherently coupled to properties of the underlying physical system, i.e. the volume conservation of phase-space elements and their evolution, respectively deformation over time. The locations of the splats are defined by the centers of mass of each of these volume elements whereas the local splat sizes are directly derived from the physically correct mass density of these elements and not from poor isotropic density estimates based on a quite arbitrary number of nearest neighbors of the tracer particles. This approach can be regarded as some kind of adaptive supersampling that determines the location of the samples by exploiting the underlying physics of the data, whereas a simple, regular supersampling that does not exploit these inherent features of the data would not reach the same image quality.

Work in the second category employs some kind of proxy grid, for example by resampling the point-based data to regular grid. A GPU-assisted resampling approach for unstructured point data is discussed by Fraedrich et al. [8]. It adaptively discretizes the view-volume onto a 3D texture, based on the distance to the current camera position. A GPU-assisted mapping of input particles into a volumetric density field using an adaptive density estimation technique that iteratively adapts the smoothing length for local grid cells has been presented by Cha et al. [5]. Another GPU-assisted resampling approach of point-data is discussed in Zhu et al. [34]. A method to obtain velocity field statistics from N-body simulations by generating Voronoi and Delaunay tessellations has been presented by Bernardelou et al. [3]

Our resampling approach differs from these in the sense that it does not operate directly on the points primitives, but uses a tetrahedral mesh that is derived from them. The mesh is neither a Voronoi nor a Delaunay tessellation of the computational domain, but is rather based on the regular layout of the points that N-body simulations use as initial conditions.

There has also been extensive work on the visualization of data on tetrahedral grids. Cell-projection methods usually employ the Projected Tetrahedra (PT) algorithm, that decomposes each tetrahedron into a set of triangles and assigns scalar values for the entry and exit points of the viewing rays to each vertex [25]. A GPU-assisted method for decomposing the tetrahedra into triangles using the PT algorithm was presented by Wylie et al. [32]. An artifact-free PT rendering approach using a logarithmically scaled pre-integration table was proposed by Kraus et al. [16]. Maximo et al. developed a hardware-assisted PT approach using CUDA for visibility sorting [17]. GPU-assisted raycasting methods for tetrahedral grids have, for example, been discussed by Weiler et al. [30] and Espinha et al. [7].

We could employ these cell-projection approaches to perform the rendering of the densities defined on our tetrahedral grid structure. However, due to the specific problem we are focussing on, i.e. high-quality density projections of N-body dark matter simulation data, we can provide a more efficient and much easier GPU-implementation that exploits the order independency and the implicit connectivity information given in this case and does not require the generation of any view-dependent decompositions of the tetrahedra faces or any intersection computations.

An alternative method to render tetrahedral grids is to resample
the data to grid structures that are more directly supported by current graphic hardware architectures. Westermann et al. [31] presented a multi-pass algorithm that resamples tetrahedral meshes onto a cartesian grid by efficiently determining the intersections between planes through the centers of slabs of cells of the target grid using the ST (Shirley-Tuchman)-classification and OpenGL’s alpha test to reject fragments outside the intersection regions. Weiler et al. [29] proposed a slice-based resampling technique to a multi-resolution grid. It discards fragments outside the intersection regions between the slice and the tetrahedra based on the barycentric coordinates of each fragment, which are obtained from a texture-lookup.

The slice-based approach is problematic in our case, as it might miss small or degenerated tetrahedra that fall between two slices. We need to distribute conserved quantities like the total mass of the tetrahedron into the cells of the target grids. The resampling algorithm we propose, estimates the volume of the intersection between the tetrahedral elements and the cubical cells and distributes the mass based on this information. It is easy to implement and does not require the generation of view-dependent decompositions of the tetrahedra faces or additional texture-lookup tables to discard fragments outside the intersection.

3 Motivation

In this section we discuss the theoretical background of the rendering methods proposed in this paper. For a more detailed discussion of the physics the reader might refer to [2, 24].

N-body simulations modeling the evolution of dark matter distributions usually discretize the computational domain by a constant number of point-like mass sources, so-called tracer particles. To reduce the computational complexity, each tracer particle represents large ensembles of physical dark matter particles, typically between 10^2 and 10^5 solar masses. Initial conditions are generated by distributing the tracer particles at the nodes of a cubical grid and imposing small perturbations on their positions and velocities according to the statistics of density fluctuations in the early Universe, as imprinted in the CMB (Cosmic Microwave Background) radiation. The position of each particle \( i \) is updated by computing the aggregate gravitational forces of all other particles \( j, j \neq i \) at the location of \( i \), and changing \( i \)’s position according to the acceleration resulting from this net force. In this process, the mass of the physical dark matter particles represented by the tracers is usually treated as if it was centered around the tracer’s position.

It is important to emphasize that the tracers do not have a direct physical equivalent, but are basically approximations introduced to keep the computational complexity of the simulations manageable. Even with these simplifications large-scale N-body dark matter simulations nowadays follow the motion of up to hundred billion tracer particles, see e.g. [4, 15]. Nevertheless, it is physically more accurate to regard the tracer particle’s mass as being spread out over the computational domain, instead of being concentrated at a set of discrete sampling locations.

The correct time-dependent evolution of an ensemble of dark matter particles is given by the collisionless Boltzmann equation, also called the Vlasov-Poisson equation [20]

\[
\frac{\partial f}{\partial t} = - v \nabla_x f - \nabla_x \phi \cdot \nabla_v f,
\]

where \( \phi \) is the gravitational potential of the system. The distribution function \( f(x, v, t) \) describes the phase-space density of the ensemble, and is defined such that \( dN = f(x, v, t) \, dx \, dv \) is the number of particles that at time \( t \) have positions between \( x \) and \( x + dx \) and velocities between \( v \) and \( v + dv \). Given \( f \), the number of dark matter particles per unit volume \( n(x, t) \) at \( x \) is \( n(x, t) = \int f(x, v, t) \, dv \), and analogously the mass density \( \rho(x, t) \) is

\[
\rho(x, t) = \int m f(x, v, t) \, dv,
\]

where \( m \) is the particle mass.

3.1 Tessellation of the Computational Domain

To illustrate how this motivates a new method to estimate the physical quantities associated with N-body dark matter simulations, consider the 2-dimensional phase-space diagram in Figure 2, that shows the location of the fluid elements on the horizontal axis, versus their velocities on the vertical axis.

Fig. 2: This 2D phase-space diagram shows the positions and velocities of the dark matter fluid for three different time-steps, in the order of red, green and blue. At the latest time depicted, several fluid elements occupy the same location in space (transparent shaded region). The dots on the lines indicate the location of the tracer particles used by N-body simulations to trace the motion of the dark matter fluid over time.

At early times, the dark matter fluid is almost uniformly distributed and as depicted by the red line in Figure 2. Over time, gravity accelerates the dark matter fluid elements and they gain velocity, denoted by the green line. At later times different streams of dark matter co-exist in the same spatial regions, in this example there are three regions per spatial location for elements on the blue line inside the transparent box. These so-called multi-stream regions provide important information about the formation of structures in the dark matter distribution on large spatial scales. The number of streams can be used to identify regions of very low matter density, so-called voids, as well as sheets, filaments and halos, which together form the so-called cosmic web. Voids correspond to regions with only one stream of dark matter particles, sheet-like structures can be defined by the existence of three streams, and higher values indicate the formation of filaments and dark matter halos, the locations where galaxies form.

The dots on the lines correspond to the tracer particles used by the simulation to sample the motion of the collisionless dark matter fluid over time. At the initial time step, the tracer particles are distributed uniformly in the computational domain and their positions define the vertices of a cubical grid in the 3D case, or squares as depicted for a 2D example in the left image of Figure 3. Since each cell initially has the same volume and the mass distribution is nearly homogeneous over the computational domain, it is physically reasonable to assign a constant distribution function \( f(x, v, t) \) and thus constant mass \( m \) to each cell \( C_i \). Over time the N-body code will update the tracer particles according to the gravitational forces acting in the computational domain and
this will cause the initial cubical grid cells to be deformed, as depicted in the middle image of Figure 3. At later times, when gravity induced even larger inhomogeneities in the matter distribution, the motion of the tracers leads to large numbers of cells overlapping in the spatial domain, as shown in the right image of Figure 3. The crucial observation for estimating physical quantities is based on the conservation of mass, that states that the mass of each co-moving volume element is constant over time. Thus, from the knowledge of the constant initial mass distribution and the time-dependent volume of each cell, derived quantities like mass densities can be computed for all times.

In principle the cubical tessellation could be employed to obtain this information, but the non-convex cells that emerge during the deformation of the grid, as shown in the right image of Figure 3, would complicate the computation of the time-dependent volumes. A preferable domain tessellation is obtained using tetrahedral elements. The advantage of this cell type is that independently of the relative motions of the vertices, these cells will remain convex, though the cells might temporarily become degenerate, when all vertices (almost) lie in the same plane. Tetrahedra with small volumes indicate regions of high mass density, since the mass per tetrahedron is constant by construction. These high-density tetrahedra indicate caustics in the dark matter fluid.

Fig. 4: The decomposition of a cubical cell into six non-overlapping tetrahedra of constant volume used in this paper. This configuration introduces no new vertices besides the tracer particles of the simulation and ensures consistent edges and faces for abutting cells.

The tessellation of the initial cubical cells should consist of tetrahedral elements that introduce no new vertices, ensure consistent faces and edges between abutting cells, and initially have identical volumes. The smallest number of elements that fulfill these constraints is six, and we chose the configuration shown in Figure 4. This choice ensures that no holes or cracks will form in the interior region of the mesh over time, even as the grid gets vastly deformed, because the vertices and edges on shared faces match up.

The connectivity that defines the tetrahedra is kept the same for all time steps. Only the spatial positions of the vertices are updated according to the actual positions of the tracer particles, as computed by the N-body simulation. This implies that for a new time step, only the positional information of the vertices must be updated, while the connectivity information can be reused. The identification of corresponding particles for different time-steps is done with help of the unique IDs that simulation codes assign to the tracers. These can be mapped to the coordinates of the particles on the initial grid, see Equation (8).

### 3.2 Density Projections

According to the discussion in Section 3.1, the local mass density of a co-moving tetrahedron \( t \) at time \( t \) is given by

\[
\rho_{\text{loc}}(t) = \frac{m}{V_i(t)},
\]

i.e. the constant mass \( m \) divided by its time-dependent volume \( V_i(t) \). Over time, the motion of the tracer particles results in large amounts of overlapping volume elements, and according to Equation (2) the total mass density \( \rho_{\text{tot}}(x,t) \) at a position \( x \) is simply given by the sum of the densities of each cell containing \( x \) at that time. To illustrate this, consider the volume element \( V_i(x,t) \) that is obtained by intersecting all cells that contain \( x \) at time \( t \). The total density is then defined by

\[
\rho_{\text{tot}}(x,t) = \sum_i \frac{m_{i\cap}}{V_i(x,t)}.
\]

where the sum runs over all cells that contain \( x \), and \( m_{i\cap} \) is the amount of mass per cell \( i \) contained in subvolume \( V_i \). Using constant spatial interpolation, we get \( m_{i\cap} = m \frac{V_i}{V(t)} \), where \( m \) is the constant mass per cell and \( V_i \) is the cell’s volume. Combining this with Equation (4) we get

\[
\rho_{\text{tot}}(x,t) = \sum_i \rho_{\text{loc}}(t),
\]

where the sum runs over all cells \( i \) that contain \( x \) at time \( t \). The number of dark matter streams at a certain spatial location, which as discussed above can be used to distinguish different regions of the cosmic web, corresponds to the number of overlapping tetrahedra.

Relevant for many scenarios in astrophysics and cosmology are projections of certain physical quantities \( q \) along the line of sight

\[
q_{\text{proj}} = \int_x q(x(\chi)) \, d\chi,
\]

where \( x(\chi) \) denotes the parametrization of that line for a certain pixel on the screen. For the tetrahedral mesh discussion above, it takes the form

\[
q_{\text{proj}} = \sum_I \bar{q}_I d_I,
\]

where the index \( i \) runs over all tetrahedra \( T \) that are intersected by the line of sight, \( \bar{q}_I \) is the constant quantity associated with \( T_i \) and \( d_I \) denotes the length of the intersections between the line of sight and \( T_i \). Particularly important are density projections

\[
\rho_{\text{proj}} = \int_x g(\rho(x(\chi))) \, d\chi,
\]

where \( g \) is some function of the density \( \rho \). \( n = 1 \) for example is relevant for experiments aiming at detecting dark matter directly in underground detectors.

The discussion in this section can be summarized as follows: Given a time-dependent 3D N-body dark matter simulation, a tetrahedral mesh is constructed, with a connectivity implicitly defined by the layout of the tracers on a regular grid at the initial time-step, which can be reconstructed at any time step from the tracer’s unique IDs. The same amount of mass is assigned to each tetrahedral element and derived quantities, like time-dependent mass densities, are computed based on the volumes of all tetrahedral elements that overlap a certain location, see Equation (5). The mass is associated with the cells and not the vertices of the tessellation. The nodes of the mesh are updated over time, according to the tracer’s actual position, changing the volumes and thus the spatial mass densities. The tessellation has consistent vertices, edges and faces for abutting cells, and in particular does not contain any dangling nodes, but at later times the tetrahedral elements will typically start to overlap.

### 4 Rendering

In the following we discuss three GPU-based rendering approaches for density projections generated from this type of input mesh. We implemented them in OpenGL and the OpenGL Shading Language, and we will use OpenGL nomenclature in the following.

#### 4.1 Data Storage and Access on the GPU

The relation between a tracer’s ID and its vertex \((i,j,k)\) on the initial regular grid is given by

\[
\text{id} = i + d_0(j + d_1 k)
\]

where \( d_0, d_1, d_2 \) are the number of vertices of the grid along the \( x, y \) and \( z \) direction. The implicit connectivity of the initial grid allows for
a very memory efficient representation of the mesh on the GPU without the need to store and transfer any explicit connectivity information or additional attributes about the tetrahedral cells, except for the locations of the tracer particles. All connectivity and derived information, like the volumes and mass densities of the tetrahedral cells can be generated on-the-fly on the GPU.

The tracer particle positions are stored in a three-dimensional floating-point RGB texture with \( d_0 d_1 d_2 \) texels, so that several particles can be accessed in the vertex shader instance. The texel coordinates \((i, j, k)\) are derived from the particle’s ID, according to Equation (8). The texture is uploaded onto the GPU and sampled in a vertex shader, which is invoked \((d_0 - 1) \times (d_1 - 1) \times (d_2 - 1)\) times via instance rendering \((\text{glDrawElementsInstanced(}...\))\). The current invocation ID is obtained from the instance counter \(\text{gl_InstanceID}\) in the vertex shader mapped to texel coordinate \((i, j, k)\), see Equation (8). The coordinates \(v_i, i = 0...7\) of the eight tracers stored at \((i \pm 1, j \pm 1, k \pm 1)\), defining a cubical cell in the initial regular grid, are read from the 3D position texture and handed over to a geometry shader as varying attributes. In the geometry shader, the six (possibly deformed) tetrahedra depicted in Figure 4, are constructed from the tracer’s positions via the connectivity table \(\{1, 0, 2, 4\}, \{3, 1, 2, 4\}, \{3, 5, 1, 4\}, \{3, 6, 5, 4\}, \{3, 2, 6, 4\}, \{3, 7, 5, 6\}\), and the volume of each tetrahedron is computed by

\[
V = \frac{(v_1 - v_0) \cdot ((v_2 - v_0) \times (v_3 - v_0))}{6}.
\]  

In order to better leverage the massive parallelism of current GPU architectures, we do not generate all six tetrahedra in the same geometry shader instance, but rather trigger six geometry shader invocations for each vertex shader instance via the “invocations” layout qualifier of the OpenGL Shading Language “layout ( points, invocations = 6 ) in;” The built-in variable \(\text{gl_InvocationID}\) is used to determine which of the six tetrahedra to be generated in a certain geometry shader instance.

For datasets that exceed the available graphics memory, we decompose the 3D texture that stores the positional information into separate blocks, each of them small enough to fit entirely into graphics memory. The blocks share a layer of texels at their interfaces, and are transferred and processed individually on the GPU, simply accumulating the partial rendering results for the tetrahedral elements encoded in each brick.

In the following subsections, we will discuss three different GPU-assisted rendering approaches that are based on this data storage and access strategy. We will focus on the rendering of density projections, see Equation (7), which are order-independent, so no sorting of the rendering primitives is required.

4.2 Centroids

In this approach, each tetrahedron \(T\) is rendered using a 2D billboard with a cubic-spline kernel, oriented perpendicular to the current viewing direction and located at \(T\)’s centroid \(c_f = \frac{1}{6} \sum_{i=0}^{6} v_i\), where the \(v_i\) denote \(T\)’s vertices. Since we are assuming constant mass (density) per cell, the centroid is identical to \(T\)’s center of mass. The kernels are scaled proportional to \(\sqrt{\rho_T}\), where \(\rho_T\) denotes the mass density of \(T\), that is computed in the geometry shader according to Equation (9), along with the vertices and texture coordinates for the billboards. The contribution of each generated fragment is accumulated in a floating-point 2D texture that is bound as a render target, using an additive blending equation.

The centroids and the sizes of the quadrilaterals could be computed in a preprocessing step and cached on the GPU using vertex buffer objects (VBOs). However, since we have about 6 times more centroids than tracer particles, storing the centroids along with the
density-dependent scaling factors would result in a considerable increase of bandwidth and graphics memory consumption. It is thus preferable to generate this information on-the-fly on the GPU.

4.3 Resampling

The second approach we propose is to resample the tetrahedral cells to a cubical grid structure \( G \), which allows to locally evaluate Equation (4) and to apply standard rendering methods for regular grids, e.g. to display level sets of the data. The mass of each tetrahedral element \( T_i \) needs to be distributed to the cubical cells \( C_j \) of \( G \). A correct solution involves the computation of the volume of the intersections between \( T_i \) and the cubical cells \( C_j \) and an assignment of the mass contribution based on its volume, see Equation (3). For the hundreds of millions of tetrahedra in typical N-body simulations, this procedure would exceed the limit of the graphics hardware. We therefore propose two approaches for density projections from a dark matter N-body simulation.

The second approach we propose is to resample the tetrahedral cells to a cubical grid structure \( G \), which allows to locally evaluate Equation (4) and to apply standard rendering methods for regular grids, e.g. to display level sets of the data. The mass of each tetrahedral element \( T_i \) needs to be distributed to the cubical cells \( C_j \) of \( G \). A correct solution involves the computation of the volume of the intersections between \( T_i \) and the cubical cells \( C_j \) and an assignment of the mass contribution based on its volume, see Equation (3). For the hundreds of millions of tetrahedra in typical N-body simulations, this procedure would exceed the limit of the graphics hardware. We therefore propose two approaches for density projections from a dark matter N-body simulation.

The second approach we propose is to resample the tetrahedral cells to a cubical grid structure \( G \), which allows to locally evaluate Equation (4) and to apply standard rendering methods for regular grids, e.g. to display level sets of the data. The mass of each tetrahedral element \( T_i \) needs to be distributed to the cubical cells \( C_j \) of \( G \). A correct solution involves the computation of the volume of the intersections between \( T_i \) and the cubical cells \( C_j \) and an assignment of the mass contribution based on its volume, see Equation (3). For the hundreds of millions of tetrahedra in typical N-body simulations, this procedure would exceed the limit of the graphics hardware. We therefore propose two approaches for density projections from a dark matter N-body simulation.

4.4 Cell-Projection

The third rendering method evaluates Equation (6) with a cell-projection method, computing the contributions \( \bar{q}_dl \) for each tetrahedron \( T_i \). Rewriting Equation (6) as

\[
q_{proj} = \sum_i \bar{q}_i |b_i - f_i| = (\sum_i \bar{q}_i |b_i - c|) - (\sum_i \bar{q}_i |c - f_i|),
\]

where \( f_i \) and \( b_i \) denote the entry and exit points of the line of sight for \( T_i \) and \( c \) is the current camera location shows that Equation (6) can be evaluated efficiently by separately adding the contributions of the front-facing and back-facing triangles.

As in the previously discussed approaches, the vertices \( v_i \) of each deformed cell are obtained by sampling the position texture in the vertex shader and the six tetrahedra are constructed in the geometry shader. The faces of the tetrahedra are rendered as triangle strips. A negative value of the volume formula Equation (9) indicates that the tetrahedron is inverted, and the order of vertices in the strip has to be adjusted to ensure consistent face orientations. The mass densities are computed from the volumes and handed over to the fragment shader as varying variables.

In the fragment shader, the contributions \( \bar{q}_dl \) are computed for each tetrahedron \( T_i \). Therefore, the blending equation is set to the additive blending equation \( C_{src} + C_{dest} \), and the fragment shader stores the contributions for fragments of front-facing triangles \( \sum \bar{q}_i |f_i - c| \) in the red channel of the frame-buffer, and the contributions for back-facing fragments \( \sum \bar{q}_i |b_i - c| \) in the green channel. After the triangles for all \( T_i \) are processed, a separate fragment shader computes the final sum in Equation (13) by subtracting the partial sums that are stored in the red and green channels and the result for each pixel is written into the frame-buffer.

5 Results

The comparison was performed using a Nvidia Quadro 6000 graphics card with 6 GByte of graphics memory, that was installed on a host with an Intel Xeon E5520 CPU and 24 GByte main memory. The rendering algorithms were implemented in OpenGL and the OpenGL Shading Language.

Figure 5 shows a rendering of a dark matter simulation with 134 million tracer particles using the cell-projection approach from Section 4.4. Figure 8 shows a visualization of the multi-stream field, that counts the number of dark matter streams at each location in the computational domain. The data were resampled to a 512\(^3\) grid, setting \( q_{i} = 1 \) in Equations (10) and (11), to count the number of tetrahedra per cell, and rendered using a GPU-assisted ray-caster. Voids, shown in blue, sheet-like structures (red) and filaments (white), can be clearly distinguished.

We further compared the image quality and performance of the three rendering methods proposed in Section 4 with three conventional approaches for density projections from a dark matter N-body simulation with 17.2 million particles, see Figure 9. The screen size was
Fig. 7: A direct comparison between our tetrahedral cell-projection approach (left) and a standard SPH adaptive kernel smoothing method. Artifacts due to the poor density estimates in low-density regions are obvious for the SPH method, whereas the tetrahedral approach achieves an overall high image quality, on small and large structures.

1460x860 pixels. In Figure 8 an emission-absorption scheme was chosen as the lighting model. In all other rendering examples shown in this paper, the resulting total mass density in the framebuffer was logarithmically rescaled in a separate fragment shader and mapped to colors via a 1D-texture lookup.

Close-ups of the resulting images are shown in Figure 9. For images a) and b), the tracer particles were rendered using cubic spline kernels, accumulating the contributions in the framebuffer. The positions were cached on the GPU using vertex-buffer arrays and the geometry for the view-port aligned billboards was generated in the geometry shader. For image a), constant kernel sizes were used for all particles. For image b), adaptive kernel sizes based on local density estimates were projected. These were obtained from the smallest spheres enclosing the 32 nearest neighbors around each particle, a standard approach in SPH simulations, as for example discussed by Springel et al. [26].

The sizes of the kernels were scaled proportional to $\rho_i^{-1/3}$, where $\rho_i$ denotes the resulting density estimation for particle $i$. The adaptive kernel sizes were computed on the CPU using a kd-tree search tree and cached on the GPU along with the positions using vertex-buffer arrays.

For image c), the Voronoi tessellation of the 17.1 million dark matter tracer particles was generated using the Voron++ library [23]. The density around each particle was computed from the volume of its Voronoi cell. The resulting density projection was generated via a cell-projection approach. Therefore, the cell faces were rendered separately in the GL_POLYGON mode. In the fragment shader, the fragment’s distance $d$ to the camera location was computed and $d \rho$ was written to the red-channel for front-facing fragments, and respectively to the green-channel for back-facing fragments. After all cells were processed, the difference between the red and green channels was written to the image buffer, yielding the line integral of the density, as discussed in Section 4. For ease of comparison, we have used the same cubic spline profile for the six times more numerous tetrahedron centroids. Image e) was generated by resampling the $6 \times 17.1 = 102.6$ million tetrahedral elements onto a regular grid with $512^3$ cells using the resampling approach discussed in Section 4.3. The resulting grid was rendered using a standard GPU-raycasting approach. Finally, image f) in Figure 9 was generated via the cell-projection approach for the 102.6 million tetrahedra, as discussed in Section 4.4. Figure 7 shows another direct comparison between the tetrahedral cell-projection approach and the SPH adaptive kernel smoothing method. An overview about the preprocessing times, the memory requirements and performance numbers are summarized in Table 1.

6 Discussion

The images in Figures 7 and 9 clearly demonstrate the improved image quality of the new rendering methods, as depicted in images d) to f). Especially image f) and Figure 7 show that the proposed cell-projection approach achieves very high image quality, both in areas of homogeneous densities, for example in the filaments emerging from the central halo, and at the same time reveal significantly more fine-scale details in the central region of the dark matter halo. Caustics, formed at the locations where orbits of the dark matter fluid turn

|                | a   | b   | c   | d   | e   | f   |
|----------------|-----|-----|-----|-----|-----|-----|
| memory [GBytes]| 0.20| 0.26| 2.80| 0.20| 1.7 | 0.2 |
| preprocessing [s]| 0.2 | 185 | 962 | 0.2 | 52  | 0.2 |
| performance [fps]| 3.0 | 3.0 | 0.001| 0.5 | 2.2 | 0.1 |
and shows no sign of artificial clumping. From these images the filaments shown in Figure 7 to be perfectly smooth and bounded correctly. Clumps originate. Our full tetrahedral projection method has we use here is physically accurate and shows directly where such in- naturally softened gravitational point masses. Remarkably, the approach in the gravitational forces by assuming that the particles are isotropi-
ically thin slices had been missing. This capability allows also for a much closer visual comparison with the hydrodynamics properties of the gaseous matter typically evolved at the same time as the dark matter in the most sophisticated computational calculations.

The images generated with the new rendering methods can be directly used as input for predictions of the gravitational lensing effect (cf. e.g. [11]). Images of the mass density directly correspond to so-called convergence maps, but also so-called shear maps can be computed in a straight-forward way once an image is at hand. The clear advantage over previous approaches is the low noise level of our images that does not come at the price of a large isotropic filtering that washes out relevant small-scale structure. We are currently working on using the rendered images for this purpose.

6.2 Scalability to Large Data
As discussed in Section 4.3, the rendering methods presented in this paper also extend to datasets that exceed the available graphics memory. In this case, the 3D texture used to store the positional information is decomposed into separate sub-bricks, each of them small enough to fit entirely into graphics memory. It would be straight-forward to apply this technique to run the algorithms on a GPU-cluster, by distributing the separate bricks to the individual cluster-nodes. Each brick could be processed in parallel and the partial density projections would be added to obtain the final rendering result. The choice between the three different methods allows for a trade-off between performance and image quality for example by choosing the high quality cell-projection methods for regions close to the camera and the faster centroid approach for regions in the far field. This decision can be made on-the-fly in the geometry shader based on the distance of the point coordinates to the camera. An example of this is shown in Figure 5.

Alternatively and/or in addition to this, a multi-resolution hierarch, for example an octree, can be constructed from the full-resolution 3D position texture. A texel on the first coarser level would store the center of mass as well as the averaged density of all tetrahedral elements represented by the texels on the highest level of resolution. The following coarser levels could then be constructed from these using techniques like for example discussed in Fraedrich et al. [9]. Again, regions close to the camera would be rendered via the cell-projection approach using the original resolution of the texture, whereas regions in the far field would be approximated using splatting techniques for the coarser resolution textures.

7 Conclusions
We presented three GPU-accelerated rendering approaches for N-body dark matter simulation data, based on a tetrahedral decomposition of the computational domain that allows a physically more accurate estimation of the mass density between the tracer particles than previous methods. They use the full phase-space information of the ensemble of dark matter tracer particles and two of them minimize pre-processing time (centroids and cell-projection) as well as data transfer between the CPU and GPU, by generating all connectivity information as well as the derived quantities, like mass density of the tetrahedral mesh elements, on the GPU. Thus these approaches are particularly well suited for time-dependent data. Their performance should benefit significantly from the increased number of cores expected for future generations of graphics hardware.
Fig. 9: Comparison between the visualizations of a dark matter halo simulation using three conventional techniques, namely *constant kernel smoothing* (a), *adaptive kernel smoothing* (b), *voronoi tessellation* (c), and the three new rendering methods based on the tetrahedral phase-space tessellation proposed in this paper, i.e. *centroids* (d), *resampling* (e) and *cell-projection* (f). The subimage on the right shows a close-up of the rectangular region of the left.
We compared these new methods to three standard rendering approaches for dark matter simulations: two based on constant and adaptive kernel sizes that estimate the local densities from the nearest-neighbors, as well as a Voronoi tessellation generated by the simulations tracers particles. We showed that our approaches yield considerably better image quality with less pre-processing times and graphics memory requirements. The full tetrahedral cell-projection methods clearly stands apart, however. Without artificial smoothing or density estimates derived from averaging over the particle distribution, features previously washed out, become clearly visible and give new insight in the physical large-scale features of the cosmic web, including voids, filaments and halos.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation through award number AST-0808398 and the LDRD program at the SLAC National Accelerator Laboratory as well as the Terman fellowship at Stanford University.

REFERENCES

[1] K. Abazajian, Z. Zheng, I. Zehavi, D. H. Weinberg, J. A. Frieman, A. A. Berlind, M. R. Blanton, N. A. Bahcall, J. Brinkmann, D. P. Schneider, Cosmology and the Halo Occupation Distribution from Small-Scale Galaxy Clustering in the Sloan Digital Sky Survey. *apj*, 625:613–620, June 2005.
[2] T. Abel, O. Hahn, and R. Kaehler. Tracing the Dark Matter Sheet in Phase Space. *arXiv*, 1111.3944v2, Nov. 2011.
[3] P. Bernardage and R. van de Weygaert. A New Method for Accurate Estimation of Velocity Field Statistics. *MNRAS*, 279:693, Mar. 1996.
[4] M. Boylan-Kolchin, V. Springel, S. D. M. White, A. Jenkins, and G. Lemson. Resolving cosmic structure formation with the Millennium-II Simulation. *mnras*, 398:1150–1164, Sept. 2009.
[5] D. Cha, S. Son, and I. Ihm. GPU-Assisted High Quality Particle Rendering. *Comput. Graph. Forum*, 28(4):1247–1255, 2009.
[6] G. Efthathiou, M. Davis, S. D. M. White, and C. S. Frenk. Numerical techniques for large cosmological N-body simulations. *apj*, 57:241–260, Feb. 1985.
[7] R. Espinosa and W. Celso. High-Quality Hardware-Based Ray-Casting Volume Rendering Using Partial Pre-Integration. In *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, pages 273–, Washington, DC, USA, 2005. IEEE Computer Society.
[8] R. Fraedrich, S. Auer, and R. Westermann. Efficient High-Quality Volume Rendering of SPH Data. *IEEE Transactions on Visualization and Computer Graphics*, 16:1533–1540, November 2010.
[9] R. Fraedrich, J. Schneider, and R. Westermann. Exploring the Millennium Run - Scalable Rendering of Large-Scale Cosmological Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 15:1251–1258, November 2009.
[10] S. Haroz, K.-L. Ma, and K. Heitmann. Multiple uncertainties in time-variant cosmological particle data. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 207–214. IEEE VGTC, March 2008.
[11] S. Hilbert, J. Hartlap, S. D. M. White, and P. Schneider. Ray-tracing through the Millennium Simulation: Born corrections and lens-lens coupling in cosmic shear and galaxy-galaxy lensing. *A&A*, 499:31–43, May 2009.
[12] M. Hopf and T. Ertl. Hierarchical Splatting of Scattered Data. In *Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*, VIS ’03, pages 57–, Washington, DC, USA, 2003. IEEE Computer Society.
[13] M. Hopf, M. Luttenberger, and T. Ertl. Hierarchical Splatting of Scattered 4D Data. *IEEE Comput. Graph. Appl.*, 24:64–72, July 2004.
[14] Z. Jin, M. Krokos, M. Rivi, C. Gheller, K. Dolag, and M. Reinecke. High-performance Astrophysical Visualization using Splotch. *Proceedia CS*, 1(1):1775–1784, 2010.
[15] A. A. Klypin, S. Trujillo-Gomez, and J. Primack. Dark matter halos in the standard cosmological model: Results from the bolshoi simulation. *The Astrophysical Journal*, 740(2):102, 2011.
[16] M. Kraus, W. Qiao, and D. S. Ebert. Projecting tetrahedra without rendering artifacts. In *Proceedings of the conference on Visualization ’04*, VIS ’04, pages 27–34, Washington, DC, USA, 2004. IEEE Computer Society.
[17] A. Maximo, R. Marroquim, and R. Farias. Hardware-Assisted Projected Tetrahedra. *Computer Graphics Forum*, 29(3):903–912, 2010.