DOI: 10.1111/cgf.14121

COMPUTER GRAPHICS *forum*
Volume 40 (2021), number 1 pp. 97–109

Stochastic Volume Rendering of Multi-Phase SPH Data

M. Piochowiak, T. Rapp and C. Dachsbacher

Karlsruhe Institute of Technology (KIT), Germany
tobias.rapp@kit.edu, dachsbacher@kit.edu

Abstract

In this paper, we present a novel method for the direct volume rendering of large smoothed-particle hydrodynamics (SPH) simulation data without transforming the unstructured data to an intermediate representation. By directly visualizing the unstructured particle data, we avoid long preprocessing times and large storage requirements. This enables the visualization of large, time-dependent, and multivariate data both as a post-process and in situ. To address the computational complexity, we introduce stochastic volume rendering that considers only a subset of particles at each step during ray marching. The sample probabilities for selecting this subset at each step are thereby determined both in a view-dependent manner and based on the spatial complexity of the data. Our stochastic volume rendering enables us to scale continuously from a fast, interactive preview to a more accurate volume rendering at higher cost. Lastly, we discuss the visualization of free-surface and multi-phase flows by including a multi-material model with volumetric and surface shading into the stochastic volume rendering.

Keywords: volume visualization, visualization, volume rendering, rendering, scientific visualization, visualization

ACM CCS: • Human-centred computing → Scientific visualization; • Computer graphics → Ray tracing

1. Introduction

In recent years, particle-based simulation methods, especially the smoothed-particle hydrodynamics (SPH) method, have become popular in science and engineering. They are especially well-suited for free-surface flows [Mon94] and multi-phase simulations [MK95] that contain two or more distinct fluid phases, for example, a liquid and a gas phase. In this work, we focus on multi-phase SPH simulations, but our method is similarly applicable to other particle-based data, where fluid attributes are evaluated by superimposing kernel functions. The visualization of such datasets thus requires the reconstruction of the continuous fluid attributes from the unstructured particle representation. Since current datasets contain millions of particles per time step, this is computationally expensive and often forms the bottleneck for scientists and engineers visually analyzing and exploring their data. Most commonly, the data is transformed to an intermediate representation, such as an uniform grid, on which surface extraction [NJB07, SSP07, AAIT12] or direct-volume rendering [CSI09, FAW10, RTW13] can be efficiently performed. However, the accuracy and flexibility is thus inherently limited by the intermediate representation. In contrast, most approaches that avoid costly preprocessing are limited to basic rendering techniques [KSN08, JFSP10].

We propose a novel approach for the visualization of particle data which requires only little preprocessing and directly uses the unstructured data for volume rendering. This enables the exploration of large, time-dependent, and multivariate particle data, where costly preprocessing would be too time-consuming. At the same time, our approach scales from a faster, interactive preview up to a more accurate visualization of the data. To address the computational complexity and enable a continuous adjustment of the rendering quality, we apply stochastic sampling during volume rendering to select only a subset of particles at each step during ray marching. The number of selected particles on the one hand depends on the desired quality. On the other hand, in order to faithfully reproduce important details, the selection also considers an importance measure derived from the data values. In particular, we introduce a measure based on the local entropy of the data that is combined with view-dependent information. Since our approach requires only the unstructured particle data and an importance measure, it is well-suited for *in situ* applications, e.g. to track the progress of a simulation, where computing and storing large intermediate representations would be infeasible.

We specifically consider the visualization of multi-phase fluid data by reconstructing the interfaces between different types of

fluids, e.g. between gas and fluid particles. To visualize different phases and their interfaces, we employ a multi-material model that includes both volumetric and surface shading. To reconstruct accurate surfaces along the phase interfaces, we adapt our importance measure accordingly. Lastly, we employ stochastic single scattering during volume rendering to further improve the spatial perception of the phase interfaces.

To summarize, our contributions are:

- A scalable method with minimal preprocessing for direct volume rendering of stochastically sampled unstructured particle data,
- An importance measure for the stochastic sampling of particles,
- The reconstruction and shading of phase interfaces during volume rendering.

2. Related Work

The SPH method and the rendering of particle-based fluids is a popular method in computer graphics [IOS*14]. Here, we focus on the visualization of SPH data, but do not consider the rendering of photorealistic fluids.

Volume rendering. To employ standard direct volume rendering, the unstructured particle data can be resampled to a regular grid. For large volume sizes, out-of-core rendering techniques are needed [BHP15]. Several approaches specific to particle data have been proposed to speed up the expensive resampling and to reduce storage requirements. Orthmann *et al*. [OKK10] enhance data-parallel octrees with several caches for fast level-of-detail ray-casting. A level-of-detail representation is also used by Fraedrich *et al*. [FAW10]. Based on the current view frustum, the appropriate levels in the hierarchy are resampled to a perspective grid that is updated in each frame. In contrast, Reichl *et al*. [RTW13] resample extreme-scale cosmological data to a compressed octree in a time-consuming preprocess while our preprocessing takes milliseconds.

Surface reconstruction. Surface extraction from particle data has been extensively studied, but still forms a major bottleneck in the visualization pipeline due to large computational time and memory requirements. Most methods are based on the marching cubes algorithm [LC87], but use different scalar fields [ZB05, APKG07, SSP07, OCv13]. The resulting surfaces usually suffer from bumpiness due to the irregular distribution of particles [AAIT12, YT13].

Alternatively, surfaces can be reconstructed and rendered directly during volume rendering [PSL*98]. Since isosurfaces correspond to a transfer function with a Dirac impulse at the isovalue, the isovalues have to be accurately found during ray marching. Hadwiger *et al*. [HSS*05] use the secant method for finding the isosurface locations. This approach has been extended by Knoll *et al*. [KHW*09], who make use of peak finding as an alternative to pre-integration of the transfer function. Igouchkine *et al*. [IZM18] recently introduced multi-material volume rendering with accurate surface reflections. To this end, they use two transfer functions, one to determine the material of a sample point and the other one to assign rendering parameters to materials.

Direct rendering of unstructured particles. Ray casting of semi-transparent points and metaballs has been investigated extensively [GIK*07, KAH07, KSN08, SI12, WKJ*15]. Here, we focus on methods that visualize the volume represented by the particles and their kernels. Jang *et al*. [JFSP10] ray cast particle data on the GPU using binary space partitioning. To speed up the computation, smaller kernel radii are used while generating the hierarchical data structures. Hochstetter *et al*. [HOK16] perform volume rendering of particles in a sparse, view-aligned grid. The grid is then traversed in bundles of rays that are adaptively sampled, bounded by a user-controlled error in screen-space. In contrast, we select a subset of particles during the SPH evaluation, which allows us to sample both view-dependent and based on the spatial data complexity. Similar to us, Reda *et al*. [RKN*13] perform direct rendering of particle densities with an acceleration grid. Compared to them, we focus mainly on datasets with dense particle distributions and introduce stochastic sampling to reduce accompanying sampling costs.

Zirr and Dachsbacher perform on-the-fly voxelization [ZD18] to interactively render photorealistic particle-based fluids. Reichl *et al*. [RCSW14] similarly use a binary representation to render large fluid simulations, but require several seconds long preprocessing while ours only takes milliseconds. Due to their binary voxel representation, their approach is limited to homogeneous fluids containing a single phase.

As SPH rendering is a special case of radial basis function (RBF) rendering, related approaches are interchangeable. Jang *et al*. [JWH*04] use texture slicing to render a small set of RBFs residing in GPU memory. Knoll *et al*. [KWN*14] perform direct volume rendering of RBFs with ray bundles traversing a bounding volume hierarchy. The construction of their bounding volumes can take several minutes for large datasets. Our solution enables fast preprocessing at the expense of higher render times, making it more suitable for dynamic or time-dependent data.

3. Volume Rendering of Multi-Phase SPH Data

In this section, we first introduce volume rendering in Section 3.1 and its application to SPH data in Section 3.2. Then, we present our material model for visualizing multi-phase data in Section 3.3.

3.1. Direct volume rendering

Visualizing particle data with volume rendering allows for presenting properties below the volume's surface. Especially in the context of multi-phase SPH data, where particle phases may be occluded or mixed with other types, volumetric rendering offers a better understanding of the data. Direct volume rendering is the established approach for visualizing volumetric data and extensive research exists on this topic [Max95, HKRs*06, JSYR14]. Its foundation is solving the volume rendering integral for light transport along per-pixel view rays reaching from position $\mathbf{x}_0$ to $\mathbf{x}$. The integral has different formulations depending on the considered optical phenomenon.

We use an emission-absorption model with additional single scattering [HKRs*06, JSYR14] defined as

$$L(\mathbf{x}, \boldsymbol{\omega}) = \int_{\mathbf{x}_0}^{\mathbf{x}} \left(q(\mathbf{x}', \boldsymbol{\omega}) + \kappa_s(\mathbf{x}')\, f(\mathbf{x}', \boldsymbol{\omega}, \boldsymbol{\omega}_l)\, L_i(\mathbf{x}', \boldsymbol{\omega}_l)\right) T(\mathbf{x}', \mathbf{x})\, \mathrm{d}\mathbf{x}'. \quad (1)$$

It describes the incoming radiance L reaching point $\mathbf{x}$ from the direction $\boldsymbol{\omega}$. The integral accumulates the radiance from all points $\mathbf{x}'$ along the ray that is attenuated according to the volume's transmittance $T(\mathbf{x}_1, \mathbf{x}_2)$ between two points $\mathbf{x}_1$ and $\mathbf{x}_2$. Given the volumetric absorption coefficient κ_a and scattering coefficient κ_s, the transmittance is

$$T(\mathbf{x}_1, \mathbf{x}_2) = e^{-\int_{\mathbf{x}_1}^{\mathbf{x}_2} \kappa_a(\mathbf{x}) + \kappa_s(\mathbf{x})\, \mathrm{d}\mathbf{x}}. \quad (2)$$

All radiance contributions at all points along the ray stem from two parts. The first one is volumetric emission q. The second one is light from an external light source that is scattered to $\mathbf{x}$ and attenuated by the scattering coefficient κ_s. The light source is a single point or directional light in direction $\boldsymbol{\omega}_l$. Before radiance emitted from the light source at $\mathbf{x}_l$ reaches $\mathbf{x}'$ as incident radiance L_i, it may be attenuated by volumetric occlusion on its way. If this occlusion is evaluated, a secondary integral has to be solved for each point $\mathbf{x}'$ along the view ray for $T(\mathbf{x}', \mathbf{x}_l)$.

The function f controls the ratio of L_i that is scattered in direction $\boldsymbol{\omega}$. Instead of just taking volumetric scattering through a scattering distribution function f_s into account, we add another distribution function f_r for reflections on surface-like structures. Both in combination describe how much of the light that reaches a point $\mathbf{x}$ from direction $\boldsymbol{\omega}_l$ is scattered or reflected in direction $\boldsymbol{\omega}$:

$$f(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_l) = R(\mathbf{x})\, f_r(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_l) + f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_l). \quad (3)$$

$R(\mathbf{x})$ is the reflection indication function that is 1 if $\mathbf{x}$ is a point in the volume where a reflection should occur and 0 otherwise. We will discuss later how to determine these points. The utilization of surface reflections is phenomenologically motivated for a better perception of structures and material interfaces inside the volume. It may violate physical correctness, for example, energy conversion [AD16]. f_r can be modeled in form of a BRDF. We employ a Blinn-Phong reflection model [Bli77]. For scattering in volumetric participating media, f_s is a phase function. We use a Henyey-Greenstein phase function [HG41] which approximates Mie-scattering.

As solving the volume rendering integral analytically is intractable, a numerical integration via ray marching is applied in practice [HKRs*06]. Optical parameters along the rays are expected to be piece-wise constant along consecutive ray segments between sampling points $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. The contribution of each segment can be computed in closed form and accumulated iteratively in a process referred to as compositing.

3.2. SPH evaluation

In our case, the volumetric data to render is the SPH domain. SPH approximates volumetric fluid or gaseous material properties with a set of particles carrying these properties. We denote attributes of a particle i as A_i, for example its position $\mathbf{x}_i$. An attribute at an arbitrary position $\mathbf{x}$ is approximated as a sum over all particle attributes weighted by a superimposed kernel function $W(\|\mathbf{x}_i - \mathbf{x}\|, h)$ [Mon92]. The parameter h is the kernel smoothing length. All of the fluid's spatial attributes A, for example its velocity, are carried by the particles with position $\mathbf{x}_i$, mass m_i, and density ρ_i. The approximated attribute at an arbitrary position $\mathbf{x}$ is defined as follows:

$$A(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} A_i\, W(\|\mathbf{x}_i - \mathbf{x}\|,\ h). \quad (4)$$

For example, the density at a point $\mathbf{x}$ can be obtained by summing over the particle density attributes ρ_i resulting in

$$\rho(\mathbf{x}) = \sum_i m_i\, W(\|\mathbf{x}_i - \mathbf{x}\|,\ h). \quad (5)$$

Since volume attributes are defined in form of a kernel weighted sum, the gradient of these attributes can be decomposed into a sum of all separate particle kernel gradients as well [Mon92]. For a position $\mathbf{x}$, the gradient is denoted as

$$\nabla A(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} A_i\, \nabla W(\mathbf{x}_i - \mathbf{x},\ h). \quad (6)$$

The gradient of the kernel function $\nabla W(\mathbf{x}_i - \mathbf{x},\ h)$ can be derived analytically. For multi-phase SPH simulations, each particle is assigned to one of m fluid phases $\Pi = \{\tau_0, \tau_1, .., \tau_m\}$ using a per-particle attribute $\pi_i \in \Pi$. Phases may be understood as distinct fluid types that have different properties and simulation behaviours. For example, one SPH phase can represent solid boundary geometry while other phases correspond to fluids inside [MK95, CBD*18].

If the support of a kernel function is finite, all particles with a contribution to $\mathbf{x}$ are limited to a local neighbourhood. The resulting neighbourhood search can be sped up significantly by storing particles in an acceleration data structure. Different data structures such as octrees or hash-tables are applicable [IOS*14]. We use a uniform grid that we explain in more detail in Section 5.2.

3.3. Multi-phase material model

Once the SPH attributes at a sampling position during ray marching are determined, they can be mapped to optical parameters for the evaluation of the volume rendering integral in a next step. Our approach is focused on multi-phase SPH data and represents each SPH phase with one user-parameterizable shading material. Shading and thus shading material definitions consist of two parts: volumetric light interaction with the data and light reflectance at regions in the data that we identify as material interfaces.

3.3.1. *Material model*

For surface shading, we use the Blinn-Phong model which is sufficient for the perception of material interfaces and surface orientations. An additional surface parameter per material specifies their opacity. Surface shading deals with the term $R(\mathbf{x})\, f_r(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_l)$ in Equation 3. Volumetric material parameters define emission $q(\mathbf{x})$ and absorption κ_a as well as the anisotropy of the Henyey-Greenstein phase function $f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_l)$. For visualization purposes, we also support mapping an SPH attribute to optical parameters

A Each sampling point during ray marching is assigned to exactly one SPH phase and thus shading material.

B Shading is performed over one ray segment and can consist of two volume and one surface shading part.

Figure 1: *Multi-material model during ray marching.*

using a transfer function. In particular, these are the material's volume emission, absorption and scattering coefficient, and simultaneously, the surface's diffuse colour and opacity coefficients. The radiance that is reflected on surfaces or scattered in volumes depends on the incoming light from the light source which is potentially attenuated through volumetric occlusion.

3.3.2. *Multi-phase SPH evaluation*

We incorporate surface and volume shading in a consistent multi-material model into the volume rendering ray marching. Similar to Igouchkine *et al.* [IZM18], volume attributes at a sampling position $\mathbf{x}$ are not mapped directly to optical properties. Instead, each sampling point is first assigned to one distinct SPH phase and therefore one shading material (Figure 1(a)). Then the optical properties are inferred only from this phase's attributes at the sampling point and the shading material parameters. We define the evaluation at $\mathbf{x}$ of an attribute A over particles of phase π, omitting all particles of other phases, as $A(\pi, \mathbf{x})$. Since an SPH phase is a categorical per-particle attribute, approximating the phase at an arbitrary position as in Equation 4 is not meaningful. Instead, we choose the distinct SPH phase assigned to a point $\mathbf{x}$ to be the one with the highest density:

$$\pi(\mathbf{x}) = \arg\max_{\tau \in \Pi} \rho(\tau, \mathbf{x}). \tag{7}$$

Finally, we set the phase and corresponding shading material of a ray marching sampling point at $\mathbf{x}$ to $\pi(\mathbf{x})$. If other SPH attributes are relevant for rendering, e.g. to apply a transfer function, these are determined as $A(\pi(\mathbf{x}), \mathbf{x})$. We discard attributes and materials of other phases at $\mathbf{x}$ as our datasets contain mostly homogeneous SPH phase distributions with clear interfaces. However, this assumption is no limitation of our technique as it can easily be extended to consider contributions of other phases as well. In the following, we explain our multi-material shading which assumes that each sampling point is uniquely assigned to one phase and, while not directly part of this work, how it can be extended to miscible fluids as well.

3.3.3. *Ray segment shading*

Based on the iterative compositing process for solving the volume rendering integral, shading is always restricted to one ray segment between the current and previous sampling points $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ with $\mathbf{x}_{i-1}$ being the point closest to the camera. Two cases regarding the multi-material shading over this ray segment can occur: If $\pi(\mathbf{x}_i) = \pi(\mathbf{x}_{i-1})$, the segment consists of only one particle phase and homogeneous volume shading is applied. If $\pi(\mathbf{x}_i) \neq \pi(\mathbf{x}_{i-1})$, a phase interface exists in the segment resulting in a ray segment that is composed of three ordered shading parts (Figure 1(b)):

1. a homogeneous volumetric part of the last phase $\pi(\mathbf{x}_{i-1})$,
2. a reflective surface of $\pi(\mathbf{x}_i)$, and
3. a volumetric part of the phase $\pi(\mathbf{x}_i)$.

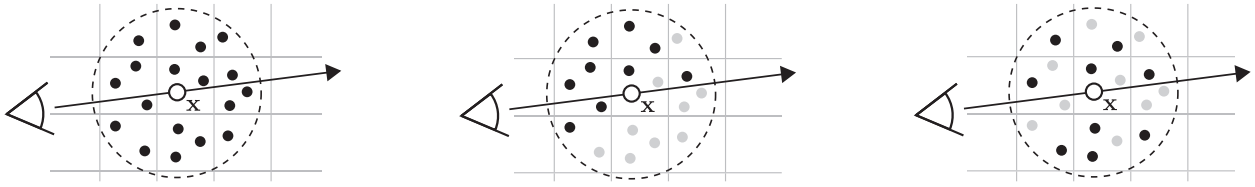
The radiance contributions of all three parts are computed separately through volume and surface shading and are combined using a ray segment internal compositing step. This complete segment radiance is used in the actual ray marching compositing. The surface position $\mathbf{x}_s$ is given by the first point where $\pi(\mathbf{x}_s) = \pi(\mathbf{x}_i)$. We approximate it using a binary search with a fixed maximum iteration count. For most cases, we found one iteration to be already sufficient for finding consistent surface positions. Both volumetric shading steps also depend on $\mathbf{x}_s$ as it defines the depth of the volumes and, thus, their radiance contributions to the ray segment. Apart from $\pi(\mathbf{x}_i)$ and the SPH attribute $A(\pi(\mathbf{x}_i), \mathbf{x}_s)$, surface shading also depends on the surface normal $\mathbf{n}(\pi(\mathbf{x}_i), \mathbf{x}_s)$ that is derived from the density gradient as

$$\mathbf{n}(\pi(\mathbf{x}_i), \mathbf{x}_s) = -\frac{\nabla\rho(\pi(\mathbf{x}_i), \mathbf{x}_s)}{\|\nabla\rho(\pi(\mathbf{x}_i), \mathbf{x}_s)\|}. \tag{8}$$

To ensure accurate surface shading, the SPH volume is additionally sampled at $\mathbf{x}_s$ for $A(\pi(\mathbf{x}_i), \mathbf{x}_s)$ and $\mathbf{n}(\pi(\mathbf{x}_i), \mathbf{x}_s)$. The proposed multi-material shading leads to a clean distinction between volume and surface shading and allows for integrating both methods in a common light transport model. It also does not rely on heuristics to determine surface occurrences, as these are only expected at actual phase transitions. While the phase maximum is a usual criterion for surface rendering, volume shading can be extended to consider miscible fluids as well [KPNS10]. To that end, a weighted sum of contributions from different phases would be computed.

4. Stochastic Sampling

Directly approximating the SPH attributes at every step during ray march ing is computationally extremely demanding. In fact, the factor that almost solely determines the rendering time are the memory accesses that are necessary to obtain the particle attributes during rendering, see Figure 7(a). For some SPH datasets, thousands of particles may be accessed at each sampling point during ray marching. In the following, we propose a novel technique to speed up direct SPH rendering by reducing the number of particle accesses, which we refer to as stochastic sampling, see Figure 2. The general idea is to consider only a subset of particles to evaluate the SPH attribute at a point $\mathbf{x}$. We thereby take each neighbouring particle only with a probability $p(\mathbf{x})$. This implies rendering with a stochastic, but not a systematic error. In particular, if $p(\mathbf{x}) \to 1$, the sampling converges to the correct SPH approximation. We derive the sampling probabilities $p(\mathbf{x})$ in Section 4.1. In Section 4.2, we



A Without subsampling, each SPH approximation accesses a large number of particles.

B Stochastically sampling particles for SPH approximations reduces particle accesses.

C Sampling subsets of particles in each grid cell leads to stratification.

Figure 2: *Accessing neighbouring particles at a sampling point during ray marching.*

further extend our sampling strategy to include stratification. Lastly, we discuss sampling during surface shading and single scattering in Section 4.3.

4.1. Particle sampling probability

Instead of reducing the particle accesses using a constant probability, $p(\mathbf{x})$ can be chosen different at each point $\mathbf{x}$. Here, different probabilities are acceptable regarding the resulting error in the final output image. This enables more aggressive subsampling in homogeneous regions while more effort is spent on heterogeneous ones. We scale the sampling ratio to a user-specified interval $[p_{\min}, p_{\max}]$ that allows to tailor the amount of sampling to different datasets and use cases. Different heuristics to determine the non-uniform probability $p(\mathbf{x}) \in [p_{\min}, p_{\max}]$ are conceivable. We propose the following criteria to affect the probability $p(\mathbf{x})$:

- a view-dependent ratio $p_t(\mathbf{x})$ that is computed in real time during rendering, and
- a data importance $p_i(\mathbf{x})$ that is precomputed for an SPH dataset.

If an attribute approximation $A(\pi(\mathbf{x}), \mathbf{x})$ is performed during ray marching, the two criteria $p_t, p_i \in [0, 1]$ are combined multiplicatively to obtain the particle sampling probability as

$$p(\mathbf{x}) = p_{\min} + (p_{\max} - p_{\min})\, p_t(\mathbf{x})\, p_i(\mathbf{x}). \tag{9}$$

4.1.1. *View-dependent ratio*

If the ray marching happens front-to-back starting from the camera, the occlusion of each sampling step during ray marching is known. Specifically, the occlusion can be obtained from the transmittance that is accumulated during ray marching compositing as $1 - T(\mathbf{x}_0, \mathbf{x})$. Sampling points that are less visible have less contribution to the final pixel colour. Therefore, a higher sampling error is tolerable regarding their SPH evaluation. We set $p_t(\mathbf{x})$ according to $T(\mathbf{x}_0, \mathbf{x})$ for all SPH approximations at a sampling point. To maintain a near-correct sampling at points that are almost completely visible and increase the subsampling at mostly occluded points we use a quadratic falloff as

$$p_t(\mathbf{x}) = 1 - (1 - T(\mathbf{x}_0, \mathbf{x}))^2. \tag{10}$$

4.1.2. *Precomputed data importance*

According to Equation 4, different particle sampling probabilities may be reasonable depending on the distribution of attributes A_i of neighbuoring particles i. If neighbouring particles have a higher attribute variation, the variance in the sampling process is increased and more particles should be considered in the SPH approximation. Such different attribute distributions result from the SPH dataset to render. As a measure of the SPH attribute variation of the particles in a local neighbourhood, the entropy [Sha48] of the attribute values can be used, which encodes the information complexity in the particle set. We compute the entropy of an SPH attribute A in a particle set $\mathcal{P}$ as follows: First, all attribute values of the particles in $\mathcal{P}$ are binned in a normalized histogram $h(b)$ with N_{bins} bins. Then, the entropy is obtained as

$$\mathbb{H}_{\mathcal{P}}(A) = -\sum_{b=0}^{N_{\text{bins}}-1} h(b)\, log_2\, h(b). \tag{11}$$

We normalize the values to be independent from the histogram size according to Wei *et al*. [WDS18] with

$$\overline{\mathbb{H}}_{\mathcal{P}}(A) = \frac{2^{\mathbb{H}_{\mathcal{P}}(A)}}{N_{\text{bins}}}. \tag{12}$$

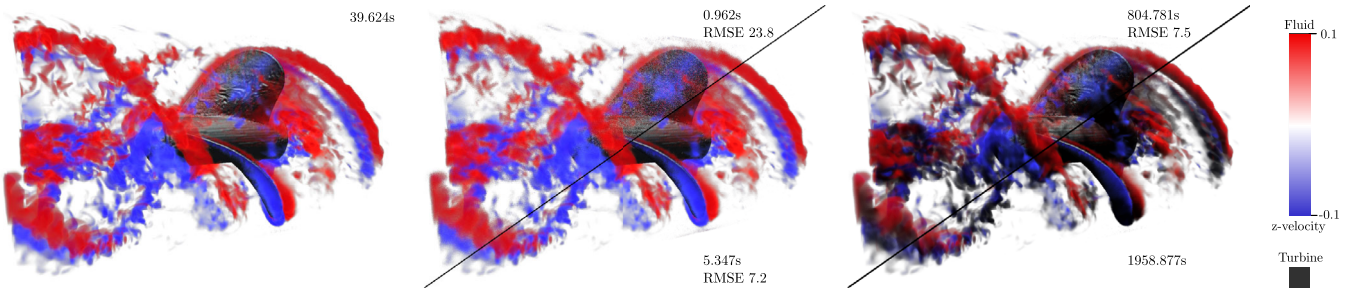
Apart from varying attributes, regions where phases differ are additionally prone to subsampling errors. This has two reasons: First, the approximation of phases according to Equation 7 is discontinuous which may lead to edge cases in sampling, where false distinct phases may be assigned to a sampling point. Second, regions with different phases may contain phase interfaces and thus provoke surface shading. Because of high-frequency specular highlights, high opacities in the ray marching compositing, and the dependency on normals, even small SPH subsampling errors lead to poor surface shading.

Based on the above observations, we define the importance value for a position $\mathbf{x}$, with $\mathcal{P}(\mathbf{x})$ being the set of particles in the local neighbourhood of $\mathbf{x}$, as

$$p_i(\mathbf{x}) = \begin{cases} 0, & \text{if} \quad |\mathcal{P}(\mathbf{x})| = 0 \\ 1, & \text{if} \quad \exists i, j \in \mathcal{P}(\mathbf{x}) : \pi_i \neq \pi_j\,, \\ \overline{\mathbb{H}}_{\mathcal{P}(\mathbf{x})}(A), & \text{otherwise} \end{cases} \tag{13}$$

where A is the SPH attribute currently visualized.

The data importance $p_i(\mathbf{x})$ is the only precomputation that we make for our method. We store the $p_i(\mathbf{x})$ in a uniform grid that



A Volume rendering of the Turbine dataset directly on the unstructured particle data.

B Our stochastic technique scales from fast (top) to high quality (bottom) renderings.

C Single scattering with (top) and without stochastic sampling (bottom).

Figure 3: *We visualize an SPH dataset of a fluid that rotates a turbine using volume rendering with surface shading (a). The datasets contains 86 million particles that are evaluated on-the-fly without significant preprocessing. We employ stochastic particle sampling during the SPH evaluation which substantially improves render times (b). This enables us to include expensive single scattering during volume rendering (c).*

covers the spatial extent of the SPH data. This means that $p_i(\mathbf{x})$ is constant for all $\mathbf{x}$ in the same grid cell. The precomputation is carried out in two steps. In the first step, an initial importance $p_i(\mathbf{x_c})$ for each cell of the importance grid is obtained according to Equation 13, whereby $\mathbf{x}_c$ denotes the centre point of the cell and $\mathcal{P}(\mathbf{x}_c)$ is assumed to be the set of particles lying in this cell.

In a second step, these per-cell importance values are filtered with a three-dimensional maximum dilation over neighbouring grid cells. This compensates for the fact that neighbouring grid cells of a sampling point are also relevant for SPH approximations even though the first precomputed $p_i(\mathbf{x}_c)$ only takes particles in the same cell into account. Especially considering the discontinuous approximation of the SPH phase at a sampling position (Equation 7), high values of $p_i(\mathbf{x}_c)$ in one cell should lead to a higher data importance for its neighbours. The dilation uses a constant kernel with a width equal to the number of grid cells that are covered by the finite support of the SPH kernel function used in rendering.

At the same time, the filtering step is used to increase the data importance for cells that may contain interfaces with regions that do not contain any particles at all, as these interfaces also lead to surface shading. As an approximation, a cell is expected to contain possible free-surfaces if more than half of the cells in the dilation kernel do not contain particles. For these cells, $p_i(\mathbf{x}_c)$ is set to one. The filtering process can be repeated with a user-specified number of iterations. In practice, we found one or two iterations to be sufficient.

At each SPH approximation during ray marching at a position $\mathbf{x}$, the precomputed data importance value $p_i(\mathbf{x})$ is queried from the respective importance grid cell and combined with the view-dependent transmittance factor $p_t(\mathbf{x})$ to determine the particle sampling probability $p(\mathbf{x})$ according to Equation 9. This leads to stronger subsampling in cases with lower risk of sampling errors and low pixel colour contribution and a more precise sampling in other cases. Note that we only precompute our sampling importance and not any shading parameters nor SPH attributes in the discrete grid. The rendering directly evaluates the original particle set.

4.2. Stratification

We use stratification of particle samples to reduce the overall sampling error by selecting well distributed samples in the neighbourhood, cf. Figure 2(b) and (c). Stratification is employed using the acceleration structure containing all particles, which in our case is a uniform grid as discussed before. Note that the acceleration structure and the data importance grid are two different constructs. Here, we assume that both uniform grids have equal cell sizes and overlap perfectly, although this is generally not necessary. Then, overlapping cells in both structures have the same centre point $\mathbf{x}_c$. The stratification is introduced as follows: Instead of randomly discarding each single particle with a probability $p(\mathbf{x})$ at a sampling position $\mathbf{x}$, a subset of

$$n = p_{\min} + (p_{\max} - p_{\min})\, p_i(\mathbf{x}_c)\, p_t(\mathbf{x})\, |\mathcal{P}(\mathbf{x}_c)|$$

particles is chosen from each cell in the acceleration structure with centre point $\mathbf{x}_c$ and containing particles $\mathcal{P}(\mathbf{x}_c)$ during the SPH approximation. More precisely, $\lfloor n \rfloor$ and an additional particle with a probability of $(n - \lfloor n \rfloor)$ are chosen. Note that as multiple cells have to be accessed at each sampling position, $p_t(\mathbf{x})$ remains a constant value while $p_i(\mathbf{x}_c)$ can differ between these accessed cells during a single SPH approximation. The stratification factor can be increased by using a finer grid resolution and thus smaller cells. However, this can also decrease performance as the optimal cell width for efficient particle access is always equal to the kernel function support [IABT11].

4.3. Surface shading and single scattering

As discussed above, phase interfaces are particularly prone to subsampling artifacts because of varying particle phases in the neighbourhood and the high-frequency nature of surface shading. Thus, if we detect a phase transition that leads to surface shading during ray marching, we use a constant sampling probability of one for the iterative search for the surface position $\mathbf{x}_s$ and the resampling of the SPH attributes at $\mathbf{x}_s$ for shading.

In contrast to surface shading, the occlusion of the light source by the volume when computing single scattering often has low

frequency and is relatively impervious to particle subsampling. This allows reducing the particle sampling probability during the secondary ray marching for querying the light source occlusion compared to the primary ray marching. We reduce the probability with a factor of 10% if an occlusion ray is being cast from a sampling point in a volumetric area. We do not reduce the probability if an occlusion ray stems from a ray segment containing surface shading. This case discrimination is used because surfaces generally have a higher opacity than volumetric ray segments and use the irradiance that is influenced from the occlusion information for surface shading. Since the occlusion ray marching has to be carried out for each primary ray sampling point, by far the most SPH approximations happen on secondary rays. The lower probability for these rays thus leads to a significant reduction in particle memory accesses for renderings with single scattering.

5. Implementation

In the following, we discuss details regarding the implementation of our proposed rendering technique. Firstly, we discuss the shading and secondly our GPU rendering and particle sampling.

5.1. Shading implementation

While we use a Blinn-Phong BRDF and a Henyey-Greenstein phase function for surface and volume shading, other reflection and scattering functions can easily be included in our multi-material model without requiring further adjustments. For ray segments consisting of a homogeneous SPH phase only volume shading is required (Section 3.3.3). For these segments, we use pre-integrated transfer functions [EKE01]. Since pre-integration with a 2D table assumes a fixed segment length, it cannot be applied to segments with phase transitions.

When approximating an attribute at a sampling point, we use either a cubic spline [Mon92] or the Epanechnikov kernel function. While the cubic spline kernel leads to smoother results, its support is twice as wide as that of the Epanechnikov kernel and it therefore requires more particle accesses at each SPH approximation. The employed kernel function is thus a user-specified parameter.

5.2. Ray marching traversal

For SPH data with finite kernel support, three-dimensional uniform grids are commonly used to accelerate the particle neighbourhood search, especially in the simulation domain [IOS*14]. We also use such a grid which allows us to reuse the particle storage from simulations without further processing. Each of its cells stores a reference to the contiguous block of memory containing all particles in this cell. In contrast to volume rendering using resampling, digital differential analyzer traversal or similar higher order traversal methods are not applicable since the grid is only used to query the particles and does not constitute a discrete signal itself.

In a ray marching step, after determining the new sampling position $\mathbf{x}$, all potentially relevant particles are queried from the grid using stochastic sampling (Section 4). Then the SPH evaluation over these particles determines phase densities, attributes, and gradients at $\mathbf{x}$ (Section 3.2). Finally, the shading of the ray segment with an optional single scattering ray cast is performed (Section 3.3.3).

5.3. GPU implementation

We have implemented our renderer in platform independent OpenCL to take advantage of parallelization. While we optimize our implementation for GPUs, the renderer is also executable on CPUs or CPU clusters. As the generation of pseudo-random numbers on GPUs is difficult, we use precomputed random values which are, for example, used during particle sampling.

Our renderer is parallelized by assigning one thread to each pixel to evaluate the volume rendering integral along its corresponding pixel ray. Neighbouring pixels are thereby grouped together into work groups. To increase the chance of coalesced memory accesses whilst still sampling different particles, we adapt our stratified stochastic sampling (Section 4.2) as follows: For all SPH evaluations, the selected subset of particles per cell is always a contiguous block of particles in memory. To this end, all threads in a work group share the same random offset, but their local work group index is added as an offset. This increases the number of contiguous or broadcasting memory operations. Still, if threads from the same work group take samples from different grid cells, more expensive accesses may occur. Adding the local index as an offset decreases sampling correlation between work items.

6. Evaluation and Discussion

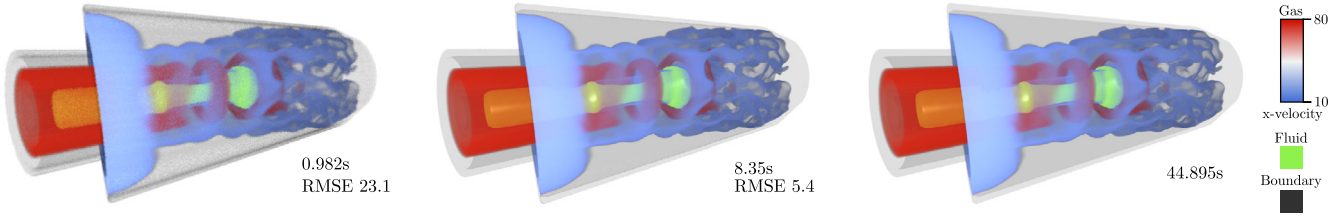
In the following section, we give an overview of several datasets to evaluate our rendering method. All renderings have a resolution of 1920×1080 pixels and are carried out on an AMD Radeon Pro SSG GPU with 16GB VRAM on a machine with an Intel Core i7-6700 CPU and 32GB of RAM.

6.1. SPH datasets

We apply our technique to three real-world SPH datasets. Even for the largest datasets, our flexible sampling technique requires negligible preprocessing if the simulation acceleration structure is reused for particle access. Table 1 lists timings and errors for different stochastic sampling configurations. We set particle mass and density $m_i = \rho_i = 1$ for all particles in our renderings.

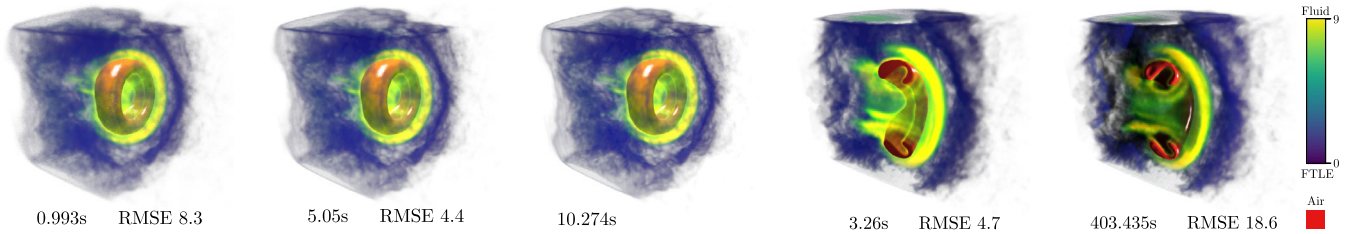
Turbine. This dataset consists of a hinged turbine that is rotated by a fluid. Both the turbine and the fluid are discretized by particles, leading to a total of 86 million particles per time step. The dataset has been simulated using the DualSPHyics [CDR*15] solver. In Figure 3(a), the velocity of the fluid phase is mapped to colour and the turbine is rendered as an opaque, gray surface. To avoid occlusion, we show only parts of the volume with a high absolute velocity component using the transfer function. This already conveys the general behaviour of the flow. Note that the absolute velocity around the turbine blades is high in positive and negative direction, which ultimately causes the turbine blade to rotate.

Direct volume rendering of the particle data is quite expensive, requiring nearly 40s. Using stochastic sampling (b), we can render the



A Fast rendering. **B** High-quality rendering with our sampling. **C** Rendering without our sampling.

Figure 4: *Volume rendering of the Spray Nozzle dataset with fast (a), with high-quality (b), and without stochastic sampling (c). The liquid phase (green) is injected on the left, together with a fast, co-flowing gas layer. This gas layer triggers instabilities on the liquid and on the surrounding gas phase.*



A Fast rendering. **B** High-quality rendering with sampling. **C** Rendering without sampling. **D** Cross section with sampling. **E** Cross section with sampling and single scattering.

Figure 5: *Volume rendering of the Bubble dataset with fast (a), with high-quality (b), and without stochastic sampling (c). To visualize the movement of the air bubble (red), we show the finite-time Lyapunov exponent of the surrounding water phase. In (d), we have removed half of the particles to visualize the flow inside the bubble. The same cross section is shown with sampling and single-scattering in (e).*

dataset with some noise in less than one second, or comparable to the reference in just over 5 s. Additionally, we can use single scattering (c). Although single scattering is still expensive with stochastic sampling, the rendering time is significantly reduced compared to the reference without sampling.

Spray nozzle. This dataset stems from an SPH simulation of a twin-fluid spray nozzle used for biofuel production [CBD*18]. The dataset is shown in Figure 4. It contains 43.2 million particles, which are divided into a liquid (green), a gas phase, and two distinct types of boundaries (grey). The liquid is injected axially on the left side at low velocity. At the same time, a co-flowing gas layer is injected at high velocity, triggering primary instabilities on the liquid surface. In the visualization shown in Figure 4, we map the velocity of the gas phase to colour and transparency. Since the co-flowing gas layer has a high velocity, it is mostly coloured in red. However, velocity changes in the middle of the cylinder lead to instabilities on the liquid and to the formation of vortices in the surrounding gas. By visualizing the phase interfaces together with the volumetric gas phase, we can effectively visualize these interactions.

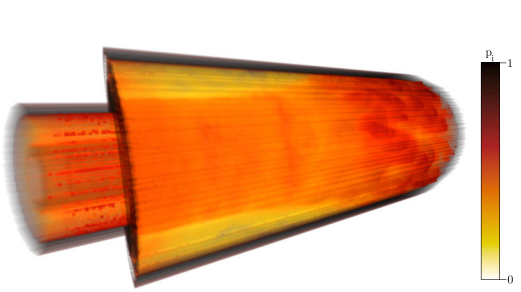
Even though the dataset contains less particles than the Turbine, rendering is more challenging due to the larger number of phases. Still, we can create an acceptable visualization in less than one second and a high-quality result in 8 s, see Figure 4(a) and (b). In comparison, without stochastic sampling the rendering takes nearly 45 s with comparable results to our high-quality rendering.

Bubble. This dataset shows a laminar, two-phase flow of an air bubble moving through water, see Figure 5. The dataset has been created with the GPUSPH [gpu] solver. The domain is discretized with 4.3 million particles in each of the 50 discrete time steps in the time interval [0s, 0.5s]. To visualize the separation of water over time as the bubble passes through, we show the forward finite-time Lyapunov exponent (FTLE) over the time interval [0.25s, 0.5s]. The air bubble, which is about to break up, is rendered as a red surface. The surface shading, and especially the specular highlights, effectively convey the shape and curvature of the sphere.

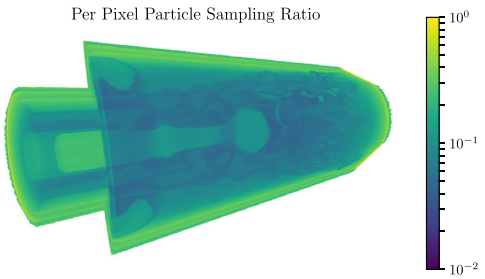
With aggressive subsampling, we are able to render this dataset in just less than one second, cf. Figure 5(a). Taking more samples reduces some noise on the bubble, as shown in (b). Compared to the reference in (c), which takes over 10 s, the quality does not visibly decrease. Since no significant preprocessing is required, we can interactively explore the dataset, compute different derived quantities such as the FTLE, and change the transfer function. In (d), we removed half of the particles, to visualize the flow inside the bubble. This shows the FTLE of the water phase, together with the water-air interface. With single-scattering (e), the perception of structure and features in the volume is further improved.

6.2. Stochastic sampling

Our proposed stochastic sampling technique increases performance by reducing the number of used particles during rendering.

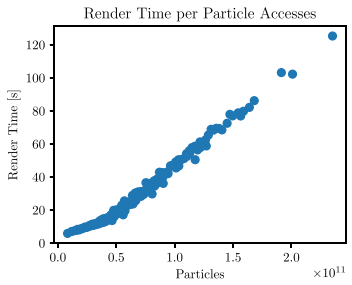


A Volume rendering of the precomputed cell importance p_i for the Spray Nozzle dataset. Higher values are assigned to complex areas containing vortices and material interfaces.

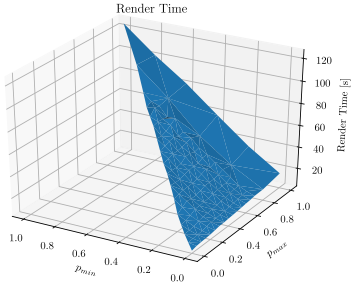


B Ratio of sampled particles to the total number of usually relevant particles when no sampling would be used. The ratio is visualized for the high quality sampling rendering in Figure 4 (b).

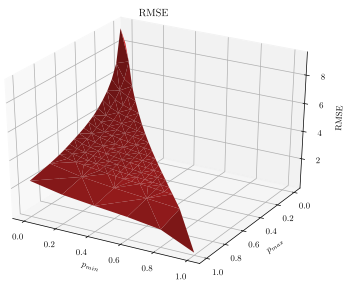
Figure 6: *Visualizations of different sampling probabilities in the Spray Nozzle dataset. Probabilities of our non-uniform particle sampling differ between pixel rays and dataset regions. In (a) we show different region probabilities. In (b), the per-pixel particle accesses are visualized.*



A Render time as a function of number of particle accesses. Render time relates linearly to particle accesses.



B Render time as a function of $p_{\min}$ and $p_{\max}$ for our sampling method. Render time is mostly affected by $p_{\min}$.



C Root mean square error as a function of $p_{\min}$ and $p_{\max}$ for our sampling method.

Figure 7: *Effects of particle subsampling on render time and root mean square error (RMSE) on the Spray Nozzle dataset. The correlation between the number of particle memory accesses and render time is shown in (a), while (b) and (c) illustrate the impact of the sampling configuration parameters p_{min} and p_{max} on render time and RMSE.*

Figure 7(a) shows that render time is lineary dependent on particle accesses and, thus, that decreasing the number of particle accesses is an appropriate method for increasing rendering performance. Instead of using uniform subsampling, we propose an non-uniform sampling strategy in Section 4 as a combination of the data-importance p_i and view-dependent criterion p_t. Figure 6(a) shows that the precomputed p_i is low for homogeneous areas in datasets and high for complex regions like the vortices in the Spray Nozzle dataset. This causes a higher sampling precision only in regions where it is necessary, specifically phase interfaces and inhomogeneous regions. In combination with p_t this leads to varying sampling probabilities during rendering as can be seen in Figure 6(b). Currently, particle sampling and preprocessing do not consider temporal properties. In cases where sampling noise is visible, it changes between frames as can be seen in the supplementary video. Handling such noise in time-dependent SPH data remains a challenge for future work.

The two parameters $p_{\min}$ and $p_{\max}$ for the minimal and maximal sampling probabilities are used to control our sampling. Figure 7(b) shows that render time mostly depends on $p_{\min}$. Figure 7(c) indicates that both parameters have a similar impact on the rendering error which is nearly linearly dependent if $p_{\min}$ and $p_{\max}$ are not assigned to extreme values near zero or one. Consequentially, the rendering error should mostly be controlled via $p_{\max}$ as it has the least effect on render times and $p_{\min}$ should generally be assigned to low values to increase performance. Intuitively, low values for $p_{\min}$ and high values for $p_{\max}$ result in a higher range of sampling probabilities that are utilized by our non-uniform sampling. Thus, employing the non-uniform sampling is beneficial as it offers shorter render times at equal error values. In contrast, for $p_{\min} \to p_{\max}$, our method collapses to uniform sampling omitting all performance advantages of our technique. Figure 8 shows an equal-time rendering of the Turbine dataset (Figure 3(b)) which has significantly higher error than our solution, further supporting the advantages of our technique. Generally, denser and more homogeneous datasets allow for more aggressive subsampling (i.e. lower values for p_{min} and p_{max}) as a larger number of particles contain the same attributes at each position. At the same time, we experienced that $p_{min} = 0$ can lead to a high sampling error at phase interfaces that were not correctly

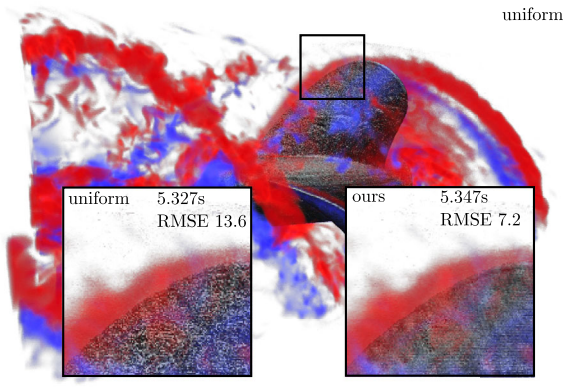


Figure 8: *Equal-time rendering using uniform particle sampling compared to our non-uniform sampling strategy in Figure 3(b). While both images took about 5.3 s to render, our strategy has significantly lower error.*

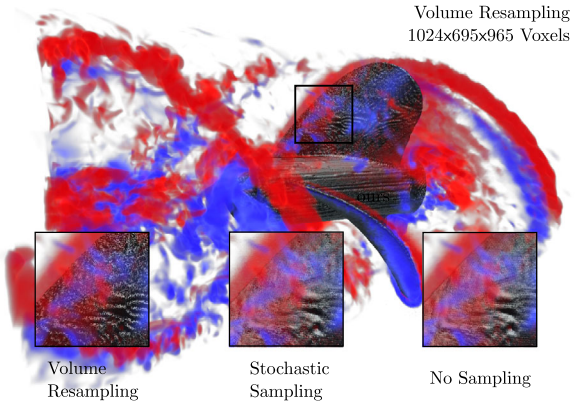


Figure 9: *Rendering using volume resampling (269.896 s preprocessing, 0.293 s rendering) with inlet comparisons to our high quality stochastic sampling configuration (0.029 s, 6.286 s) and a rendering without any sampling (0 s, 46.612 s).*

identified in the data importance p_i. We therefore set $p_{min} = 0.01$ for all high-quality stochastic sampling renderings of our datasets.

6.3. Comparison to volume resampling

In the following, we compare our direct SPH rendering to the common approach of volume resampling. This method evaluates SPH data on points of a three dimensional grid in a preprocessing step. Afterwards, direct volume rendering can be applied. While the latter is fast, the necessary large number of SPH evaluations results in long preprocessing times. For a meaningful comparison, we implement a GPU volume resampling solution that uses all parts of our multi-material rendering pipeline except for the stochastic sampling and direct access of the particle data. Instead of the direct SPH evaluation, optical parameters at a sampling point are inferred from a three dimensional texture that stores a preprocessed SPH phase as well as its attribute and density for the centre point of each voxel. Surface normals from density gradients are computed using central differences. We use hardware trilinear interpolation for attribute and density lookups and a custom interpolation of SPH phases because of the maximum operator (Section 3.3.2).

Render timings on the resampled volume lie between 82 ms for Spray Nozzle and 293 ms for Turbine with other render parameters being identical to our high quality stochastic sampling configurations in Figure 3 and Figure 4. Taking several minutes for each dataset, the volume resampling is orders of magnitude longer than our preprocessing, which takes less than 30 ms (Table 1). In combination with preintegrated transfer functions, image quality for homogenous volume materials is close to the renderings without resampling or particle subsampling at a grid resolution of 1024 voxels along the longest dimension. The precise reconstruction of phase interfaces and the accompanying surface rendering proves to be a difficult task, due to the discretization (Figure 9).

6.4. Discussion

Compared to creating an intermediate representation, such as a uniform grid, and subsequent volume rendering, direct rendering of the particles is still significantly slower. Related work that performs direct particle rendering in under one second per frame has little kernel overlap [RKN*13], is restricted to single-phase surface rendering [RCSW14], or suffers from long preprocessing [KWN*14]. The advantages of our method are clearly its negligible preprocessing times and scalability while handling dense multi-phase data. It is therefore well-suited for dynamic data, e.g. time-dependent data. As our sampling process is completely dynamic, it is possible to interactively choose screen regions where higher quality sampling is applied or to exclude particles and data regions from the rendering on the fly. If the data is mostly static, for example data with limited temporal resolution, creating an intermediate representation should be preferred.

Table 1: *Performance of our method including a comparison to volume resampling with 1024 voxels over the longest volume side.*

			Fast Sampling		Quality Sampling		No Sampling	Volume Resampling	
Dataset	Particles	Preproc.	Frame Time	RMSE	Frame Time	RMSE	Frame Time	Preproc.	Frame Time
Bubble	4,347,225	10 ms	0.993 s	8.277	5.050 s	4.360	10.274 s	143.234 s	124 ms
Spray Nozzle	43,188,662	28 ms	0.982 s	23.108	8.350 s	5.391	44.895 s	447.261 s	82 ms
Turbine	86,473,832	29 ms	0.962 s	23.828	5.347 s	7.187	39.624 s	269.896 s	293 ms

Root-mean-square errors are measured compared to the rendering without stochastic sampling.

While it would be possible to automatically choose fitting values for p_{min} and p_{max} based on the observations from the evaluation, some user refinement and control over the particle sampling remains useful. Our scalable sampling technique always allows for balancing the rendering configuration between performance and quality. The stochastic error that we introduce is data and view-dependent, and can be completely removed by only adjusting our parameters. In comparison, the systematic error stemming from discretization, quantization, or lossy compression cannot be removed without recomputing the intermediate representation. This is especially important when reconstructing the phase interfaces, which our approach is able to perform very accurately. Additionally, we can use gradient information from the SPH kernels as high-quality surface normals without explicitly storing them. In comparison, resampled volumes pose difficulties for high quality surface reconstruction in multi-phase data. Precomputing additional information such as gradients and per-voxel values for all instead of only one SPH phase is conceivable. But this would further increase the already high memory requirements. With respect to the multi-phase rendering, feedback from scientists working with SPH data was very positive. The relation of the phase interfaces and the surrounding flow is of major interest to them. By incorporating both volume and surface shading in the volume rendering, this relationship can be effectively conveyed.

7. Conclusion

In this work, we propose the direct volume rendering of large, dynamic particle data when expensive preprocessing is not desirable, e.g. for the interactive visualization of time-dependent data or for *in situ* applications. To accommodate different time and hardware budgets, we use stochastic sampling to reduce the amount of particles that are taken into consideration at each step. Our approach is specifically targeted towards free-surface and multi-phase flows. By including surface reconstruction and shading in the volume rendering, we are able to visualize the phase interfaces and the relation to the surrounding volume.

Acknowledgements

Open access funding enabled and organized by Projekt DEAL.

References

[AAIT12] Akinci G., Akinci N., Ihmsen M., Teschner M.: An efficient surface reconstruction pipeline for particle-based fluids. In *Workshop on Virtual Reality Interaction and Physical Simulation* (Geneva, Switzerland, 2012) The Eurographics Association, pp. 61–68. https://doi.org/10.2312/PE/vriphys/vriphys12/061-068.

[AD16] Ament M., Dachsbacher C.: Anisotropic ambient volume shading. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 1015–1024.

[APKG07] Adams B., Pauly M., Keiser R., Guibas L. J.: Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (2007).

[BHP15] Beyer J., Hadwiger M., Pfister H.: State-of-the-art in GPU-based large-scale volume visualization. *Computer Graphics Forum 34*, 8 (2015), 13–37.

[Bli77] Blinn J. F.: Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques* (1977), Association for Computing Machinery, New York, pp. 192–198..

[CBD*18] Chaussonnet G., Braun S., Dauch T., Keller M., Kaden J., Schwitzke C., Jakobs T., Koch R., Bauer H.-J.: Three-dimensional SPH simulation of a twin-fluid atomizer operating at high pressure. In *Proceedings of the 14th International Conference on Liquid Atomization and Spray Systems (ICLASS2018)* (2018). https://doi.org/10.2312/egst.20141034.

[CDR*15] Crespo A., Domínguez J., Rogers B., Gómez-Gesteira M., Longshaw S., Canelas R., Vacondio R., Barreiro A., García-Feal O.: DualSPHysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Computer Physics Communications 187* (2015), 204–216.

[CSI09] Cha D., Son S., Ihm I.: GPU-assisted high quality particle rendering. *Computer Graphics Forum 28*, 4 (2009), 1247–1255.

[EKE01] Engel K., Kraus M., Ertl T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (2001), pp. 9–16. https://doi.org/10.1145/383507.383515.

[FAW10] Fraedrich R., Auer S., Westermann R.: Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1533–1540.

[GIK*07] Gribble C. P., Ize T., Kensler A., Wald I., Parker S. G.: A coherent grid traversal approach to visualizing particle-based simulation data. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 758–768.

[gpu] GPUSPH. http://www.gpusph.org.

[HG41] Henyey L. G., Greenstein J. L.: Diffuse radiation in the galaxy. *Astrophysical Journal 93* (1941), 70–83.

[HKRs*06] Hadwiger M., Kniss J. M., Rezk-salama C., Weiskopf D., Engel K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Wellesley, MA, 2006.

[HOK16] Hochstetter H., Orthmann J., Kolb A.: Adaptive sampling for on-the-fly ray casting of particle-based fluids. In *Eurographics/ACM SIGGRAPH Symposium on High Performance Graphics* (2016), Eurographics Association, Geneva, Switzerland.

[HSS*05] Hadwiger M., Sigg C., Scharsach H., Buhler K., Gross M.: Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum 24*, 3 (2005), 303–312.

[IABT11] Ihmsen M., Akinci N., Becker M., Teschner M.: A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum 30*, 1 (2011), 99–112.

[IOS*14] Ihmsen M., Orthmann J., Solenthaler B., Kolb A., Teschner M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports, Strasbourg, France*, (2014). https://doi.org/10.2312/egst.20141034.

[IZM18] Igouchkine O., Zhang Y., Ma K.-L.: Multi-material volume rendering with a physically-based surface reflection model. *IEEE Transactions on Visualization and Computer Graphics 24*, 12 (2018), 3147–3159.

[JFSP10] Jang Y., Fuchs R., Schindler B., Peikert R.: Volumetric evaluation of meshless data from smoothed particle hydrodynamics simulations. In *IEEE/EG Symposium on Volume Graphics, Norrköping, Sweden* (2010). https://doi.org/10.2312/VG/VG10/045-052.

[JSYR14] Jönsson D., Sundén E., Ynnerman A., Ropinski T.: A survey of volumetric illumination techniques for interactive volume rendering. *Computer Graphics Forum 33*, 1 (2014).

[JWH*04] Jang Y., Weiler M., Hopf M., Huang J., Ebert D. S., Gaither K. P., Ertl T.: Interactively visualizing procedurally encoded scalar fields. In *Eurographics/IEEE VGTC Symposium on Visualization* (2004), Deussen O., Hansen C., Keim D., Saupe D. (Eds.), The Eurographics Association, Geneva, Switzerland. https://doi.org/10.2312/VisSym/VisSym04/035-044.

[KAH07] Kaehler R., Abel T., Hege H.-C.: Simultaneous GPU-assisted raycasting of unstructured point sets and volumetric grid data. In *Eurographics/IEEE VGTC Symposium on Volume Graphics* (2007). https://doi.org/10.2312/VG/VG07/049-056.

[KHW*09] Knoll A., Hijazi Y., Westerteiger R., Schott M., Hansen C., Hagen H.: Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1571–1578.

[KPNS10] Kang N., Park J., Noh J., Shin S. Y.: A hybrid approach to multiple fluid simulation using volume fractions. *Computer Graphics Forum 29*, 2 (2010), 685–694.

[KSN08] Kanamori Y., Szego Z., Nishita T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum 27*, 2 (2008), 351–360.

[KWN*14] Knoll A., Wald I., Navratil P., Bowen A., Reda K., Papka M. E., Gaither K.: Rbf volume ray casting on multicore and manycore cpus. *Computer Graphics Forum 33*, 3 (2014), 71–80.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: a high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[Max95] Max N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108.

[MK95] Monaghan J., Kocharyan A.: SPH simulation of multi-phase flow. *Computer Physics Communications 87*, 1 (1995), 225–235.

[Mon92] Monaghan J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574.

[Mon94] Monaghan J.: Simulating free surface flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406.

[NJB07] Navratil P., Johnson J., Bromm V.: Visualization of cosmological particle-based datasets. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1712–1718.

[OCv13] Onderik J., Chládek M., Ďurikovič R.: SPH with small scale details and improved surface reconstruction. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2013), SCCG '11, pp. 29–36. https://doi.org/10.1145/2461217.2461224.

[OKK10] Orthmann J., Keller M., Kolb A.: Topology-Caching for Dynamic Particle Volume Raycasting. In *Vision, Modeling, and Visualization* (2010), The Eurographics Association, Geneva, Switzerland. https://doi.org/10.2312/PE/VMV/VMV10/147-154.

[PSL*98] Parker S., Shirley P., Livnat Y., Hansen C., Sloan P. .: Interactive ray tracing for isosurface rendering. In *Proceedings of the Conference on Visualization '98* (1998), IEEE Computer Society Press, Los Alamitos, CA, pp. 233–238. https://doi.org/10.1109/VISUAL.1998.745713.

[RCSW14] Reichl F., Chajdas M. G., Schneider J., Westermann R.: Interactive rendering of giga-particle fluid simulations. In *Proceedings of High Performance Graphics* (2014), Eurographics Association, Geneva, Switzerland, pp. 105–116. https://doi.org/10.2312/hpg.20141099.

[RKN*13] Reda K., Knoll A., Nomura K., Papka M. E., Johnson A. E., Leigh J.: Visualizing large-scale atomistic simulations in ultra-resolution immersive environments. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)* (2013), pp. 59–65. https://doi.org/10.1109/LDAV.2013.6675159.

[RTW13] Reichl F., Treib M., Westermann R.: Visualization of big SPH simulations via compressed octree grids. In *2013 IEEE International Conference on Big Data* (2013), pp. 71–78. https://doi.org/10.1109/BigData.2013.6691717.

[Sha48] Shannon C. E.: A mathematical theory of communication. *The Bell System Technical Journal 27*, 3 (1948), 379–423.

[SI12] Szécsi L., Illés D.: Real-time metaball ray casting with fragment lists. In *Eurographics - Short Papers* (2012), Andujar C., Puppo E. (Eds.). https://doi.org/10.2312/conf/EG2012/short/093-096

[SSP07] Solenthaler B., Schläfli J., Pajarola R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82.

[WDS18] Wei T., Dutta S., Shen H.: Information guided data sampling and recovery using bitmap indexing. In *IEEE Pacific Visualization Symposium (PacificVis)* (2018), pp. 56–65. https://doi.org/10.1109/PacificVis.2018.00016.

[WKJ*15] Wald I., Knoll A., Johnson G. P., Usher W., Pascucci V., Papka M. E.: CPU ray tracing large particle data with balanced p-k-d trees. In *2015 IEEE Scientific Visualization Conference* (2015), pp. 57–64. https://doi.org/10.1109/SciVis.2015.7429492.

[YT13] Yu J., Turk G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics 32*, 1 (2013), 5:1–5:12.

[ZB05] Zhu Y., Bridson R.: Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (2005), 965–972.

[ZD18] Zirr T., Dachsbacher C.: Memory-efficient on-the-fly voxelization and rendering of particle data. *IEEE Transactions on Visualization and Computer Graphics 24*, 2 (2018), 1155–1166.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data Video S1

Data S1