Error-minimizing SPH Particle Merging for Constructing Multi-resolution Hierarchies

Jonathan Fischer
Department of Computer Science
Technische Universität Chemnitz
09111 Chemnitz, Germany
jonathan.fischer@informatik.tu-chemnitz.de

Lars Linsen
Computer Science a. Electr. Eng.
Jacobs University Bremen
28759 Bremen, Germany
l.linsen@jacobs-university.de

Paul Rosenthal
Department of Computer Science
Technische Universität Chemnitz
09111 Chemnitz, Germany
paul.rosenthal@informatik.tu-chemnitz.de

***Abstract*—We derive formulas suitable for computing the L^2 norm of the error inflicted on an attribute field by merging a given SPH particle subset into one given aggregate particle, applicable to any polynomial spline kernel function. Based on this analysis, we present an algorithm to efficiently compute this error measure. Furthermore, we provide a method for computing optimal attributes for an aggregate particle for a given set of children to be merged. Our method can be applied when constructing paritlce-based multi-resolution hierarchies, which provide representations of SPH data at different levels of detail. They allow for reduced computational complexity, for example, when applying visualization algorithms.**

I. Introduction

Scientists performing particle simulations request visualization techniques that are accurate and produce meaningful images but which at the same time can cope with today's larger data sets. In order to provide good and quick insights into simulation data, visualization must be scalable and manage the trade-off between fast and accurate rendering. Although many techniques for visualizing SPH data have been developed, they are either not directly applicable to large data sets or include a multi-resolution structure that cannot be applied to other visualization methods.

In contrast, particle-based hierarchies can serve as multi-purpose multi-resolution structures. A particle-based multi-resolution hierarchy is a tree of SPH particles. Each non-leaf particle is an aggregate of its children, serving as a more coarse replacement for them, while the leaves are the particles of the original data set, constituting the finest possible representation of the data. From such a hierarchy, an approximating representation of the data can be selected, i. e., a set of particles such that every original particle is either present or represented by exactly one of its ancestors.

While preserving the strongly varying particle resolution found in many SPH data sets well, particle-based hierarchies are especially unique in that any data representation chosen from it is itself a plain SPH data set and can therefore be visualized using any conventional single-resolution visualization method. While some of these methods require a pre-computation phase before being able to provide an interactive user experience, many may still be adaptable to work on a particle-based hierarchy. This may facilitate a straightforward enhancement of these methods to cope with large data sets. Moreover, one precomputed multi-resolution hierarchy could serve for, say, volume rendering and isosurface extraction of large SPH data sets, which could even be performed simultaneously to allow fast switching between or a combination of the two methods.

In this article, we present a solution to a fundamental problem which occurs when constructing particle-based multi-resolution hierarchies. We define a measure for the change inflicted on attribute fields when a subset of particles is merged into a single *aggregate particle* and present an algorithm to compute it efficiently. Based on this error measure, we show how to find error-minimizing attributes for aggregate particles.

We provide an exact definition of the problem in Section III, introducing the necessary notation. Section IV is dedicated to the fundamental concept of our approach, introducing the *kernel product integral*, its analytic solution, and a method for computing it efficiently, enabling a fast computation of the error. In Section V we lay out a method for finding optimal attributes for aggregate particles while in Section VI we present some results and comment on the performance of our algorithms. We conclude the article with a short summary in Section VII and provide some ideas for future work based on the presented methods.

II. Related Work

There is a large variety of different approaches for visualizing particle data. Many application scientists basically rely on displaying the particles as color-coded points or spheres [1], [2]. Similarly prevalent is the use of cutting planes or intensity integration. A well-known tool in the SPH community for such standard tasks is SPLASH [3]. All of these (often self-implemented) tools allow to gain a raw impression of the simulated data and to draw first conclusions. However, they are mostly limited to visualizing only one of the simulated variables and complex relations are hard to discover.

One direction for effective visualization of scalar variables is isosurface extraction. Many approaches rely on sampling the particle data to an auxiliary regular grid and using standard isosurface extraction techniques [4], [5]. However, using an intermediate grid always introduces the loss of adaptivity of the original particle data. This is circumvented by Rosenthal et al. [6], [7] by directly extracting isosurfaces from any point-based volume data. In addition, level sets can be used to improve the quality of the extracted surfaces in presence of highly varying particle density [8], [9]. Although implemented as narrow-band method, this approach is computationally intense and only manages to visualize relatively small data sets.

Large data sets have been visualized using view-dependent visualization approaches. Cha et al. [10] use photon tracing on the GPU for visualizing clouds simulated with SPH. A consequent advancement is the well-known approach of direct volume rendering, which has proven to generate high-quality renderings of regular data at interactive rates [11]–[13]. Such volume rendering approaches have also been applied to particle data. Kaehler et al. [14] visualize dark matter simulations by partitioning the domain in tetrahedral cells and ray-casting the data on the GPU. A direct volume rendering approach for SPH data is proposed by Jang et al. [15]. The authors utilize hierarchical space partitioning to efficiently estimate the volume of influence for each particle.

Level-of-detail algorithms have a long tradition in computer graphics [16], [17]. The main idea of utilizing different levels of abstraction for differently important parts of the data domain is also frequently used in large-scale visualization approaches, like for vector fields [18], isosurfaces [19], or for volume rendering [20]. Fraedrich et al. [21] propose a visually continuous LOD representation for SPH data. Particles are stored at discrete levels of detail in an octree. The coordinates of each particle are quantized to the respective cell's center and the domains of influence of two cells are merged if the radii do not differ too much. Afterwards, rendering is done by simply applying a transfer function to projected and accumulated particles. The same multiresolution structure is extended by Fraedrich et al. [22] to a volume rendering pipeline for SPH data using a regular perspective grid. However, these methods can hardly be applied to geometry-based visualization approaches like isosurface extraction or level sets. In contrast, particle-based hierarchies proposed by us allow the application of any SPH visualization technique.

SPH particle merging and splitting to regulate spatial resolution are well-established concepts in the field of SPH simulations. For example, Feldman and Bonet [23] propose a method for splitting particles in critical regions to enhance the accuracy of SPH simulations. Xiong et al. [24] recently presented an approach for interactively splitting and merging particles. Designed to facilitate good simulation results and to be extremely fast to compute, the merging schemes used in this context rely on simple averaging of particle attributes. Using such an approach for the visualization of a data field can introduce severe errors, which is also true for the adaptive SPH methods used in fluid simulations for computer graphics [25].

To the best of our knowledge, there exists no hierarchical method for SPH data which combines the flexibility of using particles as LOD primitives with the accuracy of an attribute-aware error metric. By facilitating the design of such methods, our algorithms fill a considerable portion of this gap.

III. Problem Formulation

An SPH data set consists of particles $p_i, i \in I$, where each particle is a tuple

$$p = (\mathbf{p}, \mu, \rho, \zeta, \alpha_1, \ldots, \alpha_u, \mathbf{b}_1, \ldots, \mathbf{b}_v) \in \mathbb{R}^3 \times \mathbb{R}^3_+ \times \mathbb{R}^u \times \mathbb{R}^{3v}.$$

Here, the different attributes denote the particle's position $\mathbf{p}$, mass μ, density ρ, radius of influence ζ, additional scalar values $\alpha_j, j = 1, \ldots, u$, and additional vector values $\mathbf{b}_j = (\beta_{j1}, \beta_{j2}, \beta_{j3}), j = 1, \ldots, v$.

A. SPH interpolation

The set of particles induces for each additional attribute a continuous field over $\mathbb{R}^3$ via the following construction. For every particle p we define its weight function $W : \mathbb{R}^3 \to [0, \infty)$ by

$$W(\mathbf{x}) = \frac{\mu \omega(\mathbf{x})}{\rho \zeta^3}, \quad \omega(\mathbf{x}) = w\left(\frac{\|\mathbf{x} - \mathbf{p}\|}{\zeta}\right),$$

where $\frac{1}{\zeta^3} w\left(\frac{\|\mathbf{x}-\mathbf{p}\|}{\zeta}\right)$ is an SPH kernel function defined by some continuous polynomial spline $w : [0, \infty) \to [0, \infty)$ of degree D with compact support and rational subdomain endpoints $q_0, \ldots, q_n$, $0 = q_0 < \cdots < q_n$, i. e.,

$$w(q) = \begin{cases} w_k(q) & q \in [q_{k-1}, q_k) \text{ for some } k \in \{1, \ldots, n\} \\ 0 & q \geq q_n, \end{cases}$$

where $w_k(q) = \sum_{d=0}^{D} w_{kd} q^d$.

As an example, we will use the broadly applied cubic kernel function [26] with two nontrivial pieces, defined by

$$w(q) = \frac{1}{4\pi} \begin{cases} (2-q)^3 - 4(1-q)^3, & 0 \leq q < 1 \\ (2-q)^3, & 1 \leq q < 2 \\ 0, & 2 \leq q\,. \end{cases} \tag{1}$$

However, since our method is applicable to any kernel function meeting the above requirements, we will continue our explanation using general terms.

Having defined a particle's weight, we can explicitly formulate the SPH interpolation defining any scalar or vector field α as

$$\alpha(\mathbf{x}) = \sum_{i \in I} \alpha_i W_i(\mathbf{x}) = \sum_{i \in I} \frac{\alpha_i \mu_i \omega_i(\mathbf{x})}{\rho_i \zeta_i^3}, \omega_i(\mathbf{x}) = w\left(\frac{\|\mathbf{x} - \mathbf{p}_i\|}{\zeta_i}\right).$$

B. Particle Merging

Given an SPH data set, we aim to merge some subset $\{p_i, i \in M\}, M \subseteq I, |M| \geq 2$ of it, i. e., replace these particles by just one single aggregate particle $\bar{p} = (\bar{\mathbf{p}}, \bar{\mu}, \bar{\rho}, \bar{\zeta}, \bar{\alpha}_1, \ldots, \bar{\alpha}_u, \bar{\mathbf{b}}_1, \ldots, \bar{\mathbf{b}}_v)$. This generally changes all of the scalar and vector fields. For every field α we define the error induced by the merge as the L^2 norm of the difference between the field and its "changed version", i. e.,

$$E_\alpha = \sqrt{\int_{\mathbb{R}^3} \left\| \bar{\alpha} \bar{W}(\mathbf{x}) - \sum_{i \in M} \alpha_i W_i(\mathbf{x}) \right\|^2 \mathrm{d}^3\mathbf{x}}.$$

Note that, while a field α is defined by the corresponding attribute data of all particles, the error E_α induced to this field by a merge depends only on the merged particles and the new aggregate particle $\bar{p}$. In a simplified manner, Fig. 1 depicts the meaning of the merge error.

We have now introduced all the necessary terms for stating precisely the problems we propose to solve:

- Given any particle subset $\{p_i, i \in M\}$ and an arbitrarily defined aggregate particle $\bar{p}$ for this subset, compute the errors induced by the merge to the attribute fields.
- Given any particle subset $\{p_i, i \in M\}$, compute a suitable aggregate particle $\bar{p}$ for merging the particles, minimizing the errors in the attribute fields.

IV. The Kernel Product Integral

For any scalar field α we can express the square of its merge error as

$$\begin{aligned} E_\alpha^2 &= \int\limits_{\mathbb{R}^3} \left(\bar{\alpha}\bar{W}(\mathbf{x}) - \sum_{i \in M} \alpha_i W_i(\mathbf{x}) \right)^2 \mathrm{d}^3\mathbf{x} \\ &= \bar{\alpha}^2 \int\limits_{\mathbb{R}^3} \left(\bar{W}(\mathbf{x})\right)^2 \mathrm{d}^3\mathbf{x} - 2 \sum_{i \in M} \bar{\alpha}\alpha_i \int\limits_{\mathbb{R}^3} \bar{W}(\mathbf{x}) W_i(\mathbf{x}) \,\mathrm{d}^3\mathbf{x} \\ &\quad + \sum_{i \in M} \sum_{j \in M} \alpha_i \alpha_j \int\limits_{\mathbb{R}^3} W_i(\mathbf{x}) W_j(\mathbf{x}) \,\mathrm{d}^3\mathbf{x}. \end{aligned} \tag{2}$$

Note that, while the integral is defined over volume regions with more than two particles contributing, we can express it in terms of integrals affected by only two particles each, i. e., we can easily compute it if we know how to compute the integral

$$\int\limits_{\mathbb{R}^3} \omega_i(\mathbf{x})\, \omega_j(\mathbf{x}) \,\mathrm{d}^3\mathbf{x} \tag{3}$$

for any two particles $\mathbf{p}_i$ and $\mathbf{p}_j$, which we call the *kernel product integral* in the following.

A. Solution

For computing (3), we can assume w.l.o.g. that $i = 1$ and $j = 2$ as well as $\mathbf{p}_1 = (0,0,0)$ and $\mathbf{p}_2 = (0,0,\Delta)$ where Δ is the distance between the two particles' positions. This can be achieved from any particle positions without changing the value of the integral by applying a translation and a rotation. Therefore, the kernel product integral is only a function of the positional distance Δ and the particles' radii ζ_1 and ζ_2. Thus we can write

$$P = P(\Delta, \zeta_1, \zeta_2) = \int\limits_{\mathbb{R}^3} \omega_1(\mathbf{x})\, \omega_2(\mathbf{x}) \,\mathrm{d}^3\mathbf{x}. \tag{4}$$

To arrive at a closed form solution of the integral (4), we transform $\mathbf{x} = (x,y,z)$ into spherical coordinates $(r,\tau,\theta) = \left(\sqrt{x^2+y^2+z^2}, \arccos\left(\frac{z}{r}\right), \arctan\left(\frac{y}{x}\right)\right)$ such that

$$\begin{aligned} P &= \int\limits_{r=0}^{\infty} \int\limits_{\tau=0}^{\pi} \int\limits_{\theta=0}^{2\pi} \omega_1(r,\tau,\theta)\, \omega_2(r,\tau,\theta) \cdot r^2 \sin\tau \,\mathrm{d}\theta \,\mathrm{d}\tau \,\mathrm{d}r \\ &= \int\limits_{r=0}^{\infty} \int\limits_{t=-1}^{1} \int\limits_{\theta=0}^{2\pi} \omega_1(r,t,\theta)\, \omega_2(r,t,\theta) \cdot r^2 \,\mathrm{d}\theta \,\mathrm{d}t \,\mathrm{d}r \end{aligned}$$

after substituting $t = \cos\tau$.

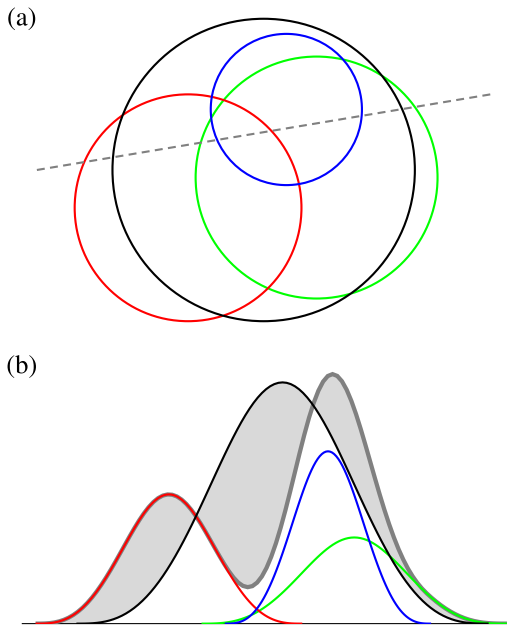

Fig. 1. Illustration of particle merging.
(a) Two-dimensional visualization of three particles selected for merging (red, green, and blue circle) into an aggregate particle (black circle). While a scalar SPH attribute field is defined at any point in three-dimensional space, we only illustrate its graph on the straight dashed gray line:
(b) The graphs of the contributions of the three particles (red, green, and blue). Replacing the particles by the aggregate particles changes the sum of the particles' contributions (gray) to the aggregate particle's contribution (black). Just like the L^2 norm in $\mathbb{R}^3$, the area between these two graphs (shaded region) is a measure for the merge error in one dimension.

We know that $\omega_k(r,t,\theta) = w\left(\frac{\delta_k(r,t,\theta)}{\zeta_k}\right)$ for any particle p_k, where $\delta_k(r,t,\theta)$ denotes the Euclidean distance between point (r,t,θ) and the particle's position $\mathbf{p}_k$. Clearly, $\delta_1(r,t,\theta) = \delta_1(r) = r$ and $\delta_2(r,t,\theta) = \delta_2(r,t) = \sqrt{r^2+\Delta^2-2\Delta r t}$, hence

$$\begin{aligned} P &= \left(\int\limits_0^{2\pi} \mathrm{d}\theta \right) \int\limits_0^{\infty} w\left(\frac{\delta_1(r)}{\zeta_1}\right) \left[\int\limits_{-1}^{1} r^2 w\left(\frac{\delta_2(r,t)}{\zeta_2}\right) \mathrm{d}t \right] \mathrm{d}r \\ &= 2\pi \sum_{k=1}^{n} \int\limits_{q_{k-1}\zeta_1}^{q_k\zeta_1} w_k\left(\frac{r}{\zeta_1}\right) f(r) \,\mathrm{d}r \end{aligned} \tag{5}$$

where $f(r)$ is the bracketed integral from the line above, i. e.,

$$f(r) = \sum_{l=1}^{n} f_l(r), \quad f_l(r) = r^2 \int\limits_{T_l(r)} w_l\left(\frac{\sqrt{r^2+\Delta^2-2\Delta r t}}{\zeta_2}\right) \mathrm{d}t,$$

and $T_l(r) \subseteq [-1,1]$ is the set of t for which $w_l\left(\frac{\delta_2(r,t)}{\zeta_2}\right)$ is defined, i. e.,

$$\begin{aligned} T_l(r) &= [-1,1] \cap \left\{ t : q_{l-1}\zeta_2 \leq \sqrt{r^2+\Delta^2-2\Delta r t} \leq q_l\zeta_2 \right\} \\ &= [-1,1] \cap \left[\frac{r^2+\Delta^2-q_l^2\zeta_2^2}{2\Delta r}, \frac{r^2+\Delta^2-q_{l-1}^2\zeta_2^2}{2\Delta r} \right]. \end{aligned}$$

One can easily verify that the indefinite integral is

$$
\begin{aligned}
& r^2 \int w_l\left(\frac{\sqrt{r^2+\Delta^2-2\Delta r t}}{\zeta_2}\right) \mathrm{d}t \\
= \; & r^2 \sum_{d=0}^{D} \frac{w_{ld}}{\zeta_2^d} \int \left(r^2+\Delta^2-2\Delta r t\right)^{\frac{d}{2}} \mathrm{d}t \\
= \; & -\frac{r}{\Delta} \sum_{d=0}^{D} \frac{w_{ld}}{(d+2)\,\zeta_2^d} \left(r^2+\Delta^2-2\Delta r t\right)^{\frac{d+2}{2}},
\end{aligned}
$$

hence, for any fixed r and index l we can solve $f_l(r)$ depending on the endpoints of $T_l(r)$, i. e., as in one of these cases:

1) $q_l\zeta_2 \leq |\Delta-r|$ or $\Delta+r \leq q_{l-1}\zeta_2$.
Then $T_l(r)=\emptyset$, implying $f_l(r)=0$.

2) $q_{l-1}\zeta_2 \leq |\Delta-r| \leq q_l\zeta_2 \leq \Delta+r$.
Then $T_l(r)=\left[\frac{r^2+\Delta^2-q_l^2\zeta_2^2}{2\Delta r},1\right]$, implying
$$f_l(r)=\frac{r\zeta_2^2}{\Delta}\sum_{d=0}^{D}\frac{w_{ld}}{d+2}\left(q_l^{d+2}-\left(\frac{|\Delta-r|}{\zeta_2}\right)^{d+2}\right).$$

3) $|\Delta-r| \leq q_{l-1}\zeta_2 \leq \Delta+r \leq q_l\zeta_2$.
Then $T_l(r)=\left[-1,\frac{r^2+\Delta^2-q_{l-1}^2\zeta_2^2}{2\Delta r}\right]$, implying
$$f_l(r)=\frac{r\zeta_2^2}{\Delta}\sum_{d=0}^{D}\frac{w_{ld}}{d+2}\left(\left(\frac{\Delta+r}{\zeta_2}\right)^{d+2}-q_{l-1}^{d+2}\right).$$

4) $q_{l-1}\zeta_2 \leq |\Delta-r| \leq \Delta+r \leq q_l\zeta_2$.
Then $T_l(r)=[-1,1]$, implying
$$f_l(r)=\frac{r\zeta_2^2}{\Delta}\sum_{d=0}^{D}\frac{w_{ld}}{d+2}\left(\left(\frac{\Delta+r}{\zeta_2}\right)^{d+2}-\left(\frac{|\Delta-r|}{\zeta_2}\right)^{d+2}\right).$$

5) $|\Delta-r| \leq q_{l-1}\zeta_2 \leq q_l\zeta_2 \leq \Delta+r$.
Then $T_l(r)=\left[\frac{r^2+\Delta^2-q_l^2\zeta_2^2}{2\Delta r},\frac{r^2+\Delta^2-q_{l-1}^2\zeta_2^2}{2\Delta r}\right]$, implying
$$f_l(r)=\frac{r\zeta_2^2}{\Delta}\sum_{d=0}^{D}\frac{w_{ld}}{d+2}\left(q_l^{d+2}-q_{l-1}^{d+2}\right).$$

Cubic Kernel Example: As an example we apply the above to the case of the cubic spline kernel (1). The kernel is defined by $n=2$, $q_0=0$, $q_1=1$, $q_2=2$, $w_1(q)=\frac{1}{4\pi}\left(3q^3-6q^2+4\right)$, and $w_2(q)=\frac{1}{4\pi}\left(-q^3+6q^2-12q+8\right)$. Hence, the kernel product integral can be computed according to

$$P=\frac{1}{2}\int_0^{\zeta_1}\left(3\frac{r^3}{\zeta_1^3}-6\frac{r^2}{\zeta_1^2}+4\right)f(r)\,\mathrm{d}r+\frac{1}{2}\int_{\zeta_1}^{2\zeta_1}\left(-\frac{r^3}{\zeta_1^3}+6\frac{r^2}{\zeta_1^2}-12\frac{r}{\zeta_1}+8\right)f(r)\,\mathrm{d}r$$

where, abbreviating $\kappa=\frac{\Delta+r}{\zeta_2}$ and $\lambda=\frac{|\Delta-r|}{\zeta_2}$,

$$f(r)=\frac{r\zeta_2^2}{4\pi\Delta}\begin{cases}
0, & 2\zeta_2\leq|\Delta-r| \\
4\left(4-\lambda^2\right)-4\left(8-\lambda^3\right)+\frac{3}{2}\left(16-\lambda^4\right)-\frac{1}{5}\left(32-\lambda^5\right), & \zeta_2\leq|\Delta-r|\leq 2\zeta_2\leq\Delta+r \\
4\left(\kappa^2-\lambda^2\right)-4\left(\kappa^3-\lambda^3\right)+\frac{3}{2}\left(\kappa^4-\lambda^4\right)-\frac{1}{5}\left(\kappa^5-\lambda^5\right), & \zeta_2\leq|\Delta-r|\leq\Delta+r\leq 2\zeta_2 \\
\frac{3}{10}+2\left(1-\lambda^2\right)-\frac{3}{2}\left(1-\lambda^4\right)+\frac{3}{5}\left(1-\lambda^5\right), & |\Delta-r|\leq\zeta_2\leq 2\zeta_2\leq\Delta+r \\
4\left(\kappa^2-1\right)-4\left(\kappa^3-1\right)+\frac{3}{2}\left(\kappa^4-1\right)-\frac{1}{5}\left(\kappa^5-1\right) & \\
\qquad +2\left(1-\lambda^2\right)-\frac{3}{2}\left(1-\lambda^4\right)+\frac{3}{5}\left(1-\lambda^5\right), & |\Delta-r|\leq\zeta_2\leq\Delta+r\leq 2\zeta_2 \\
2\left(\kappa^2-\lambda^2\right)-\frac{3}{2}\left(\kappa^4-\lambda^4\right)+\frac{3}{5}\left(\kappa^5-\lambda^5\right), & \Delta+r\leq\zeta_2
\end{cases}$$

B. Look-up Table

These formulas provide a legitimate way of computing the kernel product integral by first computing the coefficients of all pieces of $f(r)$, which is a continuous piecewise polynomial with degree smaller than or equal to $D+3$, with coefficient switches at the elements of $\bigcup_{l=0,\ldots,n}\{\Delta+q_l\zeta_2,|\Delta-q_l\zeta_2|\}$. One could then compute the coefficients of the products $w_k\left(\frac{r}{\zeta_1}\right)f(r)$ for $k=1,\ldots,n$, each of which again is a continuous piecewise polynomial whose integral could then be computed as the sum of the integrals of its pieces. Finally, the values of these integrals could be summed up and multiplied by 2π obtaining the kernel product integral according to (5).

However, in order to speed up the process of computing P we do not carry out the integrations for each triple (Δ,ζ_1,ζ_2) but compute a coefficient representation of $P(\Delta,\zeta_1,\zeta_2)$ during a precomputation step which has to be performed only once for a specified SPH kernel. We can then use this representation to efficiently evaluate P at any (Δ,ζ_1,ζ_2). As it turns out, the kernel product integral P is a piecewise trivariate Laurent polynomial. "Laurent polynomial" means that negative exponents may appear in its monomials. Its powers only appear in products $\Delta^a\zeta_1^b\zeta_2^c$ (multiplied by a coefficient) satisfying $a+b+c=3$ as well as $a\in\{-1,\ldots,2D+3\}$ and $b,c\in\{-D,\ldots,0,2,\ldots,D+4\}$. This means that for every piece of P the number of coefficients to be precomputed does not exceed $2D^2+9D+8$.

The coefficients of P change at all points (Δ,ζ_1,ζ_2) where $\Delta=q_i\zeta_1+q_j\zeta_2$ or $\Delta=\left|q_i\zeta_1-q_j\zeta_2\right|$ for any indices i and j. For the case of the cubic kernel (1) these points are plotted in Fig. 2. To facilitate the selection of the correct piece (i. e., the right coefficients set) of P, we first partition its domain into intervals for $\frac{\Delta}{\zeta_2}\geq 0$ in which the subdomain border functions for ζ_1 evaluate to a fixed (ascending) order. In other words, we seek the $\frac{\Delta}{\zeta_2}$ for which at least two of the border lines in Fig. 2 cross. They can be computed as the elements of the set

$$G=\bigcup_{\substack{a=1,\ldots,n\\ b=1,\ldots,a\\ c,d=0,\ldots,n}}\left\{\frac{q_aq_d+q_bq_c}{q_a+q_b},\frac{|q_aq_d-q_bq_c|}{q_a+q_b}\right\}\cup\bigcup_{\substack{a=2,\ldots,n\\ b=1,\ldots,a-1\\ c,d=0,\ldots,n}}\left\{\frac{q_aq_d+q_bq_c}{q_a-q_b},\frac{|q_aq_d-q_bq_c|}{q_a-q_b}\right\}.$$

Their number $|G|$ not only depends on n but also on the values q_k. For instance, for a kernel with two pieces ($n=2$) with $q_1=1$, and $q_2=2$ (as is the case for the cubic kernel (1) and depicted in Fig. 2 and 3), $G=\{0,\frac{1}{3},\frac{1}{2},\frac{2}{3},1,\frac{4}{3},\frac{3}{2},\frac{5}{3},2,3,4,5,6\}$ has 13 (distinct) elements, whereas, say, for $q_1=2$ and $q_2=7$ one would get $|G|=23$. Therefore, computing the elements of G ought to be done using rational arithmetic to prevent roundup errors from producing logic ones. After computing the elements of G we sort them, naming them g_k, $k=0,\ldots,|G|-1$ such that $0=g_0<\cdots<g_{|G|-1}$. Let $\Gamma_k=[g_{k-1},g_k)$ for $k=1,\ldots,|G|-1$ and $\Gamma_{|G|}=\left[g_{|G|-1},\infty\right)$ denote the right-open intervals bounded by the g_k.

Each of the cases $\frac{\Delta}{\zeta_2}\in\Gamma_k,k=1,\ldots,|G|$ can then be divided into subcases for which the coefficients of P stay fixed. We do this by sorting the elements of the corresponding set

$$H_k = \{0\} \cup \bigcup_{\substack{i=1,\dots,n\\ j=0,\dots,n}} \left\{ \left.\frac{q_j\zeta_2+\Delta}{q_i}\right|_{\frac{\Delta}{\zeta_2}\in\Gamma_k}, \left.\frac{|q_j\zeta_2-\Delta|}{q_i}\right|_{\frac{\Delta}{\zeta_2}\in\Gamma_k} \right\}$$

of linear border functions of Δ and ζ_2 restricted to the interval Γ_k for $\frac{\Delta}{\zeta_2}$. H_k always has $|H| = 2n^2+n+1$ elements. In analogy to G, we identify the sorted elements of H_k by h_l, $l = 0,\dots,|H|-1$, $0 = h_0 \le \dots \le h_{|H|-1}$. An example ordering for $k = 9$, $\Gamma_k = [2,3)$ can be seen in the yellow shaded area in Fig. 2.

This nested partitioning scheme for the domain of P allows us to save the coefficients of all pieces of P in a two-stage look-up table. The table contains the "keys" g_k and is organized in subtables, each of which corresponds to one of the intervals Γ_k. Each subtable consists of the two coefficients of the already sorted keys $h_l(\Delta,\zeta_2)$ and—for each of the intervals $[h_{l-1},h_l)$, $l = 1,\dots,|H|-1$ and $\left[h_{|H|-1},\infty\right)$—the coefficients of the corresponding piece of P.

This layout may seem improvable because we save the same coefficients sets several times. For instance, in case of the cubic kernel the 7th section of the first four subtables contains the same coefficients (gray shaded area in Fig. 2). However, this drawback is negligible as it does not affect the speed of the selection and evaluation algorithm after the table set-up phase, and the total number $|G|\left[1+\left(2n^2+n+1\right)\left(2D^2+9D+10\right)\right]$ of floating point values for the table fits easily into RAM for common kernels. In the example case of the cubic kernel (1) using double precision floating point values, the table has a size of 63,024 bytes.

Once the table has been computed, an evaluation of P at a triple (Δ,ζ_1,ζ_2) reduces to

1) Select the subtable corresponding to the interval Γ_k that includes $\frac{\Delta}{\zeta_2}$.
2) Evaluate the elements h_l of H_k at Δ and ζ_2. Their coefficients are given in the subtable.
3) From the subtable, select the coefficients set of P that corresponds to the interval $[h_{l-1},h_l)$ or $\left[h_{|H|-1},\infty\right)$ including ζ_1.
4) Evaluate the trivariate Laurent polynomial defined by these coefficients at (Δ,ζ_1,ζ_2).

In the special case of $p_1 = p_2$, we do not need the look-up table to compute P. Instead, during the precomputation phase, we calculate

$$K = 4\pi\int_0^{q_n}\left[r\,w(r)\right]^2\,\mathrm{d}r, \tag{6}$$

which is a constant only depending on the kernel, and then use it in computing the "kernel square integral" according to $P(0,\zeta_1,\zeta_1) = \zeta_1^3 K$.

While the table can be used to efficiently compute the kernel product integral for any values Δ, ζ_1, and ζ_2 as just explained, we can also use it for retrieving the subdomain endpoints and coefficients of any univariate piecewise Laurent polynomial P_{Δ,ζ_2}, i.e., P for fixed Δ and ζ_2 but variable ζ_1. In order to do this, after selecting the correct subtable, we evaluate not only the elements of H_k but also the coefficients of all pieces at Δ and ζ_2. We use this to compute the coefficients of an objective function for finding an optimal aggregate particle radius, as explained in Section V.

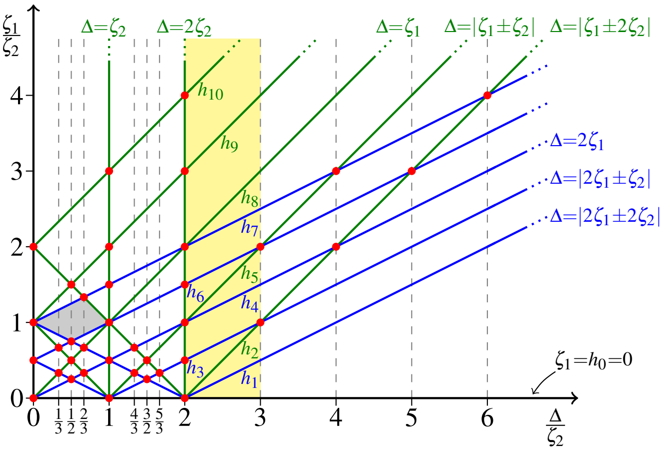

Fig. 2. Subdomain borders of $P(\Delta,\zeta_1,\zeta_2)$ for the case $n = 2, q_1 = 1, q_2 = 2$. The coefficients of the continuous piecewise Laurent polynomial function P change at the lines plotted in green and blue. Within the polygons bounded by these (e. g., the gray shaede area), the coefficients are constant. The elements of G are the abscissa coordinates of the intersection points depicted as red dots, i. e., the values for $\frac{\Delta}{\zeta_2}$ at which the order of the subdomain border lines changes. For instance, for $\frac{\Delta}{\zeta_2} \in \Gamma_9 = [2,3)$ (yellow shaded area), this order is illustrated by the labels h_i.

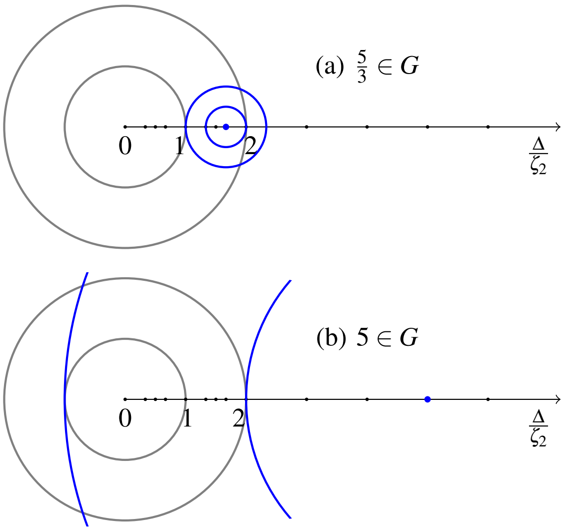

Fig. 3. The elements of G for $n = 2$, $q_1 = 1$, $q_2 = 2$, depicted as dots on the axis. The kernel coefficients for particle p_2 change at the "shells" depicted in gray. $\frac{\Delta}{\zeta_2} \in G$ if and only if there is a radius $\zeta_1 > 0$ for the "blue" particle p_1, such that there are two concurrent tangential shell-shell or shell-center contacts.

(a) The reason for $\frac{5}{3}$ being an element of G: For $\frac{\zeta_1}{\zeta_2} = \frac{1}{3}$, the shell for q_1 of particle p_1 touches the shell for q_2 of particle p_2 and vice versa.

(b) The reason for $5 \in G$: Two concurrent tangential shell contacts occur for $\frac{\zeta_1}{\zeta_2} = 3$.

V. Computing Aggregate Particles

Having precomputed the look-up table of the kernel product integral not only facilitates computing merge errors, it also helps in efficiently generating a suitable aggregate particle $\bar{p}$ for merging a given particle subset. We want to compute the attributes of $\bar{p}$, i. e., its position, mass, density, radius, and field values, such that its contribution to the attribute fields is most similar to the contributions of its children $\{p_i, i \in M\}$.

Since the aggregate particle's position $\bar{\mathbf{p}}$, mass $\bar{\mu}$, density $\bar{\rho}$, and radius $\bar{\zeta}$ affect all attribute fields at the same time, we have to prioritize some objective error metric. To account for different tasks and priorities, the user can provide weights $\eta_i \geq 0, i = 0, \dots, u+v$, $\sum_{i=0}^{u+v} \eta_i > 0$ for the attribute fields to define the objective scalar field

$$\begin{aligned} \gamma(\mathbf{x}) &= \sum_{i \in I} \gamma_i W_i(\mathbf{x}), \\ \gamma_i &= \eta_0 + \sum_{j=1}^{u} \eta_j \alpha_{ij} + \sum_{j=1}^{v} \frac{1}{3} \eta_{j+u} \left(\beta_{ij1} + \beta_{ij2} + \beta_{ij3}\right) \end{aligned}$$

whose merge error E_γ shall be used for setting "good" values for $\left(\bar{\mathbf{p}}, \bar{\mu}, \bar{\rho}, \bar{\zeta}\right)$. Formally, the objective field behaves like any other scalar field although the virtual values γ_i are not saved as proper particle attributes.

We start by setting the position $\bar{\mathbf{p}}$ to the heuristic "educated guess" value

$$\bar{\mathbf{p}} = \frac{\sum_{i \in M} \varphi_i \mathbf{p}_i}{\sum_{i \in M} \varphi_i}, \quad \varphi_i = \frac{\mu_i \gamma_i}{\rho_i},$$

i. e., to the sum of the children's positions weighted by their respective absolute contributions to the objective field. While this choice may compromise our aim to globally optimize all attributes of the aggregate particle, it will be sufficiently effective for most kernels which concentrate a particle's contribution closely to its position.

The remaining attributes will however be computed such that E_γ is indeed globally minimized, subject to the already fixed position and to some arbitrary lower bound $\zeta_{\min} > 0$ for the radius $\bar{\zeta}$. We propose to set $\zeta_{\min} = \frac{1}{q_n} \max_{i \in M} \|\bar{\mathbf{p}} - \mathbf{p}_i\|$ so that the volume of influence $B_{q_n \bar{\zeta}}[\bar{\mathbf{p}}]$ of $\bar{p}$ includes all child positions $\mathbf{p}_i$. In our experiments, this lower bound is "active", i. e., $\bar{\zeta} = \zeta_{\min}$, only in less than one percent of the cases.

Since the mass $\bar{\mu}$ serves just as a scaling factor to the other attributes, we can freely fix it without impacting optimality. In order to conserve the overall mass distribution in the best possible way, we simply set $\bar{\mu} = \sum_{i \in M} \mu_i$.

A. Minimizing the Objective Field Error

For fixed $\bar{\mathbf{p}}$ and $\bar{\mu}$, the objective field error

$$E_\gamma\left(\bar{\zeta}, \bar{\varphi}\right) = \sqrt{\int_{\mathbb{R}^3} \left(\frac{\bar{\varphi}}{\bar{\zeta}^3} \bar{\omega}(\mathbf{x}) - \sum_{i \in M} \frac{\varphi_i}{\zeta_i^3} \omega_i(\mathbf{x}) \right)^2 \mathrm{d}^3\mathbf{x}} \qquad (7)$$

only depends on the aggregate particle radius $\bar{\zeta}$ and the factor $\bar{\varphi} = \frac{\bar{\mu}\bar{\gamma}}{\bar{\rho}}$ where $\bar{\gamma} = \eta_0 + \sum_{j=1}^{u} \eta_j \bar{\alpha}_j + \sum_{j=1}^{v} \frac{1}{3} \eta_{j+u} \left(\bar{\beta}_{j1} + \bar{\beta}_{j2} + \bar{\beta}_{j3}\right)$.

In this section we describe a method for finding optimal values for $\bar{\zeta}$ and $\bar{\varphi}$, i. e., values which minimize $E_\gamma\left(\bar{\zeta}, \bar{\varphi}\right)$ subject to the constraint $\bar{\zeta} \geq \zeta_{\min}$. This two-dimensional problem can be reduced to a one-dimensional one when noting that for any given radius $\bar{\zeta}$ we can express the optimal $\bar{\varphi}$ as $\bar{\varphi}\left(\bar{\zeta}\right) = \frac{A_M\left(\bar{\zeta}\right)}{K}$ where K is the precomputed constant defined in (6) and

$$A_M\left(\bar{\zeta}\right) = \sum_{j \in M} \frac{\varphi_j}{\zeta_j^3} P\left(\left\|\bar{\mathbf{p}} - \mathbf{p}_j\right\|, \bar{\zeta}, \zeta_j\right).$$

This allows us to express E_γ only in terms of $\bar{\zeta}$ as

$$\begin{aligned} E_\gamma\left(\bar{\zeta}\right) &= \sqrt{B_M - \frac{\left[A_M\left(\bar{\zeta}\right)\right]^2}{K\bar{\zeta}^3}}, \\ B_M &= \sum_{i \in M} \sum_{j \in M} \frac{\varphi_i \varphi_j}{\zeta_i^3 \zeta_j^3} P\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|, \zeta_i, \zeta_j\right). \end{aligned} \qquad (8)$$

Its derivative vanishes iff

$$0 = C_M\left(\bar{\zeta}\right) = 3\bar{\zeta}^d A_M\left(\bar{\zeta}\right) - 2\bar{\zeta}^{d+1} A_M'\left(\bar{\zeta}\right), \qquad (9)$$

where A_M' is the first derivative of A_M. C_M is a piecewise polynomial of degree $2D+4$ with at most $|M|\left(2n^2+n\right)+1$ pieces, some of which may not have to be considered due to the restriction to the lower bound $\zeta_{\min}$. It can be shown that

$$\lim_{\bar{\zeta} \to \infty} A_M\left(\bar{\zeta}\right) = 4\pi w(0) \left[\int_{\mathbb{R}^3} r^2 w(r)\, \mathrm{d}r\right] \sum_{j \in M} \frac{\varphi_j}{\zeta_j^3} < \infty,$$

hence $\lim_{\bar{\zeta} \to \infty} E_\gamma\left(\bar{\zeta}\right) = \sqrt{B_M} = \sup_{\bar{\zeta} \in (0,\infty)} E_\gamma\left(\bar{\zeta}\right)$, which proves that we can always find an upper bound $\zeta_{\max}$ for $\bar{\zeta}$ for our search for optimizers.

We compute the coefficients of $A_M\left(\bar{\zeta}\right)$ using the look-up table of the kernel product integral coefficients. For every $j \in M$ we compute from it the subdomain endpoints and coefficients of the univariate piecewise Laurent polynomial $P_{\bar{\Delta}_j,\zeta_j}$, $\bar{\Delta}_j = \|\bar{\mathbf{p}} - \mathbf{p}_i\|$ as explained at the end of Section IV. Each of the coefficients is then multiplied by $\frac{\varphi_j}{\zeta_j^3}$. After sorting the subdomain endpoints of A_M, which are all subdomain endpoints of the $P_{\bar{\Delta}_j,\zeta_j}$, for each subdomain of A_M the coefficients of A_M are computed as the sums of the corresponding coefficients of the $\frac{\varphi_j}{\zeta_j^3} P_{\bar{\Delta}_j,\zeta_j}$.

From the coefficients of A_M we compute the coefficients of C_M according to (9) as well as of all its derivatives. Afterwards, for every domain interval and every order $o = 2d+4, \dots, 1$ we compute the roots of the o'th derivative $C_M^{(o)}$ that lie inside the interval. For this we use a Newton-Raphson method guarded by bisection, starting with two points at which $C_M^{(o)}$ has opposite signs. The roots of the o'th derivative give us the extremizers of the $(o-1)$'th derivative, which serve as starting points for the root finding procedure for order $o-1$, and so on. Having found all optimizer candidates, we compute $\frac{1}{\bar{\zeta}^3}\left[A_M\left(\bar{\zeta}\right)\right]^2$ for each of them to find the global optimizer.

The root-finding algorithm can be further optimized, since for order $o > 0$ every root of $C_M^{(o)}$ has to be computed only to the precision needed to decide whether $C_M^{(o-1)}$ evaluates to a positive or a negative value at its extremizer. The algorithm can be implemented in a perfectly stable and reliable fashion and has proven to be reasonably fast. However, any root-finding algorithm may serve well here.

B. Minimizing and Computing Field Errors

In this section, we present formulas suitable for computing the remaining aggregate particle attributes. Now that we have fixed $\bar{\mathbf{p}}$, $\bar{\mu}$, and $\bar{\zeta}$, the values $\psi_i = \frac{\mu_i}{\rho_i \zeta_i^3} P\left(\bar{\Delta}_i, \bar{\zeta}, \zeta_i\right)$ are also fixed. Just like $\bar{\mu}$, we can choose the density $\bar{\rho} > 0$ freely without restricting optimality in the field errors. We just choose the value $\bar{\rho} = \frac{K\bar{\mu}}{\sum_{i\in M}\psi_i}$, globally minimizing the "weight error" $E_W = \sqrt{\int_{\mathbb{R}^3}\left(\bar{W}(\mathbf{x}) - \sum_{i\in M} W_i(\mathbf{x})\right)^2 \mathrm{d}^3\mathbf{x}}$. For every scalar field α, we can then compute the aggregate particle's attribute as $\bar{\alpha} = \frac{\bar{\rho}}{K\bar{\mu}}\sum_{i\in M}\psi_i\alpha_i$, which globally minimizes this field's merge error to

$$E_\alpha = \sqrt{\left[\sum_{i\in M}\sum_{j\in M}\frac{\mu_i\mu_j\alpha_i\alpha_j}{\rho_i\rho_j\zeta_i^3\zeta_j^3}P\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|, \zeta_i, \zeta_j\right)\right] - \frac{1}{K\bar{\zeta}^3}\left(\sum_{i\in M}\psi_i\alpha_i\right)^2}.$$

Note that, while this formula for E_α is computationally less expensive than (2), it requires $\bar{\alpha}$ to be optimal.

Each component of every vector field $\mathbf{b} = (\beta_1, \beta_2, \beta_3)$ can be computed just like a scalar field, which minimizes its error to

$$E_{\mathbf{b}} = \sqrt{E_{\beta_1}^2 + E_{\beta_2}^2 + E_{\beta_3}^2}.$$

C. Propagating Merge Errors with respect to Original Fields

The attribute field errors computed so far reflect the error produced by one merge. However, if an aggregate particle's children are also aggregate particles, it would be beneficial to know its error with respect to its original descendants: How much has an attribute field changed during all the merges leading to this aggregate? We cannot compute this error without knowing the data of all these original particles. We can, however, provide an upper bound: Due to the L^2 norm fulfilling the triangle inequality, we can compute this upper bound by computing the sum of the children's field error bounds and adding it to the error caused by the current merge. This absolute error bound may be useful for efficiently selecting particles from a hierarchy.

VI. RESULTS

We have implemented our particle merging method in C++ and tested it, calling it about one million times on subsets of 2 to 20 particles selected from a real-world SPH data set with $4 \cdot 10^6$ particles. In addition to position, mass, density, and radius, every particle had 4 scalar attributes and 2 vector attributes. For each selected particle subset, an optimal aggregate particle was constructed along with the merge errors for all additional attribute fields, measuring the time needed. The experiments were made on a PC with an Intel Core i7-3930K CPU with 3.2 GHz frequency (3.8 GHz boost).

The computational cost of the method depends heavily on the kernel used. While we can set our program to use any kernel fulfilling the requirements stated in Section III, for our experiments we restricted ourselves to the cubic kernel defined in (1).

While the precomputation of the look-up table containing the kernel product integral coefficients is done in less than one second, the computation time for the merging depends mostly on the number of children $|M|$. $A_M\left(\bar{\zeta}\right)$ is a weighted sum of $|M|$ univariate piecewise Laurent polynomials $P_{\bar{\Delta}_j,\zeta_j}$, each of which has 11 pieces, leading to up to $10|M|+1$ pieces for A_M. Therefore, for computing the coefficients of A_M, we need quadratic time with respect to $|M|$. For the remaining steps of computing optimal attribute values for an aggregate particle, linear time complexity applies.

In order to obtain the error values for the attribute fields, we have to compute all kernel product integrals $P\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|, \zeta_i, \zeta_j\right), i, j \in M$, which is almost the same as computing the summands of B_M, as defined in (8). There are $\binom{|M|}{2}$ kernel product integrals to be computed, resulting in quadratic time complexity for computing error values. This corresponds to our time measuring experiments, the results of which are summarized in Fig. 4. For any number of children, computing A_M and B_M and optimizing the radius $\bar{\zeta}$ greatly dominates the effort of the total merging procedure. These three steps amount to about 79% of the total time for $|M| = 2$ and about 91% for $|M| = 20$.

Our implementation needs about $29.5\mu s$ for computing an aggregate particle and error values for $|M| = 2$. In terms of figures, this corresponds to about $2 \cdot 10^6$ particle mergings in one minute or nearly $3 \cdot 10^9$ in one day, not considering a possible speedup by merging several particle groups in parallel on a multi-core CPU. The method therefore seems to be suitable for building multi-resolution hierarchies for large SPH data sets. However, in order to handle large data sets, a complex out-of-core framework for loading and selecting particles, calling the merging procedure and saving the output aggregate particles, would have to be used.

VII. CONCLUSION AND OUTLOOK

We have presented a method for computing L^2 measures for the error inflicted on attribute fields by merging an arbitrary number of particles. Its efficiency is achieved by the use of a precomputed and efficiently accessible look-up table storing the coefficients of the kernel product integral as a function of the distance between two particles and the particles' radii of influence.

Based on the same precomputed table, we laid out a method for computing an optimal aggregate particle for merging a given particle subset. Except for the aggregate particle position, all attributes are computed such that the L^2 norm of the change in the attribute fields is minimal. For prioritization, the user can provide weights for the attribute fields, according to which the optimal radius is calculated.

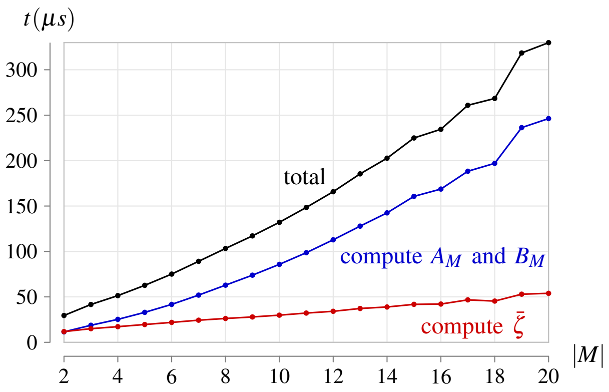

Fig. 4. Plot of the average time needed for computing an aggregate particle as a function of the number of children $|M|$. Time values are given in microseconds. The blue plot shows the time needed to compute the coefficients of $A_M\left(\bar{\zeta}\right)$ and the summands of B_M, red shows the time needed for finding the optimal radius $\bar{\zeta}$ afterwards, and black shows the total time needed, including the computation of all the remaining attributes and error values.

Our methods facilitate the construction of multi-purpose particle-based multi-resolution hierarchies useful for visualizing SPH data in various ways. The computation times stay within reasonable limits, allowing the construction of such hierarchies for large data sets.

Beyond our main purpose of constructing hierarchies, our findings and the merging method may be applicable to any field in which SPH data is being manipulated, such as for the reduction of resolution during SPH simulations.

Nonetheless, our method leaves room for further improvement, e. g., by providing a means to also rigorously optimize the position of the aggregate particle. In addition, one could try to expand it to be applicable to time-varying SPH data, by finding ways to reduce the resolution in the time dimension beyond just skipping some time steps, or by designing a strategy to efficiently update particle-based hierarchies from one time step to the next. Moreover, since most of the computation comprises adding and multiplying coefficients of piecewise defined functions, which seems to be well parallelizable, a significant speed-up by porting the algorithm to be run on the GPU is to be expected. However, advances are most probable in the application of our method to novel strategies for multi-resolution hierarchy construction for visualization.

References

[1] A. Gastelum, P. Delmas, M. Lefrancq, C. Duwig, J. Marquez, B. Prado, G. Gimel'farb, and P. Charrier, "Visualising 3D porous media fluid interaction using X-ray CT data and smooth particles hydrodynamics modelling," in *Proceedings of the International Conference of Image and Vision Computing New Zealand*, pp. 1–8, 2010.

[2] E. Rustico, G. Bilotta, G. Gallo, A. Herault, and C. Del Negro, "Smoothed particle hydrodynamics simulations on multi-GPU systems," in *Proceedings of the Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pp. 384–391, 2012.

[3] D. J. Price, "SPLASH: An interactive visualisation tool for smoothed particle hydrodynamics simulations," *Publications of the Astronomical Society of Australia*, vol. 24, no. 3, pp. 159–173, 2007.

[4] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner, "Parallel surface reconstruction for particle-based fluids," *Computer Graphics Forum*, vol. 31, no. 6, pp. 1797–1809, 2012.

[5] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled SPH and particle level set fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797–804, 2008.

[6] P. Rosenthal and L. Linsen, "Direct isosurface extraction from scattered volume data," in *EuroVis06: Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization* (B. S. Santos, T. Ertl, and K. I. Joy, eds.), (Aire-la-Ville, Switzerland), pp. 99–106, Eurographics Association, 2006.

[7] P. Rosenthal, S. Rosswog, and L. Linsen, "Direct surface extraction from smoothed particle hydrodynamics simulation data," in *Proceedings of the 4th High-End Visualization Workshop* (W. Benger, R. Heinzl, and W. Kapferer, eds.), (Berlin, Germany), pp. 50–61, Lehmanns Media, 2007.

[8] P. Rosenthal and L. Linsen, "Smooth surface extraction from unstructured point-based volume data using PDEs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1531–1546, 2008.

[9] P. Rosenthal, V. Molchanov, and L. Linsen, "A narrow band level set method for surface extraction from unstructured point-based volume data," in *Proceedings of WSCG* (V. Skala, ed.), (Plzen, Czech Republic), pp. 73–80, UNION Agency – Science Press, 2 2010.

[10] D. Cha, S. Son, and I. Ihm, "GPU-assisted high quality particle rendering," *Computer Graphics Forum*, vol. 28, no. 4, pp. 1247–1255, 2009.

[11] T. Kroes, F. H. Post, and C. P. Botha, "Exposure render: An interactive photo-realistic volume rendering framework," *PLoS ONE*, vol. 7, no. 7, p. e38586, 2012.

[12] F. Lindemann and T. Ropinski, "Advanced light material interaction for direct volume rendering," in *Proceedings of the IEEE/EG international conference on Volume Graphics*, pp. 101–108, Eurographics Association, 2010.

[13] T. Ropinski, C. Döring, and C. Rezk-Salama, "Interactive volumetric lighting simulating scattering and shadowing," in *Proceedings of the IEEE Pacific Visualization Symposium*, pp. 169–176, 2010.

[14] R. Kaehler, O. Hahn, and T. Abel, "A novel approach to visualizing dark matter simulations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2078–2087, 2012.

[15] Y. Jang, B. Schindler, R. Fuchs, and R. Peikert, "Volumetric evaluation of meshless data from smoothed particle hydrodynamics simulations," in *Proceedings of the Symposium on Volume Graphics* (R. Westermann and G. Kindlmann, eds.), pp. 45–52, 2010.

[16] J. H. Clark, "Hierarchical geometric models for visible surface algorithms," *Commun. ACM*, vol. 19, no. 10, pp. 547–554, 1976.

[17] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[18] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen, "Hierarchical streamline bundles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1353–1367, 2012.

[19] H. T. Vo, S. P. Callahan, P. Lindstrom, V. Pascucci, and C. T. Silva, "Streaming simplification of tetrahedral meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 145–155, 2007.

[20] Z. Du and Y.-J. Chiang, "Out-of-Core Simplification and Crack-Free LOD Volume Rendering for Irregular Grids," *Computer Graphics Forum*, vol. 29, no. 3, pp. 873–882, 2010.

[21] 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*, vol. 15, no. 6, pp. 1251–1258, 2009.

[22] R. Fraedrich, S. Auer, and R. Westermann, "Efficient high-quality volume rendering of SPH data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1533–1540, 2010.

[23] J. Feldman and J. Bonet, "Dynamic refinement and boundary contact forces in sph with applications in fluid flow problems," *International Journal for Numerical Methods in Engineering*, vol. 72, no. 3, pp. 295–324, 2007.

[24] Q. Xiong, B. Li, and J. Xu, "GPU-accelerated adaptive particle splitting and merging in SPH," *Computer Physics Communications*, vol. 184, no. 7, pp. 1701–1707, 2013.

[25] B. Solenthaler and M. Gross, "Two-scale particle simulation," *ACM Trans. on Graphics*, vol. 30, no. 4, pp. 81:1–81:8, 2011.

[26] S. Rosswog, "Astrophysical smooth particle hydrodynamics," *New Astronomy Reviews*, vol. 53, no. 4–6, pp. 78–104, 2009.