

# Particle-Wise Higher-Order SPH Field Approximation for DVR

Jonathan Fischer<sup>1</sup>, Martin Schulze<sup>1</sup>, Paul Rosenthal<sup>2</sup><sup>a</sup> and Lars Linsen<sup>3</sup><sup>b</sup>

<sup>1</sup>Department of Computer Science, Chemnitz Technical University, Str. der Nationen 62, 09111 Chemnitz, Germany

<sup>2</sup>Institute for Visual and Analytic Computing, University of Rostock, Albert-Einstein-Str. 22, 18059 Rostock, Germany

<sup>3</sup>Institute of Computer Science, University of Münster, Einsteinstr. 62, 48149 Münster, Germany

**Keywords:** Scientific Visualization, Ray Casting, Higher-Order Approximation, Volume Rendering, Scattered Data, SPH.

**Abstract:** When employing Direct Volume Rendering (DVR) for visualizing volumetric scalar fields, classification is generally performed on a piecewise constant or piecewise linear approximation of the field on a viewing ray. Smoothed Particle Hydrodynamics (SPH) data sets define volumetric scalar fields as the sum of individual particle contributions, at highly varying spatial resolution. We present an approach for approximating SPH scalar fields along viewing rays with piecewise polynomial functions of higher order. This is done by approximating each particle contribution individually and then efficiently summing the results, thus generating a higher-order representation of the field with a resolution adapting to the data resolution in the volume.

## 1 INTRODUCTION

Introduced by Gingold and Monaghan (Gingold and Monaghan, 1977) and independently by Lucy (Lucy, 1977), **Smoothed Particle Hydrodynamics** (SPH) is a group of methods for simulating dynamic mechanical processes, typically fluid or gas flows but also solid mechanics. The objective matter is modeled by means of particles, each representing a small portion of a simulated substance and attributed a specific mass, density, and other physical measures.

These discrete and scattered particles define scalar and vector fields on the spatial continuum through an interpolation rule, determining a physical field as the sum of the isotropic contributions of the particles, each following a smooth function only of the distance from the particle position, called the kernel function. As an example, we refer to the SPH kernel function defined by the cubic B-spline (Rosswog, 2009)

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

In this work, we assume the kernel function to have compact support. We consider this to be a minor restriction because kernel functions with compact support are appreciated in the SPH community for


bounding the particles' volumes of influence. Obviously, by simply defining some cut-off value as upper bound, any kernel function can be supported. Given such a function, a particle's contribution to a scalar field can be expressed as


$$\frac{\mu\alpha}{\rho\zeta^3} w\left(\frac{\|x-\chi\|}{\zeta}\right)$$

for the particle's position  $\chi$ , mass  $\mu$ , target field value  $\alpha$ , density  $\rho$ , and smoothing radius  $\zeta$ . While  $\mu$ ,  $\alpha$ , and  $\rho$  serve as simple multiplicative constants, defining only the amplitude of the contribution,  $\zeta$  radially scales the domain and thus defines the radius of the particle's volume of influence.

**Direct Volume Rendering (DVR):** has a long-standing tradition for visualizing scalar volumetric fields (Drebin et al., 1988) and is commonly implemented as a ray casting method. It builds on assigning visual characteristics, representing features of the field, to the target domain volume and then rendering it by casting viewing rays through the volume. The color of each pixel of the output image is the result of simulating the behavior of light traveling through the volume along a ray in opposite viewing direction.

Said visual characteristics, usually comprising light emission and absorption, are computed from local target field characteristics, like values or gradients, in a process called classification (Max, 1995). It is commonly employed based on a piecewise constant or piecewise linear approximation of the target scalar

<sup>a</sup> <https://orcid.org/0000-0001-9409-8931>

<sup>b</sup> <https://orcid.org/0000-0002-6168-8748>

field along the ray.

Direct volume rendering of astronomical SPH data was first performed by incorporating contributions of both grid and particle data into the optical model (Kähler et al., 2007). Efficient full-featured DVR applied directly to large SPH data sets on commodity PC hardware has first been managed by re-sampling the scalar fields to a perspective grid held in a 3-d texture for each rendering frame (Fraedrich et al., 2010). Later, DVR for large scattered data was proposed to be performed on the CPU by Knoll et al. (Knoll et al., 2014) and mostly applied to molecular dynamics. Although employing radial basis function kernel (RBF) interpolation similar to the SPH case, they focus on the very special task of rendering a surface defined by a density field. This is in line with more recent contributions in rendering SPH data, such as (Hochstetter et al., 2016) or (Oliveira and Paiva, 2022), targeting only simulated fluids.

The approaches for volume-rendering arbitrary SPH fields employ equidistant sampling of the target field along viewing rays and acting on a piecewise linear approximation of it, which, depending on the local particle density, may miss detailed features in regions of high particle density, or oversample particles with large smoothing length in sparse regions.

Thus, in this work, we explore the capabilities and limitations of approximating SPH scalar fields with higher-order piecewise polynomial functions, whose resolution adapts to the local resolution of the given particle data. The higher-order approximations may facilitate quantitatively more accurate output images at a worthwhile cost.

## 2 METHOD OVERVIEW

As SPH interpolation defines the target scalar field as the sum of particle contributions to it, our concept builds on approximating each individual particle contribution on a viewing ray and then employing the sum of these approximations. Although summing an indefinite number of piecewise polynomial functions may seem like a task of quadratic complexity not amenable to GPU processing, we encode these functions in a way reducing the summation to a simple sorting task. Our method comprises three passes:

1. For each particle, approximate its contribution to all relevant viewing rays.
2. For each ray, sort the contribution data assigned to it with respect to the distance from the viewer.
3. For each ray, accumulate the contributions along the ray, classify and composite the result.

In the remainder of this section, we declare our concept of representing piecewise polynomial functions and show how these can be processed during the compositing sweep. In Section 3, we present a way to efficiently compute optimal approximations for single particle contributions during the first sweep. In Section 4, we then describe a pitfall that our higher-order approximation scheme involves and develop an improvement of our method to overcome this difficulty. Finally, in Section 5, we analyze the approximation errors implied by our scheme and discuss the choice of the major parameters like the approximation order, before concluding our work in Section 6.

**Localized Difference Coefficients.** A real polynomial function  $A : \mathbb{R} \rightarrow \mathbb{R}$  of order  $D$  is generally represented by its coefficients, i. e., numbers  $a_d \in \mathbb{R}$  such that  $A(t) = \sum_{d=0}^D a_d t^d$ . They provide a direct image of how the function and all its derivatives behave at  $t = 0$  since the  $d^{\text{th}}$  derivative of  $A$  at  $t = 0$  amounts to  $d! a_d$ . If we wanted to know the value or derivative of order  $d$  at some other  $t_*$ , we could compute it as

$$A^{(d)}(t_*) = \sum_{j=d}^D \frac{j!}{(j-d)!} a_j t_*^{j-d} = d! \sum_{j=d}^D \binom{j}{d} a_j t_*^{j-d},$$

which shows that the real numbers

$$a_{*d} = \sum_{j=d}^D \binom{j}{d} a_j t_*^{j-d}$$

represent the order- $d$  behavior of  $A$  at  $t = t_*$  just as the  $a_d$  do at  $t = 0$ . In fact,  $A$  can be expressed using these *localized coefficients* as

$$A(t) = \sum_{d=0}^D a_{*d} (t - t_*)^d.$$

Now, a continuous piecewise polynomial function is defined by a sequence of border arguments  $t_0 < t_1 < \dots$  and several polynomial functions  $A_0, A_1, \dots$  such that each polynomial function  $A_k$  is applied in its respective interval  $[t_k, t_{k+1}]$ .

If we were representing each polynomial  $A_k$  by its coefficients, summing several piecewise polynomial functions would require tracking the applicable polynomials for each resulting argument interval and summing their coefficients.

Instead, we save for each border argument  $t_k$  the change that the overall function performs in all orders. Specifically, we save the localized coefficients of the difference between the polynomials  $A_k$  applicable on the right of  $t_k$  and  $A_{k-1}$  applicable on its left.

Thus, when saving a piecewise polynomial approximation of order  $D$  along a viewing ray, we encode it as a sequence of *knots*

$$(t_k, \hat{a}_{k0}, \dots, \hat{a}_{kD}), \quad k = 0, 1, \dots$$

consisting of a ray parameter  $t_k$  defining the knot position on the ray and *localized difference coefficients*  $\hat{a}_{k0}, \dots, \hat{a}_{kD}$ , such that

$$A_k(t) - A_{k-1}(t) = \sum_{d=0}^D \hat{a}_{kd} (t - t_k)^d.$$

Summing several piecewise polynomials encoded this way amounts to nothing more than sorting their joint knots for increasing knot positions. The resulting approximation can then be processed piece by piece, retrieving the localized coefficients  $a_{k0}, \dots, a_{kD}$  of each piece polynomial  $A_k$  at its left boundary argument  $t_k$  from the ones of the last piece according to the updating rule

$$a_{kd} = \hat{a}_{kd} + \sum_{j=d}^D \binom{j}{d} a_{(k-1)j} (t_k - t_{k-1})^{j-d} \quad (2)$$

for  $d = 0, \dots, D$ , which is a computation of constant complexity per piece, irrespective of the number of particles contributing.

### 3 APPROXIMATING SINGLE PARTICLE CONTRIBUTIONS

#### 3.1 Deduction from Unit Particle

Since the same SPH kernel function is applied for all particles, their contributions to the volume differ in only a few scaling and translation parameters, namely their position  $\chi$ , mass  $\mu$ , density  $\rho$ , smoothing radius  $\zeta$ , and applicable scalar field attribute  $\alpha$ .

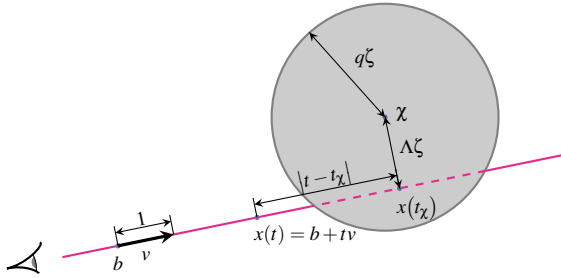


Figure 1: Sketch of measures involved in the positional relationship between viewing ray and particle. The volume of influence of a particle with smoothing length  $\zeta$  is intersected by a viewing ray, defined by base point  $b$  and unit direction vector  $v$ . The particle's contribution on the ray at a point  $x(t)$  is determined by its distance to the particle position  $\chi$ . We denote by  $q$  the upper bound of the kernel function's support, such that  $q\zeta$  is the radius of the particle's volume of influence.

To model a viewing ray, we fix a straight line with vector equation  $x(t) = b + tv$  for some base point

$b \in \mathbb{R}^3$ , unit direction vector  $v \in \mathbb{R}^3$ , and parameter  $t \in \mathbb{R}$ . We consider a normalized particle, i. e., one of unit mass, unit density, unit field attribute, and unit smoothing radius. Assuming the base point  $b$  on the ray is the one closest to the particle position, the normalized particle's contribution to this line is

$$B_\Lambda(t) = w\left(\frac{\sqrt{\Lambda^2 + t^2}}{\zeta}\right),$$

where  $\Lambda$  is the distance between the particle's position and the line. Then, the contribution of a specific data particle on a ray at distance  $\Lambda\zeta$  from its position  $\chi$  amounts to

$$\frac{\mu\alpha}{\rho\zeta^3} B_\Lambda\left(\frac{t - t_\chi}{\zeta}\right),$$

where  $t_\chi$  is the parameter of the point on the ray closest to the particle position and  $\Lambda = \frac{1}{\zeta} \|\chi - x(t_\chi)\|$ . The situation is depicted in Figure 1.

Finding optimal piecewise polynomial approximations for  $B_\Lambda$  for all  $\Lambda$  within the SPH kernel's support suffices to generate optimal approximations for any particle by just translating and scaling it in the same way. For quick access, we thus prepare a look-up table containing the localized difference coefficients of order- $D$  approximations of  $B_\Lambda$  for many equidistant values of  $\Lambda$ . In the remainder of this section, we show how we can find these optimal approximations.

#### 3.2 Optimization Problem Definition

Before we can find optimal approximations, we need to define what optimality shall mean in this context. Specifically, we have to settle on:

1. The space of eligible candidates, i. e., the condition of what we want to consider a feasible approximation.
2. The error measure defining whether one approximation is better than another.

The main restriction on the space of eligible approximation candidates is the imposition of a maximum polynomial degree  $D$ , i. e., the order of approximation, and the number  $K$  of non-trivial polynomial pieces per particle, i. e., the number of non-zero polynomials defining the approximation of a single particle. We dedicate Section 5 to evaluating choices of  $D$  and  $K$  but leave them unspecified for now. Beyond that, we demand that our approximations shall be continuous and even functions with compact support. This seems reasonable, as our approximation target  $B_\Lambda$  also has these properties.

For measuring the approximating quality of any piecewise polynomial function candidate  $S : \mathbb{R} \rightarrow \mathbb{R}$ ,

we apply its  $L^2$  distance to  $B_\Lambda$ , i. e., we seek to minimize the approximation error

$$E_\Lambda(S) = \|S - B_\Lambda\|_2 = \left( \int_{\mathbb{R}} [S(t) - B_\Lambda(t)]^2 dt \right)^{\frac{1}{2}}. \quad (3)$$

In contrast to the supremum norm often used in approximation theory, this error measure punishes not only the maximum pointwise deviation from the target, but also the length of segments of high deviation. Moreover, another advantage of the  $L^2$  norm is its associated inner product, which allows us to generate a closed-form solution of the optimal approximation and its error value as a function only of the knot positions, as shown in Section 3.3. We then find optimal knot positions through standard non-linear optimization as explained in Section 3.4. Detailed proofs of our findings can be found in the appendix of our preprint (Fischer et al., 2024) hosted on arxiv.org.

### 3.3 Solution for Fixed Knot Positions

As a prerequisite for finding truly optimal approximations, we first handle the case of arbitrary fixed knot positions. Due to our evenness requirement, the negative knot positions are determined from the positive ones. Hence, our optimization domain is the set of even continuous piecewise polynomial functions with compact support, maximal degree  $D$ , maximal number of non-trivial pieces  $K$ , and positive knot positions  $\theta_1, \dots, \theta_{\lceil K/2 \rceil}$ . We denote this set by  $\mathcal{S}$ .

Consider the vector space  $L^2(\mathbb{R})$  of square-integrable functions  $\mathbb{R} \rightarrow \mathbb{R}$ , which by the Riesz-Fischer theorem is complete with respect to the  $L^2$  norm and therefore a Hilbert space when equipped with the inner product

$$\langle A, B \rangle = \int_{\mathbb{R}} A(t)B(t) dt \quad \text{for } A, B \in L^2(\mathbb{R}),$$

which induces the  $L^2$  norm  $\|A\|_2 = \sqrt{\langle A, A \rangle}$ . The approximation target  $B_\Lambda$  is clearly an element of  $L^2(\mathbb{R})$  as it is continuous and has compact support.

For any  $A, B \in L^2(\mathbb{R})$ ,  $B \neq \mathbf{0}$ , we denote by

$$\text{proj}_B(A) = \frac{\langle A, B \rangle}{\langle B, B \rangle} B$$

the orthogonal projection of  $A$  on  $B$ .

As we will see shortly,  $\mathcal{S}$  is a subspace of  $L^2(\mathbb{R})$  of finite dimension  $\lfloor \frac{KD}{2} \rfloor$ , for which we can compute an orthogonal basis  $\tilde{\mathcal{A}}$ , which in turn we can use to calculate the orthogonal projection of  $B_\Lambda$  on  $\mathcal{S}$  as

$$S_\Lambda = \sum_{A \in \tilde{\mathcal{A}}} \text{proj}_A(B_\Lambda). \quad (4)$$

It is easy to show that  $S_\Lambda$  is the unique error-optimal approximation of  $B_\Lambda$  among the elements of  $\mathcal{S}$  (proof in (Fischer et al., 2024)). Therefore, all we need for computing the optimal approximation for fixed knot positions is a suitable orthogonal basis.

We specify a non-orthogonal basis as a starting point here: Let  $\mathcal{J}$  be the set of pairs  $(k, d)$  of positive integers  $k \leq \lceil K/2 \rceil$  and  $d \leq D$  but excluding elements  $(1, d)$  for uneven  $d$  if  $K$  is uneven. Then the set  $\tilde{\mathcal{A}} = \{\tilde{A}_{kd} : (k, d) \in \mathcal{J}\}$  of functions

$$\tilde{A}_{kd}(t) = \begin{cases} 1 & \text{if } |t| \leq \theta_{k-1} \\ 1 - \left( \frac{|t| - \theta_{k-1}}{\theta_k - \theta_{k-1}} \right)^d & \text{if } \theta_{k-1} \leq |t| \leq \theta_k \\ 0 & \text{if } |t| \geq \theta_k \end{cases}$$

is a basis of  $\mathcal{S}$  (proof in (Fischer et al., 2024)).

Given  $\tilde{\mathcal{A}}$ , we convert it into an orthogonal basis  $\mathcal{A} = \{A_{kd} : (k, d) \in \mathcal{J}\}$  by employing the Gram-Schmidt process. Specifically, we recursively set

$$A_{k^*d^*} = \tilde{A}_{d^*k^*} - \sum_{\substack{(k,d) \in \mathcal{J} \\ k < k^* \vee (k=k^* \wedge d < d^*)}} \text{proj}_{A_{kd}}(\tilde{A}_{d^*k^*})$$

in lexicographical order of pairs  $(k^*, d^*) \in \mathcal{J}$ .

### 3.4 Optimal Knot Positions

While we can directly compute optimal approximations for given knot positions as shown above, finding error-optimal knot positions is a nonlinear optimization problem over the variables  $\theta_1, \dots, \theta_{\lceil K/2 \rceil}$ .

The objective function  $E_\Lambda$  is continuous within the interior of the feasible domain defined by the constraints  $0 < \theta_1 < \dots < \theta_{\lceil K/2 \rceil}$ , due to the continuity of the inner product and of the constructed basis with respect to the  $\theta_k$ , which would even hold for a discontinuous SPH kernel function. Also, we can expect to find a global optimizer within the interior of the feasible domain, i. e., without any of the constraints being active, because an equality of any two variables would be equivalent to a reduction of the number of non-zero pieces, diminishing the freedom for approximating and therefore resulting in higher or equal error. Hence, if a local optimum was attained at the feasible domain border, it could not be isolated because shifting one of the colliding  $\theta_k$  and defining the polynomials on both of its sides to be equal would result in the same approximation function and therefore in the same error value.

While evaluating  $E_\Lambda$  could be done following the steps above for every set of fixed  $\theta_1, \dots, \theta_{\lceil K/2 \rceil}$ , it is worthwhile to only fix  $K$  and  $D$  and perform the process in a symbolic manner, generating an explicit formula of the objective error function, which can later

be evaluated for any knot positions and distance parameter  $\Lambda$ . The generation of a closed-form representation of an orthogonal basis for variable knot positions and  $\Lambda$  has to be done only once as it does not depend on the SPH kernel used. However, the explicit expression of the approximation error following (4) requires closed-form solutions of the integrals defining the inner products  $\langle A, B_\Lambda \rangle$  for orthogonal basis functions  $A$ . In the case of an SPH kernel defined by a continuous piecewise polynomial function such as the cubic B-spline kernel (1), this can clearly be achieved. In any case, as the symbolic computations are rather involved, computer algebra systems are of great help during this preparatory process.

We have conducted this process for the cubic B-spline kernel, going to considerable length to find a global optimizer for many discrete  $\Lambda$ . For  $\Lambda$  close to the upper bound  $q$ , however, we have found the evaluation of some of the formulae to become unstable. To obtain reliable results, we employed the GNU MPFR library to perform the computations in multiple precision arithmetic. Since we have not encountered any severe problems during these optimizations, we have reason to hope they are manageable for any piecewise polynomial SPH kernel function.

## 4 QUANTIZATION TO PREVENT HIGHER-ORDER ERRORS

### 4.1 Higher-Order Rounding Error Propagation

For the stated efficiency reasons, the recursive computation of polynomial coefficients according to (2) is an integral component to our particle-wise approximation approach. However, it comes at a substantial price which may not be obvious at first glance. Directly applying (2) during the compositing sweep using floating-point numbers is problematic because rounding errors are propagated at higher order from front to back along the viewing ray, resulting in unreliable coefficients especially for higher  $t$ .

To illustrate the issue, consider a piecewise polynomial function modeling the contribution of a small number of particles. Clearly, the last piece polynomial of this approximation should be the zero polynomial. However at its starting knot, its value is most probably not computed to be zero but some value close to zero, due to rounding errors in the computation. Although this zeroth-order distortion may be negligible by itself, small rounding errors in higher-order terms cause large errors further down the ray.

We may thus find the polynomial's value to have grown far from zero for larger  $t$ .

The direct effect of these errors is an unstable result: When using a transfer function with focus on lower attribute values typically reached just before leaving the volumes of influence of the last contributing particles, the resulting pixel colors are highly unstable and the generated images show strong "sprinkling" artifacts, as shown in Figure 2 (a). Hence, solving this problem is indispensable if we want our higher-order ray casting concept to be of any use.

There are several conceivable measures for alleviation. One is to limit the maximum distance on the ray that the error may use to grow by introducing a number of special reinitialization knots at predetermined  $t$  on all rays, as performed for the rendering of Figure 2 (b). When processing a particle during the first sweep, in addition to the knots encoding its higher-order approximation as difference coefficients, we also add to all covered reinitialization knots the localized coefficients of this particle's contributions. Later when processing the sorted sequence of knots, whenever we encounter a reinitialization knot, we directly take the coefficients attached to it instead of computing the coefficients following the update rule (2), thus eliminating the effect of past errors for future pieces. However, this approach not only introduces the complexity of two different kinds of knots but also can only partially solve the problem. Besides, defining how many reinitialization knots to utilize is non-trivial.

Clearly, an alternative would be to avoid the cause of higher-order error propagation altogether and proceed to a direct, possibly localized, coefficient representation of the polynomials, considering for the computation of each result piece only the contributions with overlapping support. While this would most probably facilitate stable results, it would mean accepting the expense of determining for each piece the set of relevant contributions.

### 4.2 Exact Arithmetic Through Quantization

We propose yet another approach to fully avoid rounding errors during the computation of (2), namely by transferring all involved quantities from floating-point to fixed-point numbers, which allow an exact arithmetic. More precisely, we slightly shift all numbers encoding the individual contribution approximations into integer multiples of some quantum values, a process we call quantization. While this increases the approximation errors for the individual contributions, the update operations in (2) are reduced to exact

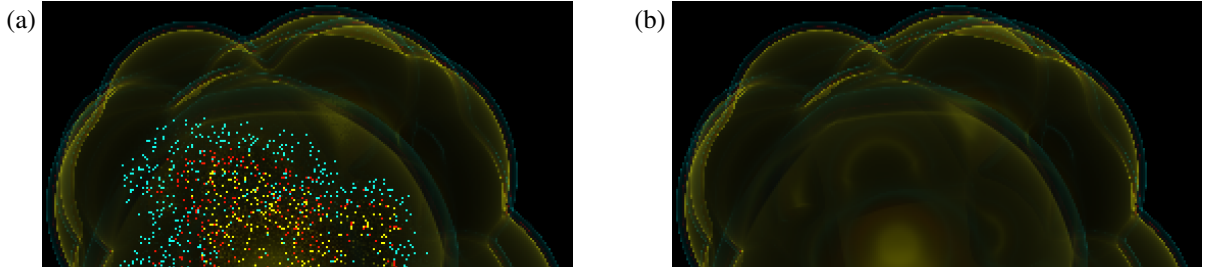


Figure 2: Extracts of sample renderings of the temperature field of an SPH data set using our higher-order SPH field approximation scheme and a transfer function emphasizing three rather low temperature value regions by mapping them to an emission of blue, yellow, and red light. All computations are performed in single precision on the GPU.

(a) clearly shows “sprinkling” artifacts caused by higher-order rounding error propagation, which “randomly” cause the field approximation on the ray to stay within one of the highlighted temperature regions far “behind” the particle cluster. (b) shows the result of employing 10 slices of reinitialization to mitigate the problem.

integer manipulations.

Adhering to the notation used in (2), the values to be quantized are the elements of the knots  $(\bar{t}_k, \hat{a}_{k0}, \dots, \hat{a}_{kD})$  and the localized coefficients  $a_{kd}$ . We thus seek to fix real quantum values  $\tau > 0$  and  $\zeta_d > 0$  for  $d = 0, \dots, D$  such that  $t_k \approx \bar{t}_k \tau$  for some  $\bar{t}_k \in \mathbb{N}$  and  $\hat{a}_{kd} \approx \bar{a}_{kd} \zeta_d$ ,  $a_{kd} \approx \bar{a}_{kd} \zeta_d$  for integers  $\bar{a}_{kd}$  and  $\bar{a}_{kd}$ . The quantized knots can then be encoded as the integer components  $(\bar{t}_k, \bar{a}_{k0}, \dots, \bar{a}_{kD})$ .

Written in these terms, (2) becomes

$$\bar{a}_{kd} = \bar{a}_{kd} + \sum_{j=d}^D \binom{j}{d} \frac{\zeta_j \tau^{j-d}}{\zeta_d} \bar{a}_{(k-1)j} (\bar{t}_k - \bar{t}_{k-1})^{j-d},$$

which we have to guarantee to result in an integer for arbitrary integer values of previously computed  $(\bar{t}_k - \bar{t}_{k-1})$  and  $\bar{a}_{(k-1)j}$ . This can only be accomplished by requiring  $\binom{j}{d} \frac{\zeta_j \tau^{j-d}}{\zeta_d} \in \mathbb{Z}$  for all  $d$  and  $j \geq d$ , which we can fulfill by simply setting  $\zeta_d = \frac{\zeta}{\tau^d}$  for all  $d$ , where we abbreviate  $\zeta = \zeta_0$ . This leaves us with only two quantum values: one length quantum  $\tau$  and one field value quantum  $\zeta$ . We show our strategy of choosing the two in Section 4.4.

However, we first specify in Section 4.3 how we set up the quantized knot positions  $\bar{t}_k$  and difference coefficients  $\bar{a}_{kd}$ , which later form the input of the coefficients update rule (2), now reduced to the integer-only representation

$$\bar{a}_{kd} = \bar{a}_{kd} + \sum_{j=d}^D \binom{j}{d} \bar{a}_{(k-1)j} (\bar{t}_k - \bar{t}_{k-1})^{j-d}. \quad (5)$$

In Section 4.3, we specify how we compute quantized knots from the particle data and the optimal unit particle approximations in the look-up table while in Section 4.4 we explain how we find appropriate quantum values  $\zeta$  and  $\tau$ . Detailed proofs and derivations can be found in the appendix to our preprint (Fischer et al., 2024) hosted on arxiv.org.

### 4.3 Setting Quantized Knots

In order to not introduce higher-order errors through the back door after all, we have to ensure that the input to the integer computations does not contain such errors already. Each individual particle approximation has to have compact support, i. e., its knots have to exactly neutralize each other. This guides us to compute the knots of single particle contributions according to the following rules (derivation in (Fischer et al., 2024)).

Given a particle with attributes  $\mu$ ,  $\rho$ ,  $\zeta$ , and  $\alpha$ , positioned at distance  $\Lambda \zeta$  from the ray with closest point parameter  $t_\chi$ , we select from the look-up table the optimal normalized positive knot positions  $\theta_k$  and localized difference coefficients  $\hat{s}_{kd}$  corresponding to the normalized distance closest to  $\Lambda$ , all in floating-point representation.

We then set the knot position quantum counts to

$$\bar{t}_0 = \left\lceil \frac{t_\chi}{\tau} \right\rceil \text{ and } \bar{t}_k = \bar{t}_0 + \left\lceil \frac{\zeta \theta_k}{\tau} \right\rceil, \bar{t}_{-k} = 2\bar{t}_0 - \bar{t}_k$$

for  $k = 1, \dots, \lceil K/2 \rceil$ , where we use  $\lceil \cdot \rceil$ , a notation inspired by (Hastad et al., 1989, page 860), to refer to just a usual nearest integer rounding function. In this context it is irrelevant whether we round  $z + \frac{1}{2}$  to  $z$  or to  $z + 1$  for  $z \in \mathbb{Z}$ .

Then, for  $k \geq 1$ , except for  $k = 1$  and uneven  $d$  if  $K$  is uneven, the quantized localized difference coefficients are computed as

$$\bar{a}_{kd} = \left\lceil \frac{\tau^d \mu \alpha \hat{s}_{kd}}{\zeta \rho \zeta^{d+3}} \right\rceil \text{ and } \bar{a}_{-kd} = (-1)^{d+1} \bar{a}_{kd}.$$

In case of  $K$  being even, there is a middle knot with possibly non-zero coefficients

$$\bar{a}_{0d} = -2 \sum_{k=-\lceil K/2 \rceil}^{-1} \sum_{j=d}^D \binom{j}{d} \bar{a}_{kj} (\bar{t}_0 - \bar{t}_k)^{j-d}$$

for uneven  $d$ . Otherwise, there is no middle knot, i. e.,  $\tilde{a}_{0d} = 0$  for all  $d$ , but we set

$$\tilde{a}_{-1d} = - \sum_{k=-\lceil K/2 \rceil}^{-2} \tilde{a}_{kd} - \sum_{j=d+1}^D \binom{j}{d} \sum_{k=-\lceil K/2 \rceil}^{-1} \tilde{a}_{kj} (\bar{t}_0 - \bar{t}_k)^{j-d}$$

and  $\tilde{a}_{1d} = (-1)^{d+1} \tilde{a}_{-1d}$  in descending order of uneven  $d$ .

#### 4.4 Specifying Quantum Values

Computing optimal quantum values requires the definition of a manageable error measure for minimization. In our attempts to measure the changes to the field approximation originating from quantization, we have developed the data-independent and ray-independent relative quantization error estimate (derivation in (Fischer et al., 2024))

$$Q_D(\tau, \zeta) = \frac{1}{4\kappa} \sqrt{\kappa'^2 \tau^2 + \sum_{d=0}^D \frac{2q^{2d+3}}{(2d+1)(2d+3)} \cdot \frac{\zeta^2}{\tau^{2d}}},$$

where  $q$  is the upper bound of the kernel function's support and we have abbreviated the constants

$$\kappa = \left( 4\pi \int_{t=0}^{\infty} [t w(t)]^2 dt \right)^{\frac{1}{2}} \quad \text{and} \\ \kappa' = \left( \int_0^q \Lambda \int_{\mathbb{R}} \left[ \frac{d}{dt} B_{\Lambda}(t) \right]^2 dt d\Lambda \right)^{\frac{1}{2}},$$

which only depend on the SPH kernel.

$Q_D$  clearly grows with  $\zeta$ , which is reasonable because the smaller we set  $\zeta$ , the closer the quantized approximations get to the optimal ones. However, smaller  $\zeta$  require larger integer values  $\tilde{a}_{kd}$  and  $\tilde{a}_{-kd}$ . Hence, to guard against integer overflow, we propose to set it to the optimal lower bound

$$\zeta = \frac{a_{\max}}{\text{INT\_MAX}},$$

where INT\_MAX is the maximum representable integer for an integer bit-length yet to be chosen, and  $a_{\max}$  an overall upper bound of the values expected to occur in the approximation, which can be generated by a short analysis of the data set to be visualized.

The situation is not as straight-forward for the length quantum  $\tau$ . On the one hand, large  $\tau$  result in large quantization errors by distorted knot positions. On the other hand, small  $\tau$  mean large higher-order quantum values  $\zeta_d$ , as we have seen in Section 4.2. However, given  $\zeta$ , it is easy to show that  $Q_D(\tau, \zeta)$  is a convex function with respect to  $\tau$ , whose minimizer

can be found by common root-finding methods (proof in (Fischer et al., 2024)).

We have to note, though, while  $Q_D(\tau, \zeta)$  estimates the relative quantization error of a normalized particle, the quantization error for a particle from the data set is better represented by

$$Q_D \left( \frac{1}{\zeta} \tau, \frac{\rho \zeta^3}{\mu \alpha} \zeta \right),$$

i. e., it depends on the particle attributes. Thus, for the purpose of determining a suitable global  $\tau$ , we first specify “representative” attributes  $\mu_{\text{repr}}$ ,  $\rho_{\text{repr}}$ ,  $\zeta_{\text{repr}}$ , and  $\alpha_{\text{repr}}$  of the data set to be visualized. These could be, for example, average or mean values or the attributes chosen from any particle in a region of interest. Afterwards, abbreviating  $\phi_{\text{repr}} = \frac{\mu_{\text{repr}} \alpha_{\text{repr}}}{\rho_{\text{repr}} \zeta_{\text{repr}}^3}$ , we set  $\tau$  to minimize  $Q_D \left( \frac{\tau}{\zeta_{\text{repr}}}, \frac{\zeta}{\phi_{\text{repr}}} \right)$ .

## 5 CHOOSING APPROXIMATION DIMENSIONS

Having set forth our quantized higher-order approximation field concept, the question remains how to choose its most fundamental parameters: the approximation order  $D$ , the number of non-trivial polynomial pieces per particle  $K$ , and the bit length defining the maximum representable integer INT\_MAX. All three parameters can significantly impact both approximation accuracy and performance. While the quality of any configuration  $(D, K, \text{INT\_MAX})$  ultimately requires thorough testing to be readily evaluated, we do want to provide an overview in theory here.

### 5.1 Performance Implications

The method's asymptotic complexity with respect to  $K$  and  $D$  is easily seen:  $K$  has linear impact on the number of knots per ray and therefore a linearithmic one on the overall process time due to the search sweep. The overall time effect of  $D$  is quadratic since both the number of coefficients to be updated and the number of operations for each coefficient update according to (5) grow linearly with  $D$ .

The performance implications of the integer bit length require special attention. In spite of the focus on floating-point performance, modern GPUs intrinsically support calculations on integer types of 16-bit and 32-bit lengths. For 32-bit integers, the throughput of additions is comparable to 32-bit floating-point operations while multiplications are commonly processed about five times slower. 64-bit integer operations are formally supported in some GPU program-

ming contexts (Vulkan API, OpenCL, CUDA, extension of GLSL) but seem to be always emulated in software based on 32-bit operations.

Such an emulation can easily be constructed for integer types of any size, which means that in theory we can go with arbitrarily small quantum values, albeit at a high price performance-wise. To get an impression about the performance implications of using such long integers, we have analysed such algorithms. Compared to 32-bit integers, our results show a cost increase by a factor of roughly 9 for bit-length 64, 24 for 96 bit, and 45 for 128 bit, although we expect 64-bit integers to be especially well-optimized in the implementations by GPU vendors.

To be able to decide whether these cost factors are worthwhile and, consequently, choose adequate quantum values, we have to form an idea of the quality implications of smaller or larger integer sizes. In other words, we have to relate the quantization cost to the quantization error.

## 5.2 Combined Accuracy Measure

In order to estimate the overall accuracy implications of  $D$ ,  $K$ , and INT\_MAX, we combine the error estimate for piecewise polynomial approximation and the one for quantization.

In Section 3.2, we have already defined the error  $E_\Lambda(S)$  for a non-quantized approximation along a ray with normalized distance  $\Lambda$ . To become independent from  $\Lambda$  and the screen resolution, we integrate the optimal per-ray error over all viewing rays in one direction. We then divide it by the  $L^2$  norm  $\kappa$  of a normalized particle's contribution to arrive at the per-particle relative error for higher-order approximation

$$E_{K,D} = \frac{1}{\kappa} \left( 2\pi \int_0^q \Lambda E_\Lambda^2(S_\Lambda) d\Lambda \right)^{\frac{1}{2}},$$

where we have used  $S_\Lambda$  to refer to the optimal approximation at distance  $\Lambda$  with optimal knot positions. As we are given these approximations only implicitly through a minimization process, we compute  $E_{K,D}$  only approximately as a Riemann sum. It constitutes a precise relative approximation error measure for any particle, not just for the normalized ones.

By contrast, the quantization error is difficult to quantify exactly, due to the statistical and rather complex distortion caused by quantization. We thus content ourselves with the rather rough estimate  $Q_D \left( \frac{\tau}{\zeta_{\text{repr}}}, \frac{\varsigma}{\phi_{\text{repr}}} \right)$  defined in Section 4.4. While this measure is already in the form of a per-particle relative error derived from an all-rays integration similar

to the one for  $E_{K,D}$  above, it has the disadvantage of apparently depending on  $\tau$ ,  $\zeta$ , and the particle data. Fortunately, a closer look reveals that the only property of the data set that  $Q_D$  really depends on is

$$\frac{a_{\text{max}}}{\phi_{\text{repr}}},$$

i. e., the ratio between a “representative” particle's factor to the SPH kernel and an upper bound of the target field value. It is a measure for the variance of the target scalar field. Fixing this value at, say,  $10^5$  to be robust against integer overflow for at least some level of data variance, we can compute  $\frac{\varsigma}{\phi_{\text{repr}}}$  and thus the optimal ratio  $\frac{\tau}{\zeta_{\text{repr}}}$  providing the error  $Q_D$ .

Committed on a quantization error estimate  $Q_D$ , we can combine it to  $E_{K,D}$  above to form an overall approximation accuracy measure. Simply summing the two errors would introduce an overestimation bias as it would model all distortions acting in the same direction. Instead, we treat the two sources of error as if they were perpendicular and take the  $L^2$  norm of their sum, arriving at the overall approximation error

$$\sqrt{E_{K,D}^2 + Q_D^2}$$

for evaluating configurations  $(D, K, \text{INT\_MAX})$ .

The overall error clearly falls with growing  $K$  and INT\_MAX as these two parameters only effect one of the two components. However, the effect of  $D$  is more interesting because higher  $D$  cause  $E_{K,D}$  to fall but  $Q_D$  to grow. Thus, for any fixed  $K$  and INT\_MAX, there is one error-minimizing  $D$ , such that raising the approximation order further will not be worthwhile as it will reduce the accuracy at even higher cost. Figure 3 shows a plot of  $E_{K,D}$  and  $Q_D$ , as well as the combined error, for the cubic B-spline kernel (1). It covers values for approximation order  $D$  up to 6 and per-particle non-trivial pieces count  $K$  up to 4, assuming an integer bit length of 64 and a data variance ratio of  $a_{\text{max}}/\phi_{\text{repr}} = 10^5$ . One can see that for, e. g.,  $K = 3$  and  $K = 4$ , raising  $D$  above 4 is clearly not worthwhile.

While it is hard to select one universal configuration from the analysis presented here, a visualization tool using our method can implement a single one or several, and even more than one SPH kernel function. For each configuration,  $E_{K,D}$  can be computed at compile time. Also  $Q_D$ , being a one-dimensional function of  $a_{\text{max}}/\phi_{\text{repr}}$  for each  $D$  and INT\_MAX after all, can quickly be obtained from a look-up table, facilitating an efficient computation of the combined error estimate when loading a data set. Thus, although choosing an ideal configuration depends on user preferences for balancing quality and speed, a visualization tool may restrict a set of implemented configura-



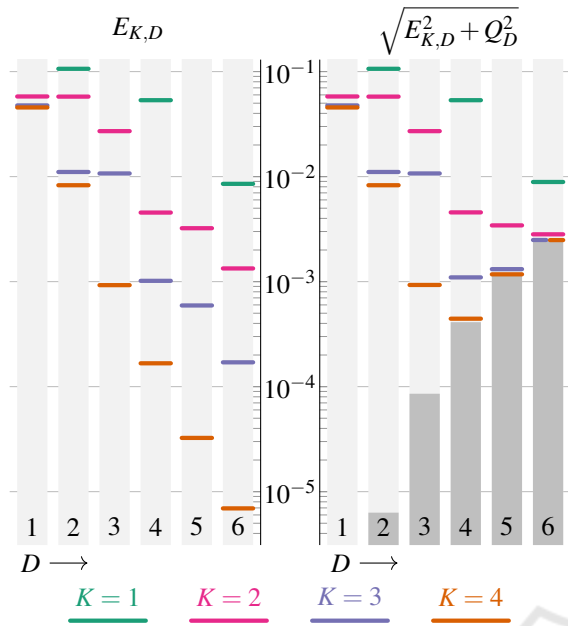


Figure 3: Plot of error values in the example case of the cubic B-Spline SPH kernel (1), for  $D \leq 6$  and  $K \leq 4$ . The horizontal colored marks show the polynomial approximation error component  $E_{K,D}$  on the left and the combined error  $\sqrt{E_{K,D}^2 + Q_D^2}$  on the right. The grey bars on the right-hand side depict the quantization error component  $Q_D$ , which has been computed assuming an integer bit length of 64 and a data variance factor  $a_{\max}/\phi_{\text{repr}} = 10^5$ .

tions to a preselection of worthwhile configurations in view of the target data set for the user to choose from.

## 6 CONCLUSION

Seeking a new level of quantitative accuracy in scientific visualization, we have presented a novel approach to approximating SPH scalar fields on viewing rays during volume ray casting. It features a locally adaptive spatial resolution and efficient summation scheme. We have shown how to efficiently compute the best possible higher-order approximations of particle contributions of any given order and resolution, and provided a thorough theoretic analysis of the approximation errors involved. Conveying these error estimates to the user, could meet a field expert's need for quantitatively assessing the errors involved in the visualization process.

While we have confined our explanations on representing field values, the procedure for gradients or other field features is analogous, as long as these are additively generated from particle contributions. Also, despite our focus on SPH data, our concepts

may very well be applicable to direct volume renderings of scattered data.

Clearly, our findings have yet to prove their competitiveness in practise. However, we are confident that they will help in advancing scientific visualization of scattered data in terms of quantitative accuracy.

## REFERENCES

- Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. In *ACM Siggraph Computer Graphics*, volume 22.4, pages 65–74. ACM.
- Fischer, J., Schulze, M., Rosenthal, P., and Linsen, L. (2024). Particle-wise higher-order SPH field approximation for DVR. arXiv:2401.02896.
- Fraedrich, R., Auer, S., and Westermann, R. (2010). Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1533–1540.
- Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389.
- Hastad, J., Just, B., Lagarias, J. C., and Schnorr, C.-P. (1989). Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal on Computing*, 18(5):859–881.
- Hochstetter, H., Orthmann, J., and Kolb, A. (2016). Adaptive sampling for on-the-fly ray casting of particle-based fluids. In *Proceedings of High Performance Graphics*, pages 129–138. The Eurographics Association.
- Kähler, R., Abel, T., and Hege, H.-C. (2007). Simultaneous gpu-assisted raycasting of unstructured point sets and volumetric grid data. In *Proceedings of the Sixth Eurographics/IEEE VGTC conference on Volume Graphics*, pages 49–56. Eurographics Association.
- Knoll, A., Wald, I., Navratil, P., Bowen, A., Reda, K., Papka, M. E., and Gaither, K. (2014). Rbf volume ray casting on multicore and manycore cpus. In *Computer Graphics Forum*, volume 33.3, pages 71–80. Wiley Online Library.
- Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024.
- Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108.
- Oliveira, F. and Paiva, A. (2022). Narrow-band screen-space fluid rendering. In *Computer Graphics Forum*, volume 41.6, pages 82–93. Wiley Online Library.
- Rosswog, S. (2009). Astrophysical smooth particle hydrodynamics. *New Astronomy Reviews*, 53(4-6):78–104.