Higher Order Algebraic Signed Distance Fields

Gábor Valasek[1] , Róbert Bán[2]

[1]Eötvös Loránd University, Hungary, valasek@inf.elte.hu
[2]Eötvös Loránd University, Hungary, rob.ban@inf.elte.hu

Corresponding author: Gábor Valasek, valasek@inf.elte.hu

Abstract. This paper investigates a higher order generalization of signed distance fields in a high performance context. The data within these constructs approximate the distance function over a larger area. Algebraically, a straightforward realization of this is derived from formulating the general sampling and reconstruction process as a function approximation problem that combines local Taylor approximations. The degree of these approximations is referred to as the order of the field. By noting that the error bound of a combination of Taylor approximations is closed under convex barycentric combination, we derive a characterization of accuracy for arbitrary sampling topologies and interpolation schemes. We extend this discussion to least squares fit polynomial fields. In case of regular grids, we show that GPU filtering can be utilized to implement highly efficient queries on both types of fields. Additionally, we investigate the direct storage of partial derivatives in combination with Hermite interpolation. Although the latter type of fields are more expensive in terms of query times, they provide higher accuracy.

We evaluate the performance-storage trade-off of these generalizations and show various optimization techniques to balance it. We propose a general framework that converts traditional, zero order signed distance fields to higher order representations. We present an empirical analysis of the first, second, and third order constructs in terms of storage and query efficiency, and propose optimizations in both regards, such as decreasing filtering footprint size via simplical interpolation schemes. Finally, we show how higher order filtering techniques can leverage the presence of higher order derivative data in the samples to improve quality without increasing the filtering footprint, and how higher order geometric properties can be inferred from these data, such as surface normals.

Keywords: Surface and Volume Representation, Implicit Representation, Geometric Modeling, Collision Detection
DOI: https://doi.org/10.14733/cadaps.2023.1005-1028

1 INTRODUCTION

Signed distance functions (SDFs) form a special subset of implicit functions. For each $\boldsymbol{x} \in \mathbb{E}^n$ point in space, an $f : \mathbb{E}^n \to \mathbb{R}$ SDF maps the signed distance of the argument $\boldsymbol{x}$ to the $\{\boldsymbol{y} \in \mathbb{E}^n \mid f(\boldsymbol{y}) = 0\}$ zero level-set. As such, they go beyond merely encoding volumes, they also convey global geometric information about the scene. This versatility makes SDFs ubiquitous in various venues of computing. However, this power comes at the cost of representation, as a closed-form formulation of the exact SDF is infeasible for all but the simplest geometries and configurations. In practice, we have to rely on approximate SDFs.

In high performance real-time applications, the most common approximation is a collection of SDF samples combined with an interpolation method. The latter is used to infer a continuous approximation to the SDF from the samples. We refer to the combination of data and interpolation as a discrete signed distance field (DSDF). Usually, the samples are arranged in a two or three dimensional regular grid [21, 42] or stored in a more adaptive spatial subdivison structure, such as octrees [17, 25].

We propose a theoretical generalization of discrete signed distance fields to higher orders via the samples and investigate three particular realizations of our formulation. This requires us to first establish our interpretation of a single sample. Due to the domain of application, one can either treat it algebraically, i.e. as an isolated function sample, or geometrically, as a volume.

The most prominent application of the latter is Hart's paper on sphere tracing [22]. He used the SDF values to form unbounding spheres around sample positions. These are open volumes that do not contain any boundary points which facilitates robust and efficient empty space skipping during ray tracing. This is the most widely used visualization algorithm for rendering SDFs and their approximations, it illustrated in Figure 1. Recently, this approach was generalized to quadrics of revolution in the form of quadric tracing [5], as they offer a richer vocabulary to express the inside/outside partitioning of space induced by the shape. Unfortunately, inferring a continuously differentiable approximation to the SDF from a set of such samples is difficult as it is not evident how to combine geometric unbounding entities in a correct and nontrivial way, a topic not yet addressed in the literature. Traditionally, sphere tracing simply takes a linear combination of the centers and radii of the unbounding spheres of the enclosing cube of samples.

In this regard, treating signed distance values purely algebraically, that is, as function samples simplifies the reconstruction of smooth SDFs. We can apply any filtering to them from function approximation and interpolation theory. However, this also means that there are no guarantees that an arbitrary combination of SDF values yields a geometrically valid distance everywhere, e.g. a valid unbounding sphere. When invalid unbounding spheres are inferred for points in space during sphere tracing, they are identified as view-dependent visual artifacts, often referred to as flickering or floating geometries. These are usually heuristically mitigated by choosing appropriate step size multipliers, an art form on its own right [32, 33].

Our paper focuses on the algebraic interpretation of an SDF sample. This choice also implies that the majority of our discussion is independent of SDFs, it can be applied to the approximation of arbitrary functions. In the constructs we propose, a higher order sample expresses the behaviour of the source function in an area around the sample position.

A straightforward realization of such a descriptor is the Taylor expansion of the target function. In Section 3.1, we show that the error bound of a convex barycentric combination of Taylor approximations is within the convex hull of the per-sample error bounds. By choosing an appropriate polynomial basis, we also show how GPU hardware filtering can be used to accelerate inference.

A higher order approximation over an area may be also achieved by polynomial regression. In our case, we show that least squares fitting to a fine grid of SDF values yields a higher order sample. In Section 3.2, we present an efficient GPU implementation of such a fitting framework. It can be applied to any representation for which point-surface distance queries are defined.

We investigate Hermite interpolation in Section 3.3 for higher order field construction and propose a particular Ferguson-like [44] solution to the underlying underdetermined interpolation problem. We show that

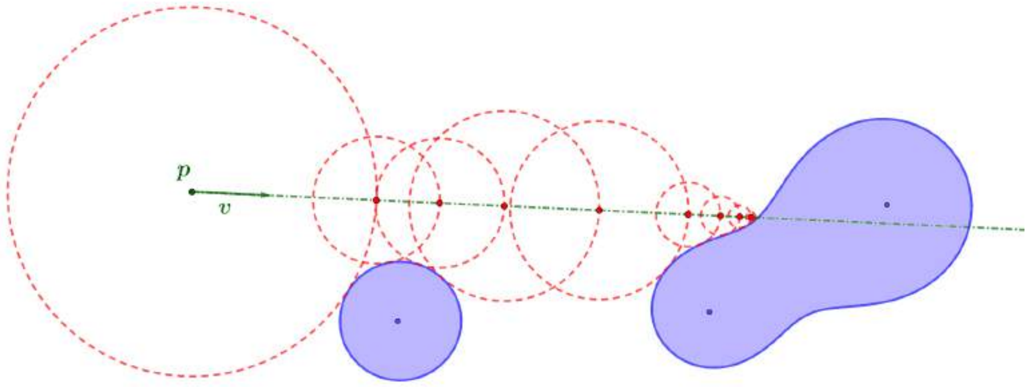


Figure 1: Sphere tracing computes the intersection between a $\boldsymbol{p}(t) = \boldsymbol{p} + t\boldsymbol{v}$, $t \geq 0$ ray (dashed green half line) and the scene (purple blended circles). At each iteration, sphere tracing evaluates the SDF and takes a step equal to the radius of the current unbounding sphere (dashed red circles). The trace terminates if either the ray enters a volume, i.e. the SDF is negative, or if the ray is very close to a boundary. Entering a volume is possible because the interpolation of SDF values may not yield a correct unbounding sphere.

these provide superior accuracy and efficient storage at the expense of greatly increased evaluation times.

We also address the issue of sample representation and interpolation in Section 4. We propose filters that preserve the higher order data at the samples based on Hermite interpolation that are akin to geometrically continuous blending of implicit surfaces [23].

The overall goal of this paper is to present high performance, GPU-friendly higher order DSDF representations. As such, we mainly consider regular grids of samples, however, we do discuss the advantages and drawbacks of combining samples from enclosing tetrahedrons instead of cubes within these regular structures and also briefly compare our constructs to non-regular grids. Additionally, we enumerate several optimization techniques in Section 6 to speed up the query of these fields on the GPU in general, and their rendering in particular. Finally, we present our experimental results in Section 7.

2 PREVIOUS WORK

Signed distance functions have applications in a broad range of fields, from geometric modeling through computer graphics to physics [1, 21, 42, 15, 4, 28, 18, 41, 11]. Recently, deep learning applications have also started utilizing SDFs [29], or their generalizations such as medial fields [34].

Sampled signed distance functions have been used with great efficiency in high quality real time font rendering [21]. For three dimensional applications, an attractive property of a distance field is that it can resolve multiple ray-surface intersection queries via cone-tracing [22]. This was used in [42] and [8] to approximate soft shadows from spherical light sources more accurately. Recently, distance fields have been shown as a viable primary geometric representation for real time games [2]. Storage improvements were also studied in the literature via hierarchical space partitioning [17] and the idea of incorporating gradient data into the samples has been presented as well [13, 9]. Our main contribution in this regard is noting that higher order samples allow us to reduce total distance field memory storage even on regular grids.

Using higher order polynomials for DSDF representation has been proposed by Koschier et al. [25] in the context of collision detection. They chose shifted normalized Legendre polynomials as their polynomial basis so that fitting can be implemented via projections onto the basis functions. This, combined with an adaptive spatial subdivision strategy such as in [17], resulted in a compact representation of shapes. Unfortunately, high degree polynomials require hundreds of coefficients which is not suitable for real time rendering. Moreover, although their fitting strategy proved to be sufficient for collision detection, they have undesirable visual discontinuities along cell boundaries in rendering, even with high polynomial degrees. Changing their fitting to LSQ resolved this in our experiments. In this sense, their approach can be considered as a nearest neighbor filtered special case of our LSQ constructs but they offer better spatial adaptability.

A similar spatial subdivision was used by Song et al. in [38] to construct hierarchical polynomial splines over T-meshes with a C^1 Hermite interpolation at the vertices. We propose a similar solution to this problem in our Ferguson-Hermite formulation over regular grids and generalize it to second order.

Pottmann and Hofer used quadratic approximations to the squared distance function in [31]. Discarding the sign as such makes set theoretic operations inapplicable except for unions, that are otherwise elegantly handled by implicit representations [43, 35, 20]. Pottmann and Hofer's construction assumes a continuous representation of the underlying geometry and in practice uses the squared distance from the tangent plane at the footpoint as means of approximation. Our least squares approach differs in that it can be generalized to higher orders, generates a signed distance field, and can be applied to any representation where point-boundary signed distance queries can be carried out.

In real time computer graphics, filtering distance fields is usually restricted to nearest neighbor and trilinear filtering. In volume rendering, there are several higher order filtering schemes [14] showing that carefully selected and combined trilinearly filtered samples can be used to achieve a higher quality. The drawback of this and related [37] higher order filtering approaches is the increased filtering footprint, i.e. they require more samples per query. We propose a higher order filtering technique, similar to blending in geometric modeling [23], that does not fetch extra samples, instead, it utilizes the higher order information available in the generalized Taylor samples. These results hold for arbitrary $f : \mathbb{R}^n \to \mathbb{R}$ functions. We also show that a Ferguson-type solution to the resulting Hermite interpolation problem produces superior error metrics and visuals for first and second order fields.

Hart showed that sphere tracing provides robust means to the high quality rendering of signed distance lower bounds [22]. There have been several advances in making sphere tracing more efficient in how it takes steps [24, 6] or how sharper bounds can be achieved [19], and how other unbounding geometries [5] or proxy shapes may be used. We show that higher order samples can contribute to performance by replacing finite difference methods for surface normal computation with the exact gradient of the interpolated higher order polynomial approximations, making certain fields not only more performant but also more memory efficient.

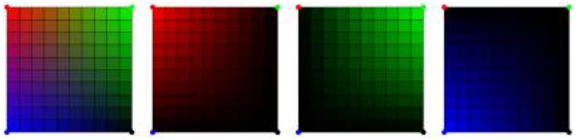
The motivation for the investigation of Taylor DSDFs is their maximum error bounds that can be used to quantify the accuracy of a particular reconstruction from a set of samples. Most other work in this topic relies on a frequency-based argument for accuracy analysis [12], but other approaches, e.g. based on an initial polygonization of the represented geometry and R-functions [30, 36] have been also presented. We prove that Taylor's theorem provides a convenient, local characterization of accuracy that does not require any fixed sampling topology or homogeneous sample order.

A first and second order geometric generalization of DSDFs to planar curves can be found in [10]. Tangent lines and osculating circles were used as first and second order sample data and interpolation was carried out in the range, that is, by using the barycentric combination of the signed distances taken separately from the samples. Our paper provides an algebraic generalization to this that works in arbitrary dimensions and orders.

3 HIGHER ORDER SAMPLE FIELDS

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that is sampled at an $\boldsymbol{x_i} \in \mathbb{R}^n$ set of points, $\boldsymbol{i}$ ranging over an appropriate set of multi-indices; please refer to Appendix A for a short summary on multi-index notation. A **sample field** refers to a $\{\boldsymbol{s_i} = (\boldsymbol{x_i}, \boldsymbol{d_i})\}$ set of position and data pairs.

The data part may consist of simple sample values, i.e. $\boldsymbol{d_i} = f(\boldsymbol{x_i})$, or mappings defined over $\mathbb{R}^n$ itself, such as polynomials, or data that can be used to construct a $f_{\boldsymbol{i}}(\boldsymbol{x})$ mapping that approximates f in


(a) Bilinear interpolation over the rectangle.

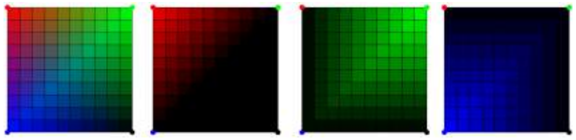

(b) Barycentric interpolation over two triangles.

Figure 2: Comparison between tensor product and simplical linear interpolation. The input colors are denoted by the red, green, blue, and black circles on the vertices. The rectangle is decomposed into the top-left and the bottom-right pair of triangles on the right. Note the more pronounced diagonal on the right and the absence of the fourth interpolated color.

the neighborhood of $\boldsymbol{x_i}$, potentially using additional data from other samples. In general, a field query is decomposed into the following steps:

1. Fetch all $\boldsymbol{s_i},\ \boldsymbol{i} \in \mathcal{J}$ samples under the filtering footprint
2. Create a $f_{\boldsymbol{i}} : \mathbb{R}^n \to \mathbb{R}$ inference mapping from $\boldsymbol{d_i},\ \boldsymbol{i} \in \mathcal{J}$
3. Evaluate $f_{\boldsymbol{i}}(\boldsymbol{x})$, i.e. the inference at the query position

Step 1 determines the memory bandwidth of the query. We investigate two types of footprints: that of the smallest enclosing cube and tetrahedron of samples. In $\mathbb{R}^n$, the former requires 2^n samples, whereas the latter only fetches $n+1$.

In practice, Step 2 and 3 are usually carried out simultaneously, for example using bi- or trilinear interpolation. We consider two different strategies to create inference mappings: blending of higher order polynomials and per-cell Hermite interpolation.

3.1 Taylor Polynomial Fields

The degree k multivariate Taylor approximation of an $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^{k+1}$ function about $\boldsymbol{x}_0$ in multi-index notation is

$$T_{k,\boldsymbol{x}_0}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| \le k} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_0)}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} \ . \tag{1}$$

Sample field queries usually aggregate several samples to construct an approximation to the sampled function. It can be easily shown that the error of a convex barycentric combination of Taylor approximations can be bounded by the largest individual local error of the members of the combination.

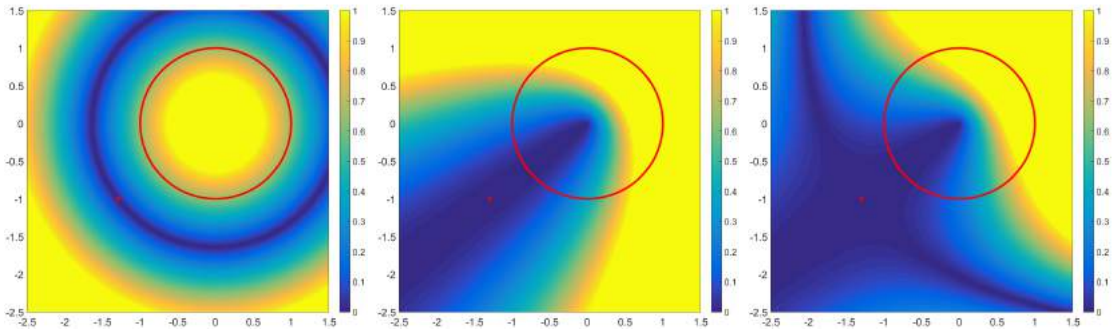
Indeed, given an arbitrary $\lambda_{\boldsymbol{i}}(\boldsymbol{x})$ collection of barycentric weight functions, i.e. $\lambda_{\boldsymbol{i}} : \mathbb{R}^n \to \mathbb{R}$ such that $\forall \boldsymbol{x} \in \mathbb{R}^n : \sum_{\boldsymbol{i}} \lambda_{\boldsymbol{i}}(\boldsymbol{x}) = 1$, we have

$$\left| f(\boldsymbol{x}) - \sum \lambda_{\boldsymbol{i}}(\boldsymbol{x}) T_{\boldsymbol{i}}(\boldsymbol{x}) \right| = \left| \sum \lambda_{\boldsymbol{i}}(\boldsymbol{x}) R_{\boldsymbol{i}}(\boldsymbol{x}) \right| \le \sum |\lambda_{\boldsymbol{i}}(\boldsymbol{x})| \cdot |R_{\boldsymbol{i}}(\boldsymbol{x})| \ ,$$

where $R_{\boldsymbol{i}}(\boldsymbol{x})$ are the Lagrange error terms about sample positions $\boldsymbol{x_i}$. If the query position $\boldsymbol{x}$ is within the convex hull of the $\boldsymbol{x_i}$ collection of samples, i.e. $\forall \boldsymbol{i} : \lambda_{\boldsymbol{i}}(\boldsymbol{x}) \ge 0$, then

$$\left| f(\boldsymbol{x}) - \sum \lambda_{\boldsymbol{i}}(\boldsymbol{x}) T_{\boldsymbol{i}}(\boldsymbol{x}) \right| \le \sum \lambda_{\boldsymbol{i}}(\boldsymbol{x}) \cdot |R_{\boldsymbol{i}}(\boldsymbol{x})| \le \max_{\boldsymbol{i}} |R_{\boldsymbol{i}}(\boldsymbol{x})| \ . \tag{2}$$

This shows that the error bound of the combination is always within the convex hull of the individual errors. This suggests that by carefully selecting barycentric weight functions, we may improve the cumulative error of the combination.



(a) a_{00} (b) $a_{10}x + a_{01}y + a_{00}$ (c) $a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{10}x + a_{01}y + a_{00}$

Figure 3: Error heatmaps of planar signed distance fields extrapolated from single order zero (a), one (b), and two (c) Taylor samples. The samples are represented by the $a_{ij} \in \mathbb{R}$ coefficients shown in the subfigure captions. The sample position is denoted by the red X. The geometry is the origin centered, radius 1 circle (in red). Blue corresponds to low error, yellow is high error.

Moreover, Inequality (2) also demonstrates that the number of samples in the combination does not necessarily improve the precision. Thus, combining the samples of the enclosing simplex ($d + 1$ samples $\mathbb{R}^n$) instead of the enclosing hypercube (2^d samples in $\mathbb{R}^n$) may yield the same, or even better Taylor error bounds as it may rule out potentially far away samples from the combination. However, using enclosing simplices results in more boundaries along which continuity issues may arise. See Figure 2 for a visual comparison between bilinear interpolation of colors over a rectangle versus the barycentric interpolation of the same colors over two triangles constructed from the rectangle.

In theory, the remainder term itself may be used to correct the Taylor approximation of SDFs such that they never infer invalid unbounding sphere radii. This, however, requires the ability to bound the higher order derivatives of the SDF. As an example, let us consider the interpolation of a single SDF sample.

By definition, traditional sample fields contain sample values, i.e. zero order Taylor approximations, since $T_{0,\boldsymbol{x}_0}(\boldsymbol{x}) = f(\boldsymbol{x}_0)$. Following the above argument, the error bound of an order 0 sample is derived as

$$f(\boldsymbol{x}) = f(\boldsymbol{x_i}) + \sum_{|\boldsymbol{\alpha}|=1} \frac{\partial^{\boldsymbol{\alpha}} f(\tilde{\boldsymbol{x}})}{\boldsymbol{\alpha}!}(\boldsymbol{x} - \boldsymbol{x_i})^{\boldsymbol{\alpha}} = f(\boldsymbol{x_i}) + \nabla f(\tilde{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \boldsymbol{x_i})$$

which is arranged as

$$|f(\boldsymbol{x}) - f(\boldsymbol{x_i})| = |\nabla f(\tilde{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \boldsymbol{x_i})| = \|\nabla f(\tilde{\boldsymbol{x}})\|_2 \cdot \|\boldsymbol{x} - \boldsymbol{x_i}\|_2 \cdot |\cos\alpha| \leq \|\boldsymbol{x} - \boldsymbol{x_i}\|_2 \quad (3)$$

since $\|\nabla f(\tilde{\boldsymbol{x}})\|_2 \equiv 1$ for all regular points of the SDF. According to this, the maximum radius of an unbounding sphere at $\boldsymbol{x}$ is $f_i - \|\boldsymbol{x} - \boldsymbol{x_i}\|_2$. Interestingly, this is the exact same step sphere tracing would make, thus unbounding spheres can be derived from the Lagrange form of the remainder as above.

Nevertheless, the actual error may be lower than the one given by Taylor's bound. Figure 3 visualizes the real absolute error of order 0, 1, and 2 Taylor signed distance field samples taken from the origin centered unit circle. Note the expansion of the high precision areas (in blue) around the higher order sample positions.

The Taylor expansion of the SDF to a mesh can be analytically computed. The key observation is that the SDF of a mesh can be decomposed into particular instances of three classes of distance functions: one induced by a vertex, an edge, and a planar face. All three of these have exact SDF realizations, i.e. the distance function of a point, a line segment, and a plane. This demonstrates that even piece-wise and discontinuous geometric representations can be converted to higher order discrete signed distance fields (DSDFs).

If a parametric or implicit representation of the target function is available, the partial derivatives can be computed analytically. If a sample lies on the cut locus of the distance function, the Taylor expansion simply falls back to an order zero sample. For procedural representations, the analytic derivatives may not be readily available. In this case, numerical approximation schemes can be used to compute the necessary derivatives or one could use automatic differentiation. We propose an alternative in the next subsection.

3.2 Least-Squares Polynomial Fields

To overcome the difficulties of obtaining exact analytical or approximate numerical derivatives of the target function, we fit a higher order polynomial $f_{\boldsymbol{i}}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}|\le k} a_{\boldsymbol{i},\boldsymbol{\alpha}} \cdot \boldsymbol{x}^{\boldsymbol{\alpha}}$ to an $f_{\boldsymbol{i},\boldsymbol{j}} = f(\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}})$ collection of function values taken around $\boldsymbol{x}_{\boldsymbol{i}}$.

A bound on the accuracy of an LSQ polynomial approximation may be derived by considering the LSQ polynomial as an approximation to a collection of zero order Taylor samples.

If the $\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}}$ samples are arranged in a regular grid, the spacing between them determines their individual Taylor error bounds. Let S denote the length of the diagonal of the $\boldsymbol{V}(\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}}) \subset \mathbb{R}^3$ Voronoi box around an $\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}}$ sample and $D > 0$ an upper bound on the magnitude of $\nabla f(\boldsymbol{x})$ in $\boldsymbol{V}(\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}})$. Then from (3) we have

$$\forall \boldsymbol{x} \in \boldsymbol{V}(\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}}) \ : \ |f(\boldsymbol{x}) - f_{\boldsymbol{i}}| \le D\frac{S}{2} \ .$$

Let M denote the residual sum of squares. Then the maximum deviation of the $f_{\boldsymbol{i}}(\boldsymbol{x})$ LSQ fit from the samples over a Voronoi cell is $\sqrt{M} + E\frac{S}{2}$, where E is an upper bound on $\|\nabla f_{\boldsymbol{i}}(\boldsymbol{x})\|_2$. The above combined give the bound on the maximum deviation of the LSQ fit from the target function f over the union of the Voronoi cells as

$$\max|f(\boldsymbol{x}) - f_{\boldsymbol{i}}(\boldsymbol{x})| \le \sqrt{M} + \frac{S}{2}(D+E) \ . \tag{4}$$

As the number of $\boldsymbol{x}_{\boldsymbol{i},\boldsymbol{j}}$ samples increases, the Voronoi diagonals shrink. This means that S decreases, however, the increase in the number of samples is more likely to cause the increase of the residual error as well, once it is larger than the number of points that the polynomial can interpolate.

We show a practical, high performance algorithm for the construction of LSQ DSDFs in Section 5.

3.3 Hermite Interpolation Fields

Unlike the previous two local higher order approximations, Hermite interpolants cannot be separated from the interpolation method. However, they also offer better accuracy over an interval at a reduced degree [40].

A tri-cubic tensor product patch possesses $4^3 = 64$ control data. These data cover the entire cube of the enclosing 8 vertices of the query position. An order 1 sample constraints 4 of these control data per vertex, leaving us $64 - 32 = 32$ scalar degrees of freedom.

Ferguson phrased this in terms of partial derivatives at the vertices of a parametric surface and set all the missing partial derivatives to 0 [16]. Doing the same with our data and storing the SDF values and first partial derivatives per vertex results in an Hermite interpolant that first fetches the control data of the 8 vertices ($[f(\boldsymbol{x}_{\boldsymbol{i}}), f_x(\boldsymbol{x}_{\boldsymbol{i}}), f_y(\boldsymbol{x}_{\boldsymbol{i}}), f_z(\boldsymbol{x}_{\boldsymbol{i}})], \boldsymbol{i} \in \{0,1\}^3$), then uses the Hermite basis to reconstruct the approximated function, essentially setting the missing coefficients to zero.

Using the basis functions $\alpha_0(x) = 1 - 3x^2 + 2x^3$, $\alpha_1(x) = 3x^2 - 2x^3$, $\beta_0(x) = x - 2x^2 + x^3$ and

$\beta_1(x) = -x^2 + x^3$, the reconstructed value at $\boldsymbol{x} = [x, y, z]^T \in \mathbb{R}^3$ is

$$\hat{f}(\boldsymbol{x}) = \sum_{\boldsymbol{i}} [f(\boldsymbol{x_i})\alpha_i(x)\alpha_j(y)\alpha_k(z) + f_x(\boldsymbol{x_i})\beta_i(x)\alpha_j(y)\alpha_k(z) + \\ + f_y(\boldsymbol{x_i})\alpha_i(x)\beta_j(y)\alpha_k(z) + f_z(\boldsymbol{x_i})\alpha_i(x)\alpha_j(y)\beta_k(z)] \,.$$

The main drawback of this method is that it cannot be accelerated with hardware interpolation as the Hermite interpolant cannot be expressed as a blending of local polynomials. However, it is straightforward to extend the above method to second order interpolation. The number of unconstrained coefficients grows even more and setting them to zero proved to offer less improvement in accuracy compared to other second order filtering methods than it was in the first order. Due to performance limitations, we did not pursue the third order solution to this problem.

4 FILTERING HIGHER ORDER FIELDS

4.1 Sample Representation

A sample of a Taylor or LSQ higher order algebraic field stores a degree k multivariate polynomial. Filtering is a barycentric combination of the values of these polynomials evaluated at the query position.

There are multiple bases to represent a polynomial. By choosing one where evaluation is invariant under barycentric combination, we can optimize evaluation and leverage GPU hardware filtering.

In the 1D case, the above invariance means that $(1-\lambda(x))f_i(x) + \lambda(x)f_{i+1}(x)$ can be reduced to a single evaluation after interpolating the coefficients of $f_i(x)$ and $f_{i+1}(x)$. It follows immediately that, for example, the global power basis satisfies this property, that is

$$(1-\lambda(x))\sum_j a_{i,j}x^j + \lambda(x)\sum_j b_{i,j}x^j = \sum_j \Big((1-\lambda(x))a_{i,j} + \lambda(x)b_{i,j}\Big)x^j$$

holds. As a result, the evaluation of an arbitrarily filtered higher order sample reduces to the following steps:

1. Fetch the coefficients of all $\boldsymbol{d_i}$ samples in the filtering footprint.
2. Apply higher order filtering to the coefficients of the samples.
3. Evaluate the resulting polynomial at $\boldsymbol{x}$.

This is a considerable performance gain: instead of evaluating all N polynomials under the footprint and weighting the N results by the filter, we filter the coefficients of the N samples under the footprint and evaluate the resulting single polynomial once.

There are various bases for representing polynomials, each with various properties. Any global basis is applicable to our use case, however, power bases are the most efficient for GPU evaluation. In our tests with $[-1, 1]^3$ normalized sample positions they exhibited no numerical issues with 32 bits floating point, i.e. binary32, textures. Occasionally, binary16 caused visual artifacts for high resolution higher order fields, even with range compression of the coefficients. A global Legendre basis exhibited the same visual artifacts with binary16. Using a per-sample normalized basis alleviates this problem, however, it also means that the samples are no longer in a global basis, as such, manual interpolation has to be implemented for their filtering.

Orthonormal bases have various numerical and computational advantages over power bases, however, note that tensor product constructs can not be trivially used. For example, the 3 dimensional tensor product of quadratic basis polynomials requires $3^3 = 27$ coefficients, whereas a second order Taylor polynomial only constrains 10. One either stores the extra coefficients or computes them from the data of the Taylor expansion, for example using techniques similar to Ferguson's cubic tensor product Bézier surfaces that reconstruct first order Hermite data, as shown in Section 3.3.

4.2 Higher Order Filtering

By filtering, we refer to the interpolation scheme that combines samples into a local approximation to the sampled function. GPUs offer hardware accelerated uni-, bi- and trilinear interpolation of 1D, 2D, and 3D textures.

Even though the filtering techniques presented in this section can leverage these units, GPUs may carry out the hardware accelerated interpolation using more coarsely quantized subtexel texture coordinates than desired [27], that may necessitate manual interpolation of the data. This could be the case for low resolution higher order fields where the filtering cells span a large spatial extent.

As discussed in the previous subsection, Taylor and LSQ higher order samples consist of coefficients of multivariate polynomials in a global basis. Interpolating these coefficients yields another polynomial that can be evaluated to compute the approximation to the function at the query position.

More generally, filtering can be formulated as a $\sum_{\boldsymbol{j}} \lambda_{\boldsymbol{j}}(\boldsymbol{x}) f_{\boldsymbol{j}}(\boldsymbol{x})$ barycentric combination of $f_{\boldsymbol{j}}(\boldsymbol{x})$ higher order sample functions with some, not necessarily linear, barycentric weighting functions $\lambda_{\boldsymbol{j}}(\boldsymbol{x})$ that satisfy $\sum_{\boldsymbol{j}} \lambda_{\boldsymbol{j}}(\boldsymbol{x}) \equiv 1$.

The $\lambda_{\boldsymbol{i}}(\boldsymbol{x})$ barycentric filtering functions have to be such that they preserve the higher order derivatives specified by the samples at the sample positions, that is

$$\partial^{\boldsymbol{\alpha}} \sum_{\boldsymbol{j}} \lambda_{\boldsymbol{j}}(\boldsymbol{x_i}) f_{\boldsymbol{j}}(\boldsymbol{x_i}) = \partial^{\boldsymbol{\alpha}} f_{\boldsymbol{i}}(\boldsymbol{x_i}) \quad , \forall 0 \leq |\boldsymbol{\alpha}| \leq k \tag{5}$$

should hold under the $\mathcal{J}$ filtering footprint, $\boldsymbol{i} \in \mathcal{J}$. This can be done by combining univariate Hermite solutions to the problem of

$$\lambda(1) = 0 \ , \ \forall i \in \{0, 1\} : \forall m \in \{1, \ldots, k\} : \partial^m(i) = 0 \tag{6}$$

The degree 1, 3, 5, and 7 polynomial solutions to this are

$$h_1(t) = t \quad , \quad h_3(t) = -2t^3 + 3t^2$$
$$h_5(t) = 6t^5 - 15t^4 + 10t^3 \quad , \quad h_7(t) = -20t^7 + 70t^6 - 84t^5 + 35t^4 \ .$$

These blending functions retain the zero, first, second, and third order derivatives, respectively, at the sample positions $\boldsymbol{x_i}$.

Let us first consider the case of cubical filtering footprints. In this case interpolation uses the samples at the vertices of the enclosing hypercube. Thus, we can decompose the n-variate $\lambda_i(\boldsymbol{x})$ functions into a tensor product of single variable smoothing functions.

To use GPU hardware interpolation in conjunction with these blending functions, one has to replace the $u, v, w \in [0, 1]$ fractional part of the texture coordinates by $\lambda(u), \lambda(v), \lambda(w)$. In our experiments, this only caused a minor increase in query times.

In addition to satisfying the partial derivative reconstruction constraints of (5), higher order filtering may be also used to increase the continuity of the reconstructed surface. It is important to note, however, that by Inequality (2), the filtered error is within the convex hull of the individual errors. As such, higher order interpolation may result in higher error than nearest neighbor sampling.

Let us now consider the problem of defining a multivariate blending function from 1D functions for the case of arbitrary sample positions in $\mathbb{R}^n$ and the restricted filtering footprint of the simplex formed by the closest $n+1$ samples. Let $\{\boldsymbol{x}_{\boldsymbol{\alpha}}\}_{|\boldsymbol{\alpha}| \leq 1} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n\}$ be the vertices of this simplex and $\boldsymbol{u} = (u_0, u_1, \ldots, u_n) \in \mathbb{R}^{n+1}$ the n dimensional barycentric coordinates of $\boldsymbol{x}$ with respect to $\{\boldsymbol{x}_{\boldsymbol{\alpha}}\}_{|\boldsymbol{\alpha}| \leq 1}$, i.e. $\boldsymbol{x} = \sum_{i=0}^{n} u_i \boldsymbol{x}_i$.

Then the multivariate blending functions that satisfy (6) can be constructed from a single variable $\lambda : \mathbb{R} \to \mathbb{R}$ blending function as

$$\lambda_i(\boldsymbol{u}) = \lambda(u_i) \quad (i = 1, \ldots, n), \qquad \lambda_0(\boldsymbol{u}) = 1 - \sum_{i=1}^{n} \lambda_i(\boldsymbol{u}) \ .$$

by direct computation, provided $\lambda(t)$ is a solution to (6).

5 LEAST SQUARES DISTANCE FIELD GENERATION ALGORITHM

Taylor and Hermite sample field generation is a straightforward process, as we only have to evaluate or approximate the partial derivatives of the target function at the sample positions. Moreover, on the data level these constructs only differ in the manner in which these derivative data are stored. Hermite constructs use them directly, while Taylor fields first construct local Taylor polynomials and then expand them in a global basis and store those coefficients.

In contrast, least squares fields require a fitting and there are various construction parameters that affect the quality of the fit. Let us now derive a GPU-friendly fitting algorithm for LSQ fields.

For every $\boldsymbol{x_i}$ sample position of the grid, let us construct a small *fine grid*, that collects a number of distance samples in the neighborhood of $\boldsymbol{x_i}$. Let this fine grid be $\boldsymbol{s_{ij}} = \boldsymbol{x_i} + [i \cdot \Delta a, j \cdot \Delta b, k \cdot \Delta c]^T$, $\boldsymbol{j} = (i, j, k) \in \{-H, \ldots, H\}^3$, i.e. symmetric about $\boldsymbol{x_i}$ and let us denote the collection of fine grid distance samples by $f_{\boldsymbol{ij}}$. Let N denote the total number of samples, $N = (2H+1)^3$.

The best fit polynomial satisfies the following

$$\sum_{|\boldsymbol{k}| \le n} \boldsymbol{a_{ik}} \boldsymbol{s_{ij}^{k}} \approx f_{\boldsymbol{ij}} \quad , \boldsymbol{j} \in \{-H, \ldots, H\}^3 \tag{7}$$

where $(a_{\boldsymbol{i}1}, a_{\boldsymbol{i}2}, \ldots, a_{\boldsymbol{i}K})$, $K = \binom{k+3}{3}$ are the sample coefficients and the fine grid distance samples are $f_{\boldsymbol{i}1}, f_{\boldsymbol{i}2}, \ldots$

The best fit polyomial can be sought in various norms. For the ease of GPU implementations, we propose the use of the least-squares (LSQ) optimal fit, as optimization simplifies to a matrix-vector multiplication in this case. Moreover, the matrix in question only needs to be computed once.

To show this, recall that the Moore-Penrose pseudoinverse is the least-squares solution to (7). If $\boldsymbol{X_i^T} \cdot \boldsymbol{X_i}$ is invertible, it can be written as $\boldsymbol{X_i^+} = (\boldsymbol{X_i^T} \cdot \boldsymbol{X_i})^{-1} \cdot \boldsymbol{X_i^T}$ and the optimal coefficients are

$$\boldsymbol{a_i} = \boldsymbol{X_i^+} \cdot \boldsymbol{f_i} \ .$$

Weighting can be also incorporated to lessen the impact of samples far away from the sample position. The pseudoinverse can be computed even if $\boldsymbol{X_i^T} \cdot \boldsymbol{X_i}$ is not invertible, e.g. via SVD. As we are only dealing with GPU-friendly degrees, i.e. up to 3, the condition number of the normal matrices is low enough to avoid numerical precision issues during fitting.

Now, consider the zero centered pre-computed pseudoinverse $\boldsymbol{X_0^+}$. Then $\boldsymbol{X_0^+} \cdot \boldsymbol{f_i}$ is the best fit as if the origin was at $\boldsymbol{x_i}$, that is, in a power basis that changed origins from $[0, 0, 0]$ to $\boldsymbol{x_i}$. All that remains is to expand the coefficients of the polynomial after a translation by $\boldsymbol{x_i}$. In terms of memory and computation, this is more efficient and robust than precomputing the pseudoinverse for every sample position individually.

Algorithm 1 summarizes the proposed general construction of order k algebraic signed distance fields. Figure 4 illustrates the fitting process.

Regardless of the norm chosen, the best fit polynomial converges to the Taylor polynomial as the fine grid extent tends to zero. Intuitively speaking, the size of the fine grid acts as a low pass filter on the geometry. In this sense, the fine grid is a level-of-detail control.

We found that the estimated gradients for first order samples are almost always of unit-length. They only deviate around the cut locus, where the samples collapse to piece-wise constant approximations consisting of the distance to the boundary.

```
In  : distance query $f$ from input geometry;
      $k \geq 0$ distance field order;
      $\boldsymbol{x_i}$ distance field sample positions;
      $\Delta x, \Delta y, \Delta z > 0$ grid sizes;
      $\boldsymbol{s_{0j}}$ origin centered fine grid, $\boldsymbol{j} \in \{-H, \dots, H\}^3$;
      $\boldsymbol{X_0^+}$ zero centered pseudoinverse
Out: $(a_{\boldsymbol{i}1}, a_{\boldsymbol{i}2}, \dots, a_{\boldsymbol{i}\binom{k+3}{3}})$, order $k$ distance field samples.
for all $\boldsymbol{x_i}$ sample position do
    1. Sample distance function on translated fine grid
       $\forall \boldsymbol{j} : \boldsymbol{x_{ij}} = \boldsymbol{x_i} + \boldsymbol{s_{0j}} : \quad f_{\boldsymbol{j}} \leftarrow f(\boldsymbol{s_{ij}}) = f(\boldsymbol{x_i} + \boldsymbol{s_{oj}})$
    2. Flatten fine grid samples into $\boldsymbol{f} \leftarrow [f_1, \dots, f_{(2H+1)^3}]^T$
    3. Compute $\boldsymbol{x_i}$ centered approximation $\tilde{\boldsymbol{a}} \leftarrow \boldsymbol{X_o^+} \cdot \boldsymbol{f}$
    4. Translate $\tilde{\boldsymbol{a}}$ to global power basis and store as $f_{\boldsymbol{i}}$
end
```

Algorithm 1: Uniform higher order distance field construction.

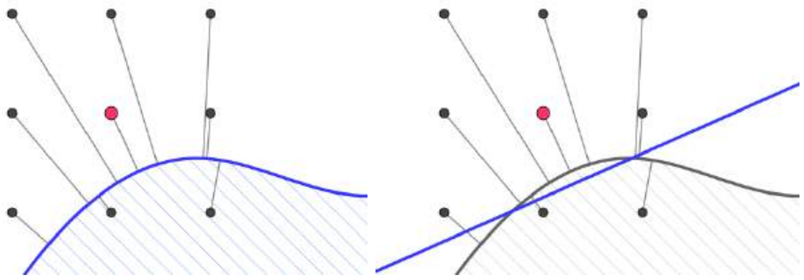


Figure 4: The first order grid sample $\boldsymbol{x_i}$ (in red) uses distance samples on the fine grid (in black) to fit a plane (blue line).

6 IMPROVEMENTS

6.1 Bit Packing of Samples

This optimization holds for first order fields. A first order sample only possesses 3 scalar degrees of freedom, since the gradient of the SDF is of unit length at regular points. Using e.g. octahedral encoding [26], $a_{\boldsymbol{i}}, b_{\boldsymbol{i}}, c_{\boldsymbol{i}}$ can be represented by two scalars with high precision, reducing the per-sample scalar storage requirements from 4 to 3.

The remaining 3 scalars of a first order sample can be further compressed by packing them into 32 bits. In our particular test scenes, a 10-10-12 encoding in the sense of allocating 10+10 bits to an octahedral direction plus 12 bits for the constant term in signed normalized integer format provided the highest accuracy. Note that this approach increases the arithmetic load due to unpacking and hardware interpolation cannot be used.

Packing remains a way to decrease the per-sample payload sizes regardless of the target function. However, it must be noted that the coefficients are arbitrary signed real numbers. We suggest range compression with

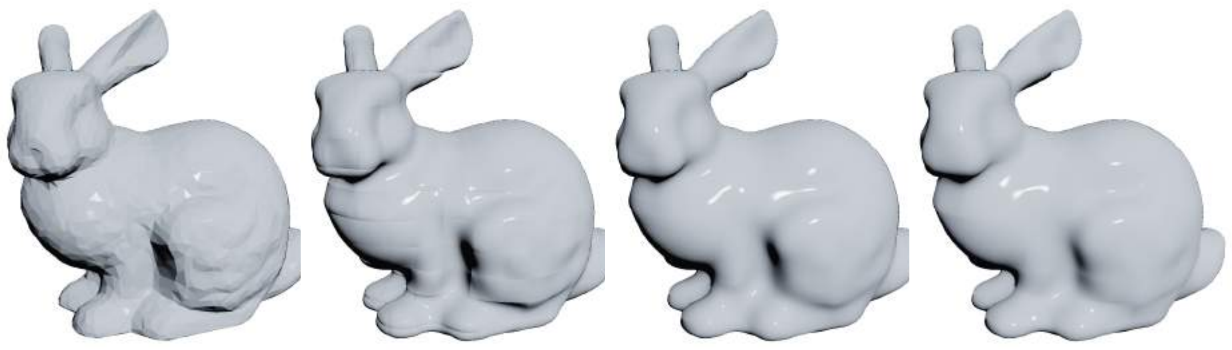

(a) Original mesh (b) $\epsilon = 0.01$ differences (c) $\epsilon = 1.0$ differences (d) Polynomail normals

Figure 5: Comparison of different surface normal estimation techniques. Note the effect of ϵ in central differences. (a) The original mesh with face normals (5K triangles). (b) and (c) calculate the normal with central differences (6 samples) with $\epsilon = 0.01$ and $\epsilon = 1.0$, respectively, on a second order DSDF with 16^3 resolution. (d) Taylor approximation for the normal on the same 16^3 second order SDF as (b) and (c).

per coefficient ranges and using signed normalized packing in these cases.

If manual packing is used, it is possible to redistribute the bits among the coefficients unevenly. To maximize precision, recall that perturbations in the lower order coefficients introduce the largest error according to the Taylor bound. As such, using less bits for the higher-order terms is preferred.

6.2 Polynomial Normals

Traditional techniques rely on finite differences, usually central differences, to estimate the partial derivatives of the SDF and the surface normal or store them directly in the field. In our constructs, order 1 and above, we can use the gradient of the filtered polynomial as the surface normal approximation at the ray-surface intersection point $\boldsymbol{x}$. Let $\hat{f}(\boldsymbol{x}) = \sum_{i+j+k \leq d} a_{ijk} x^i y^j z^k$ be the interpolated order d approximation. Then the surface normal is approximated by evaluating

$$\nabla f = \begin{bmatrix} \sum i \cdot a_{ijk} x^{i-1} y^j z^k \\ \sum j \cdot a_{ijk} x^i y^{j-1} z^k \\ \sum k \cdot a_{ijk} x^i y^j z^{k-1} \end{bmatrix} .$$

For a $a_i x + b_i y + c_i z + d_i$ interpolated first order sample, this simplifies to $[a_i, b_i, c_i]^T$.

Figure 5 shows the visual difference between surface normals estimated by central differences and the higher order samples.

6.3 Tetrahedral Tracking

As noted in Section 3.1, the error bounds are independent of the sampling topology and Section 4.2 shows that the filter footprint topology is not restricted to rectangles or boxes. By tetrahedral tracing we refer to the technique of combining samples of the enclosing simplex – a tetrahedron – instead of a cube.

The distance field itself is still composed of a regular grid, but we only fetch the 4 vertices of the tetrahedron instead of the 8 required for tensor interpolation. This allows us to efficiently compute barycentric weights.

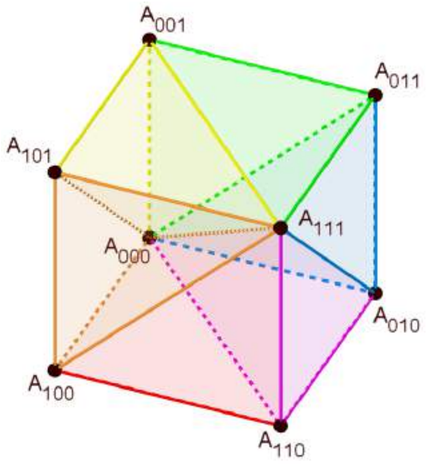


Figure 6: Decomposition of a grid cell into 6 tetrahedra. All of them share an edge which is one of the cell's main diagonals ($A_{000}A_{111}$).

To do so, first we have to identify which tetrahedron contains the query position, then compute the barycentric coordinates with respect to the vertices of that tetrahedron. The calculations are done in normalized cell coordinates ($\boldsymbol{x} = (x, y, z) \in [0, 1]^3$). The enclosing tetrahedron can be found by ordering the coordinates, since the six tetrahedra are

$$z \le y \le x, \qquad y \le z < x, \qquad z < x \le y, \qquad x \le z < y, \qquad y < x \le z, \qquad x < y < z. \tag{8}$$

Let us consider the configuration shown on Figure 6 and derive the weights for the first tetrahedron, highlighted with red in the figure. The vertices are A_{000}, A_{111}, A_{100}, and A_{110}, where the coordinates of A_{ijk} is $[i, j, k]^T \in \{0, 1\}^3$. The two equations defining the weights are

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = w_0 \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T + w_1 \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T + w_2 \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T + w_3 \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T, \tag{9}$$

$$1 = w_0 + w_1 + w_2 + w_3. \tag{10}$$

The solution is $w_0 = 1 - x$, $w_1 = z$, $w_2 = x - y$, $w_3 = y - z$. The weights in the remaining five tetrahedra can be calculated similarly but we can write this more concisely. Let $a \le b \le c$, where $a = \min(x, y, z)$, $c = \max(x, y, z)$ and b the remaining coordinate. Then

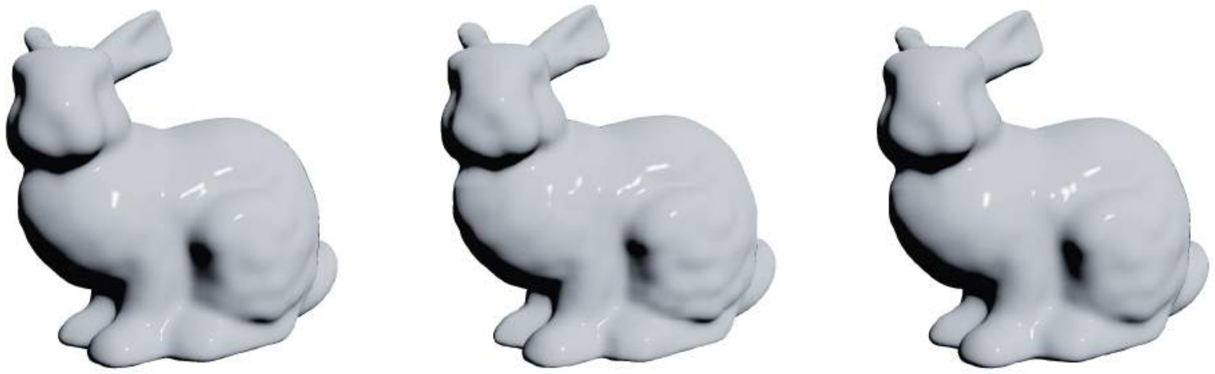
$$w_0 = 1 - c, \qquad w_1 = a, \qquad w_2 = c - b, \qquad w_3 = b - a. \tag{11}$$

The vertices are in a similar order: A_{000} and A_{111} are always the first two, w_2 corresponds to the remaining vertex closer to A_{000} and w_3 to the vertex closer to A_{111}.

7 TEST RESULTS

The tests focused on three aspects of distance fields: accuracy, storage, and query performance. The latter was measured in DSDF rendering scenarios.

Increasing the order can be considered as means to optimize computation localization: the higher order the sample, the more descriptive power it has and its evaluation cost increases respectively. The trade-off to be made is how much computation we distribute through the samples of the field and how much is done



(a) Best $\|\cdot\|_1$ fit. (b) Best $\|\cdot\|_2$ fit. (c) Best $\|\cdot\|_\infty$ fit.

Figure 7: Comparison of best LSQ fits in different norms. The field is a second order DSDF with the resolution of $32 \times 32 \times 32$. Note the subtle differences in reflections. Since both visual and error metric results were similar, we chose $\|\cdot\|_2$ due to its simplicity and efficiency for field generation.

locally. Our results suggest that algebraic signed distance fields up to order 2 are viable in real time graphics applications, while order 3 with its 20 scalars per-sample payload is less fit for real time graphics, unless used with nearest neighbor sampling.

We implemented the proposed higher order signed distance field construction framework in C++, using OpenGL 4.6. For meshes, we resolved the distance query of Algorithm 1 by converting models into a high resolution (512^3) distance field and filtering it trilinearly to obtain the fine grid samples. In the procedural scenes the distance functions were evaluated directly. The performance tests used perturbed texture coordinates in conjunction with GPU hardware interpolation as described in Section 4.2, except for tetrahedral and Ferguson-Hermite solutions that necessitated manual interpolation.

Our tests were carried out on an AMD RX 5700 and an NVIDIA GeForce RTX 2080 video card.

7.1 Least-Squares Field Construction Parameters

The number of fine grid samples had a small effect on the error beyond size 5^3. We tested grids up to size 51^3. The finer grids had slightly better error metrics, $1-2\%$ on average, depending on the models. For the tests we chose 23^3 as a default, but in general 5^3 is usually a sufficient choice.

We tested the effect of the fine grid extend with an $\alpha \in [0,1]$ parameter, that controlled whether the fine grid was a singular point ($\alpha = 0$) or if it extended all the way to the neighboring sample positions ($\alpha = 1$). On first, second, and third order fields the optimum for this parameter was found at about 0.6 by fitting these fields to various models.

An interesting note is that by using small values, or even exact Taylor expansions, lower resolution higher order fields can maintain geometric detail that are otherwise unattainable by the same resolution zero order fields. This comes at the expense of the overall approximation precision.

We compared $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$ norms for the fit, but there were no statistically significant differences in field inference precision. As such, we have chosen $\|\cdot\|_2$, as fits with respect to it can be easily and efficiently implemented on the GPU. Please refer to Figure 7 for a visual comparison.

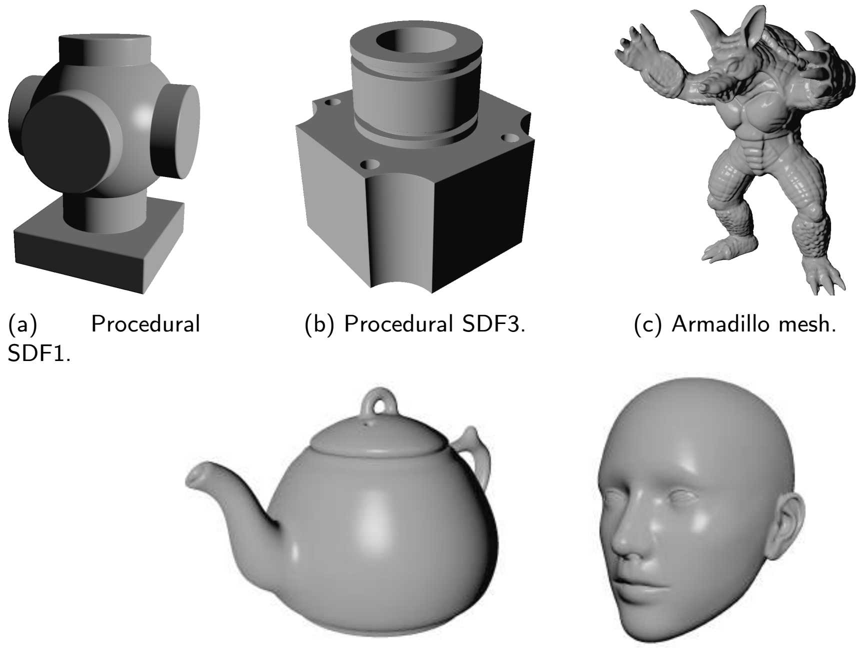

(a) Procedural SDF1.
(b) Procedural SDF3.
(c) Armadillo mesh.
(d) Teapot.
(e) Head.

Figure 8: The test models used in our accuracy and performance tests.

7.2 Accuracy Tests

Accuracy tests resampled lower resolution (8^3 to 128^3) higher order DSDFs to 257^3 zero order ground-truth DSDFs. The geometries were normalized to be zero centered and within the bounds of $[-1, 1]^3$. We computed the mean absolute error between the ground truth and the upsampled distance fields. The geometries used for accuracy measurements are shown on Figure (8) and the detailed results can be found in Appendix C. The Teapot and Human Head models are from the dataset published in [39].

As established in the section on filtering, the interpolated field values may not necessarily be more precise than the results of nearest point sampling. In general, trilinear interpolation does improve accuracy, but an interesting observation deserves specific mention. Other than the Ferguson-Hermite construction, no other filtering could boost accuracy as much as trilinear interpolation of zero order samples. One could argue that this is due to the fact that the latter can be considered as a zero order Ferguson-Hermite solution.

The most important aspect of accuracy for our purposes is whether the per sample accuracy increase is enough to reduce the overall storage requirements, given a set precision. The baseline for these measurements were trilinearly filtered zero order fields. We measured at what resolution do higher order fields match the precision of $32^3, 64^3, 96^3, \ldots, 256^3$ trilinearly filtered zero order DSDFs and averaged the scalar storage ratio.

The table below shows our average storage percentages such that the higher order field with the given filtering produces at most the same maximum error. We used the Armadillo model in the measurements below. O1, O2, O3 refers to DSDFs of order 1, 2, and 3. H1, H3, H5, H7 are blending based interpolation over cubical domains, as presented in Section 4.2. O1F and O2F are tensor tricubic and triquintic Ferguson-like solutions to first and second order Hermite interpolation problems.

Armadillo	O1 H3	O1F	O2 H5	O2F	O3 H7
Maximum	95.85%	52.55%	**50.65%**	51.76%	52.76%
Mean	137.49%	**55.95%**	100.21%	56.35%	102.56%
Median	130.07%	**31.16%**	96.63%	37.62%	118.63%

As noted in the Section 4.2, reconstruction of all higher order partial derivatives at the data points with the Hermite blending may not necessarily improve on the error metrics. Its main purpose is to increase the order of continuity between the cells formed by the samples. Indeed, comparing the cubic, quintic, and septic Hermite blended errors with trilinear interpolation of the the first, second, and third order polynomials, Hermite blended error is actually worse in maximum error. Please refer to the table below. More interestingly, nearest point sampling yields consistently better error metrics than the blended Hermite solutions for order 3 fields, and performs similarly in some for order 2.

Armadillo	O1 nearest	O1 linear	O2 nearest	O2 linear	O3 nearest	O3 linear
Maximum	300.71%	82.76%	75.58%	**46.71%**	52.96%	56.64%
Mean	174.25%	156.09%	105.52%	116.59%	**103.66%**	135.46%
Median	156.09%	132.44%	**70.58%**	106.86%	76.95%	131.80%

Let us compare these statistics to the ones taken on a torus. The mean and median error statistics are an order of magnitude better for DSDF order appropriate filtering.

Torus	O1 cubic	O1F	O2 quintic	O2F	O3 septic
Maximum	21.24%	**21.26%**	26.05%	22.60%	25.40%
Mean	105.12%	6.60%	12.48%	**5.27%**	9.83%
Median	102.73%	2.92%	6.17%	**2.63%**	5.20%

The nearest and trilinear filtered data is as follows. Note the precision of the nearest neighbor sampled second and third order fields.

Torus	O1 nearest	O1 linear	O2 nearest	O2 linear	O3 nearest	O3 linear
Maximum	46.05%	**19.56%**	25.41%	24.38%	25.02%	21.52%
Mean	184.91%	104.70%	12.06%	13.98%	**6.83%**	11.46%
Median	184.91%	101.18%	4.94%	4.86%	**1.48%**	2.35%

In general, the best storage is attained by using Ferguson-Hermite fields, however, the DSDF order appropriate blending proved to be close to them in maximum error metrics. For median and mean errors, the blended filtering falls behind, for complex models rather significantly.

On the same resolution, higher order fields provide higher accuracy. The trend roughly follows that order $N+1$ nearest sampling is more precise than order N trilinear. Order 3 nearest neighbor sampling is a unique case in the sense that even though it fetches 20 scalars per query, the visual quality is still very high given enough resolution, see Figure 9. Recall that 20 scalars are still only 62.5% of the data a trilinearly interpolated first order field requires per query.

Additionally, polygonal models consisting of a small number of large planar faces, such as a dodecahedron, do not benefit from higher order fields. This can be explained by the fact that large portions of the space

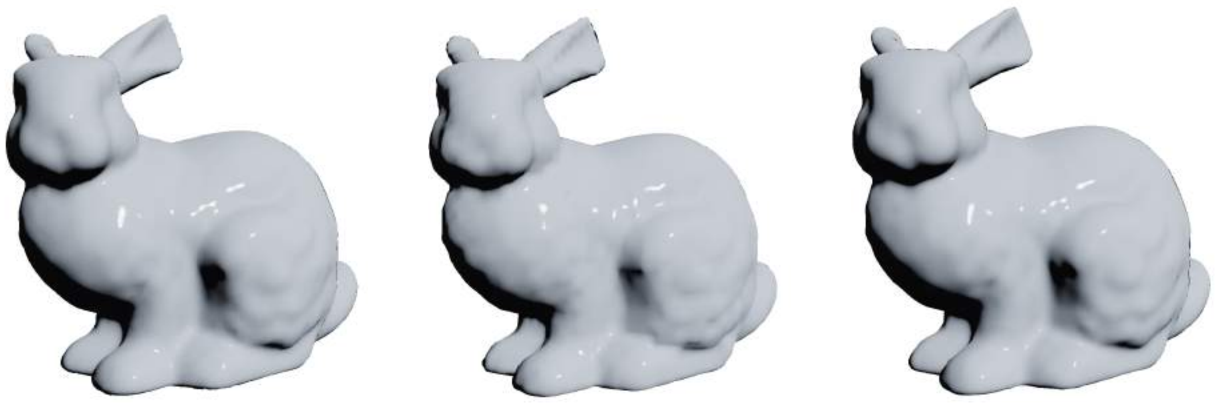


Figure 9: Comparison between a nearest point sampled (left) and trilinear filtered (right) third order DSDF of dimensions $32 \times 32 \times 32$ DSDF with a trilinearly filtered first order field of dimensions $51 \times 51 \times 51$ (center).

in the small neighborhood of the model possess linear distance functions that can be exactly represented by trilinearly interpolated zero order fields.

For simple models, tetrahedral interpolation did not increase error metrics considerably (about 3.3%), however, for complex models it tends to fall significantly behind.

Overall, our numbers show that the theoretically expected storage reduction based on Taylor bounds (see Appendix B) was not met on complex models, but simple ones did approach it. Other than these fields being only least-squares fits, the most likely cause to this is the cut locus and the non-differentiability of the signed distance function along it. This hinders the capacity of higher order samples to accurately cover larger areas.

7.3 Performance Tests

We compared the run time costs of rendering various DSDF geometries. O0-3 corresponds to DSDF order, TN denotes that the normals were computed using the gradient of the interpolated polynomial instead of central differences. Render times are in milliseconds, averaged over 100 frames each on 1920x1080 resolution with the SDF spanning the majority of the scene. The tables below used binary32 components.

On average, using binary32 components over binary16 increased render times by 7.1% on AMD with higher order polynomial normals (3.75% with central differences), while NVIDIA average frame render duration increased by 34.7% with higher order polynomial normals (26.86% with central differences). However, it must be noted that due to precision artifacts the renders were not equivalent, and as such the trace durations may not be comparable.

On an AMD RX 5700 and NVIDIA 2080 GPU, our measurements are summarized as

	AMD TN FP32				NV TN FP32			
Sampling	**O0**	**O1**	**O2**	**O3**	**O0**	**O1**	**O2**	**O3**
Nearest	0.33	0.3	0.35	0.41	0.24	0.17	0.26	0.49
Trilinear	0.3	0.52	0.92	1.61	0.24	0.74	1.82	3.64
Cubic Hermite		0.56	0.94	1.64		0.73	1.79	3.7
Quintic Hermite			0.98	1.68			1.86	3.53
Septic Hermite				1.69				3.52
Tetrahedral linear	0.71	0.55	0.69	0.92	0.6	0.4	0.99	1.81
Ferguson-Hermite		1.02	1.94			1.08	2.90	

Polynomial normals are a considerable performance boost on both vendors. Compared to central differences normals, they reduced frame times to 79.83% on AMD and 77.98% on NVIDIA.

In terms of manual interpolation, a particular AMD specific optimization we applied was related to fetching samples. The RDNA architecture provides a fast direct path to GPU RAM for image load operations [3] as long as they are not mixed with filtered texture accesses. Although it improved performance on AMD, our NVIDIA tests took a performance hit compared to the texture gather approach.

Note that tetrahedral tracing and Ferguson-Hermite figures in the above two tables correspond to manual filtering, without GPU interpolation hardware. Yet, tetrahedral interpolation improved performance over hardware accelerated interpolation on NVIDIA even for first order fields, and second order and up on AMD.

The cost of higher order blended filtering is minimal. These were implemented by changing the fractional part of the lookup texture coordinate.

We also implemented Koschier et al.'s hp adaptive construction [25], however, the resulting query times were very high. Rendering a 1920x1080 image of a non-uniform grid of second order polynomials took 14ms on average. This already used a LUT optimization to avoid the full traversal of the space division octree. We modified their approach by replacing projection by LSQ fitting. This produced higher visual quality at orders of magnitude reduced construction times.

8 CONCLUSIONS

We presented a general higher order sample field framework and showed three realizations of it: Taylor polynomials, least squares polynomials, and Ferguson-Hermite interpolants. Theses constructs were validated by applying them to signed distance functions.

We presented a higher order distance field construction framework that converts arbitrary distance queryable representations to order 1-3 fields. The least squares fit version can be trivially implemented on the GPU. We also evaluated $\|\cdot\|_1, \|\cdot\|_\infty$ best fits but they did not improve on our particular error metrics and test scenes.

With appropriate filtering, both first and second order distance fields can be rendered within 1ms at FullHD resolutions on an AMD RX 5700 and an NVIDIA 2080. Third order fields are only viable in real time scenarios with nearest neighbor sampling, however, in this case these are almost always more performant than first order filtered fields.

As the distance field samples now store higher order data, filtering needs to be adjusted so that these pieces of information are preserved at sample positions upon reconstruction. To this end, we proposed blend-like combination of samples both for tensor product and simplical interpolation approaches. We showed that the tensor product constructs can be used with hardware interpolation by re-adjusting the texture coordinates. Alternatively, we explored the Ferguson-type solution to the Hermite interpolation problem of interpolating first and second order partial derivatives of the signed distance function. We showed that this creates a visually improved interpolated result with significantly better accuracy, even compared to higher order fields, however, they are also more expensive to evaluate. Overall, blending based Hermite interpolation did not improve the error metrics of reconstruction, it mainly contributed to the visual smoothness of the reconstructed surfaces. In terms of accuracy, the Ferguson-Hermite fields performed the best, however, second order already showed overflattening due to the increasing number of zero partial derivatives.

Our results showed that higher order distance fields enable more efficient storage due to their higher per-sample accuracy. In practice, carrying this generalization to even higher orders is hindered by the coefficient explosion of the Taylor polynomials used as samples. Additional research is required into decreasing the per-sample payload sizes, either via compression, hierarchical storage, or alternative higher order constructions that still possess theoretically sound approximation order properties.

We showed that the performance hit of higher order samples can be mitigated by estimating surface normals from the gradient of the interpolated polynomials, and using simplical interpolation in the form of tetrahedral tracing.

ACKNOWLEDGEMENTS

Supported by the ÚNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

We would like to thank Visual Concepts for providing the AMD GPU used in the tests.

Gábor Valasek, http://orcid.org/0000-0002-0007-8647
Róbert Bán, http://orcid.org/0000-0002-8266-7444

REFERENCES

[1] A. Pasko, M.S.e.: International conference on shape modeling and applications, 2001, 7-11 may 2001, genoa, italy. IEEE Computer Society, 2001. ISBN 0-7695-0853-7.

[2] Aaltonen, S.: GPU-based clay simulation and ray-tracing tech in Claybook, Game Developers Conference 2018. In Game Developers Conference. San Francisco, CA, 2018.

[3] AMD: Introducing RDNA architecture. Whitepaper. https://www.amd.com/system/files/documents/rdna-whitepaper.pdf.

[4] Angles, B.; Tarini, M.; Wyvill, B.; Barthe, L.; Tagliasacchi, A.: Sketch-based implicit blending. ACM Trans. Graph., 36(6), 181:1–181:13, 2017. ISSN 0730-0301. http://doi.org/10.1145/3130800.3130825.

[5] Bálint, C.; Kiglics, M.: A geometric method for accelerated sphere tracing of implicit surfaces. Acta Cybernetica, 2021. http://doi.org/10.14232/actacyb.290007.

[6] Bálint, C.; Valasek, G.: Accelerating Sphere Tracing. In O. Diamanti; A. Vaxman, eds., EG 2018 - Short Papers. The Eurographics Association, 2018. ISSN 1017-4656. http://doi.org/10.2312/egs.20181037.

[7] Bálint, C.; Valasek, G.; Gergó, L.: Operations on signed distance functions. Acta Cybernetica, 24(1), 17–28, 2019. http://doi.org/10.14232/actacyb.24.1.2019.3.

[8] Bán, R.; Bálint, C.; Valasek, G.: Area Lights in Signed Distance Function Scenes. In P. Cignoni; E. Miguel, eds., Eurographics 2019 - Short Papers, 85–88. The Eurographics Association, 2019. http://doi.org/10.2312/egs.20191021.

[9] Bán, R.; Valasek, G.: First Order Signed Distance Fields. In A. Wilkie; F. Banterle, eds., Eurographics 2020 - Short Papers. The Eurographics Association, 2020. ISBN 978-3-03868-101-4. ISSN 1017-4656. http://doi.org/10.2312/egs.20201011.

[10] Bán, R.; Valasek, G.: Geometric distance fields of plane curves. Acta Cybernetica, 25(2), 187–203, 2021. http://doi.org/10.14232/actacyb.289248.

[11] Biswas, A.; Shapiro, V.: Approximate distance fields with non-vanishing gradients. Graphical Models, 66(3), 133–159, 2004. http://doi.org/10.1016/j.gmod.2004.01.003.

[12] Blu, T.; Unser, M.: Quantitative fourier analysis of approximation techniques: Part i - interpolators and projectors. IEEE Transaction on Signal Processing, 47, 2000.

[13] Cohen-Or, D.; Solomovic, A.; Levin, D.: Three-dimensional distance field metamorphosis. ACM Trans. Graph., 17(2), 116–141, 1998. ISSN 0730-0301. http://doi.org/10.1145/274363.274366.

[14] Csébfalvi, B.: Beyond trilinear interpolation: Higher quality for free. ACM Trans. Graph., 38(4), 2019. ISSN 0730-0301. http://doi.org/10.1145/3306346.3323032.

[15] Evans, A.: Learning from failure: a survey of promising, unconventional and mostly abandoned renderers for 'dreams ps4', a geometrically dense, painterly ugc game. In Advances in Real-Time Rendering in Games. MediaMolecule, SIGGRAPH, 2015.

[16] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design (3rd Ed.): A Practical Guide. Academic Press Professional, Inc., San Diego, CA, USA, 1993. ISBN 0-12-249052-5.

[17] Frisken, S.F.; Perry, R.N.; Rockwood, A.P.; Jones, T.R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In 27th Annual Conf. on Computer Graphics and Interactive Techniques, SIGGRAPH, 249–254. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. ISBN 1-58113-208-5. http://doi.org/10.1145/344779.344899.

[18] Fuhrmann, A.; Sobottka, G.; Groß, C.: Distance fields for rapid collision detection in physically based modeling, 2003.

[19] Galin, E.; Guérin, E.; Paris, A.; Peytavie, A.: Segment tracing using local lipschitz bounds. Computer Graphics Forum, 39(2), 545–554, 2020. http://doi.org/10.1111/cgf.13951.

[20] Gomes, A.; Voiculescu, I.; Jorge, J.; Wyvill, B.; Galbraith, C.: Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms. Springer Publishing Company, Incorporated, 1st ed., 2009. ISBN 184882405X, 9781848824058.

[21] Green, C.: Improved alpha-tested magnification for vector textures and special effects. In ACM SIGGRAPH 2007 Courses, SIGGRAPH '07, 9–18. ACM, New York, NY, USA, 2007. ISBN 978-1-4503-1823-5. http://doi.org/10.1145/1281500.1281665.

[22] Hart, J.C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer, 12, 527–545, 1996.

[23] Hartmann, E.: Parametric Gn blending of curves and surfaces blending functions. The Visual Computer - VC, 17, 1–13, 2001. http://doi.org/10.1007/PL00013398.

[24] Keinert, B.; Schäfer, H.; Korndörfer, J.; Ganse, U.; Stamminger, M.: Enhanced Sphere Tracing. In A. Giachetti, ed., Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference. The Eurographics Association, 2014. ISBN 978-3-905674-72-9. http://doi.org/10.2312/stag.20141233.

[25] Koschier, D.; Deul, C.; Bender, J.: Hierarchical hp-adaptive signed distance fields. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '16, 189–198. Eurographics Association, Goslar Germany, Germany, 2016. ISBN 978-3-905674-61-3.

[26] Meyer, Q.; Suessmuth, J.; Sussner, G.; Stamminger, M.; Greiner, G.: On Floating-Point Normal Vectors. Computer Graphics Forum, 2010. ISSN 1467-8659. http://doi.org/10.1111/j.1467-8659.2010.01737.x.

[27] NVIDIA Corporation: CUDA C++ Programming Guide; G.2. Linear Filtering, 2019. CUDA Toolkit v10.2.89.

[28] Osher, S.; Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer Verlag, 2003.

[29] Park, J.J.; Florence, P.; Straub, J.; Newcombe, R.; Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation, 2019.

[30] Pasko, A.; Adzhiev, V.; Sourin, A.; Savchenko, V.: Function representation in geometric modeling: concepts, implementation and applications. The Visual Computer, 11(8), 429–446, 1995. ISSN 1432-2315. http://doi.org/10.1007/BF02464333.

[31] Pottmann, H.; Hofer, M.: Geometry of the squared distance function to curves and surfaces. In H.C. Hege; K. Polthier, eds., Visualization and Mathematics III, 221–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-662-05105-4.

[32] Quilez, I.: Greek Temple on shadertoy. https://www.shadertoy.com/view/ldScDh. Accessed: 2021-10-02.

[33] Quilez, I.; Jeremias, P.: Siggraph Shadertoy Hackathon. In SIGGRAPH Studio, 25:1. ACM, 2014. ISBN 978-1-4503-2977-4. http://dblp.uni-trier.de/db/conf/siggraph/siggraph2014studio.html#QuilezJ14.

[34] Rebain, D.; Li, K.; Sitzmann, V.; Yazdani, S.; Yi, K.M.; Tagliasacchi, A.: Deep medial fields. ArXiv, abs/2106.03804, 2021.

[35] Ricci, A.: A Constructive Geometry for Computer Graphics. The Computer Journal, 16(2), 157–160, 1973. http://doi.org/10.1093/comjnl/16.2.157.

[36] Shapiro, V.: Semi-analytic geometry with r-functions. Acta Numerica, 16, 239–303, 2007. http://doi.org/10.1017/S096249290631001X.

[37] Sigg, C.; Hadwiger, M.: Fast third-order texture filtering, 313 – 329. Addison-Wesley, Upper Saddle River, 2005. ISBN 0-321-33559-7. .

[38] Song, X.; Jüttler, B.; Poteaux, A.: Hierarchical spline approximation of the signed distance function. In 2010 Shape Modeling International Conference, 241–245, 2010. http://doi.org/10.1109/SMI.2010.18.

[39] Takikawa, T.; Glassner, A.; McGuire, M.: A dataset and explorer for 3d signed distance functions. Journal of Computer Graphics Techniques (JCGT), 11(2), 1–29, 2022. ISSN 2331-7418. http://jcgt.org/published/0011/02/01/.

[40] Varma, A.; Katsifarakis, K.: Optimal error bounds for hermite interpolation. Journal of Approximation Theory, 51(4), 350–359, 1987. ISSN 0021-9045. http://doi.org/10.1016/0021-9045(87)90043-8.

[41] Wolff, S.; Bucher, C.: Distance fields on unstructured grids: Stable interpolation, assumed gradients, collision detection and gap function. Computer Methods in Applied Mechanics and Engineering, 259, 77 – 92, 2013. ISSN 0045-7825. http://doi.org/10.1016/j.cma.2013.02.015.

[42] Wright, D.: Dynamic occlusion with signed distance fields. In Advances in Real-Time Rendering in Games. Epic Games (Unreal Engine), SIGGRAPH, 2015.

[43] Wyvill, B.; Guy, A.; Galin, E.: Extending the CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. Computer Graphics Forum, 1999. ISSN 1467-8659. http://doi.org/10.1111/1467-8659.00365.

[44] Yamaguchi, F.: Curves and Surfaces in Computer Aided Geometric Design. Springer-Verlag, Berlin, Heidelberg, 1988. ISBN 0387174494.

Appendices

A Notation and theoretical background

We denote the components of a vector $\boldsymbol{x} \in \mathbb{R}^n$ by $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$. In case of 2D and 3D vectors, we also use $\boldsymbol{i} = [i, j]^T$, $\boldsymbol{x} = [x, y, z]^T$. Let $\boldsymbol{x} \cdot \boldsymbol{y}$ denote the dot product of $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

An $\boldsymbol{\alpha} = (\alpha_1, .., \alpha_n) \in \mathbb{N}^n$ tuple is a **multi-index**. Multi-indices are convenient tools for the manipulation of multivariable expressions. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a sufficiently smooth function and let us define the following operations:

$$|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \cdots + \alpha_n \;,$$
$$\boldsymbol{\alpha}! = \alpha_1! \cdot \alpha_2! \cdot \ldots \cdot \alpha_n! \quad \text{(where } 0! = 1\text{)},$$
$$\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n} \;,$$
$$\partial^{\boldsymbol{\alpha}} f = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \ldots \partial_n^{\alpha_n} f = \frac{\partial^{|\boldsymbol{\alpha}|} f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_n^{\alpha_n}} \;.$$

The **order of a multi-index** is $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \ldots + \alpha_n$.

If $f : \mathbb{R}^k \to \mathbb{R}$ is k times continuously differentiable on a set $A \subset \mathbb{R}^n$, we denote it by $f \in C^k[A]$.

Definition 1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be $f \in C^{k+1}$. The degree k multivariate Taylor approximation of f about $\boldsymbol{x}_0$ is*

$$T_{k,\boldsymbol{x}_0}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| \le k} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_0)}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} \;. \tag{12}$$

We omit the degree k in expressions involving multiple Taylor polynomials of the same degree. If the Taylor polynomials are taken about a fixed $\{\boldsymbol{x_i}\}$ set of samples, we also denote them by $T_{\boldsymbol{i}}(\boldsymbol{x})$.

Theorem 1 (Taylor's theorem) *Let $S \subset \mathbb{R}^n$ convex and $f : \mathbb{R}^n \to \mathbb{R}, f \in C^{k+1}[S]$. Then $\forall \boldsymbol{x}_0 \in S : \forall \boldsymbol{x} \in S :$*

$$f(\boldsymbol{x}) = T_{k,\boldsymbol{x}_0}(\boldsymbol{x}) + R_{k,\boldsymbol{x}_0}(\boldsymbol{x}) \;,$$

where $\exists t \in (0,1) : \tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + t \cdot (\boldsymbol{x} - \boldsymbol{x}_0)$ such that

$$R_{k,\boldsymbol{x}_0}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| = k+1} \frac{\partial^{\boldsymbol{\alpha}} f(\tilde{\boldsymbol{x}})}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} \;.$$

The above statement of the Taylor theorem uses the Lagrange form of the remainder. We default to it because it will provide the most insight into the nature of our constructions. However, a very convenient result of the above is the following

Corollary 1.1 *Let $S \subset \mathbb{R}^n$ convex and $f : \mathbb{R}^n \to \mathbb{R}, f \in C^{k+1}[S]$. Let $M \ge 0$ such that $\forall \boldsymbol{x} \in S : \forall |\boldsymbol{\alpha}| = k+1 : |\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x})| \le M$. Then*

$$|f(\boldsymbol{x}) - T_{k,\boldsymbol{x}_0}(\boldsymbol{x})| \le \frac{M}{(k+1)!} \|\boldsymbol{h}\|_1^{k+1} \;,$$

where $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{x}_0$ and $\|\boldsymbol{h}\|_1^{k+1} = (|h_0| + |h_1| + \cdots + |h_n|)^{k+1}$.

We denote the **set of all distance functions** by $SDF[\mathbb{R}^n]$. We call $\boldsymbol{x} \in \mathbb{R}^n$ **a regular point of** $f \in SDF[\mathbb{R}^n]$ iff the derivative of f exists at $\boldsymbol{x}$. All $f \in SDF[\mathbb{R}^n]$ are continuous [7], but $f \notin C^1[\boldsymbol{x}]$ if there are more than one boundary points closest to $\boldsymbol{x}$. F is the volume represented by f, that is $f = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \le 0\}$. The zero level-set of an arbitrary $f : \mathbb{R}^n \to \mathbb{R}$ is denoted by ∂F, i.e. $\partial F = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) = 0\}$.

B Storage analysis

Storing higher order polynomial approximations increases the per-sample and filtered accuracy alike. This enables sparser sampling without affecting the error. On the other hand, storing these higher order polynomials increases the per-sample payload. This section investigates the superposition of these two trends.

A degree k Taylor polynomial in n variables requires $\binom{n+k}{n}$ coefficients. Numerically, this translates to

degree	0	1	2	3	4	5	...	k
$\mathbb{R}$	1	2	3	4	5	6	...	$k+1$
$\mathbb{R}^2$	1	3	6	10	15	21	...	$\frac{(k+1)(k+2)}{2}$
$\mathbb{R}^3$	1	4	10	20	35	56	...	$\frac{(k+1)(k+2)(k+3)}{6}$

Thus, N order k samples require a storage of $N \cdot \binom{n+k}{n}$ scalars.

For the sake of simplicity, let us consider a regular grid of samples in the remainder of this section, i.e. $\boldsymbol{x_i} = \boldsymbol{o} + \sum_{j=1}^{n} \Delta_j \cdot \boldsymbol{e}_j$, $\Delta_j > 0$, where $\boldsymbol{e}_j$ are the canonical basis vectors of $\mathbb{R}^n$.

Let us investigate how much sparser the grid spacing $\Delta_1, \Delta_2, \ldots, \Delta_n$ can become when increasing the order from k to $k+1$ without inflating the error.

The ratio of the number of per-sample coefficients between the orders $k+1$ and k is $\frac{n+k+1}{k+1}$.

Let $M_i > 0$ denote the upper bound on the i-th derivative of $f : \mathbb{R}^n \to \mathbb{R}$ and let $\boldsymbol{d}_k = \boldsymbol{x_{i+1}} - \boldsymbol{x_i} = [\Delta_1, \ldots, \Delta_n]^T$. We assume that the farthest a query uses a sample is $\|\boldsymbol{d}_k\|_2$ away from the sample.

Let us find all $s_k > 0$ multipliers such that if $\boldsymbol{d}_{k+1} = s_k \cdot \boldsymbol{d}_k$, the error bound does not increase. From Corollary 1.1, this means

$$\frac{M_k}{k!} \|\boldsymbol{d}_k\|_1^k \geq \frac{M_{k+1}}{(k+1)!} \|\boldsymbol{d}_{k+1}\|_1^{k+1} = \frac{M_{k+1}}{(k+1)!} s_k^{k+1} \|\boldsymbol{d}_k\|_1^{k+1} ,$$

should hold, that is,

$$\sqrt[k+1]{\frac{M_k \cdot (k+1)}{M_{k+1} \cdot \|\boldsymbol{d}_k\|_1}} \geq s_k .$$

Solving the above for equality allows us to evaluate the maximum decrease of sample counts and total scalar storage as we use higher order fields while retaining the accuracy of the denser lower order field. These numbers also depend on the dimensionality n and the value of $\|\boldsymbol{d}_i\|_1 > 0$.

Since the bound in Corollary 1.1 uses the Manhattan norm, s_k corresponds to the multiplier of spacing, i.e. $\boldsymbol{x}_{\boldsymbol{i}}^{k+1} = \boldsymbol{o} + \sum_{j=1}^{n} s_k \cdot \Delta_j \cdot \boldsymbol{e}_j$. As such, in n dimensions, the ratio of the number of cells of the order $k+1$ field and that of the order k field is $\frac{1}{s_k^n}$, while the increase in the per-sample payload is $\frac{n+k+1}{k+1}$. Combining these, the total scalar storage multiplier as we increase the field order is

$$\frac{n+k+1}{(k+1) \cdot s_k^n} .$$

As long as spacing increase satisfies $s_k > \sqrt[n]{\frac{n+k+1}{k+1}}$, the total storage requirements decrease as the order goes from k to $k+1$.

The following tables illustrate the potential decrease in sample counts when increasing the field order to $1-5$ from 0, assuming $\|\boldsymbol{h}\|_1 = 0.1$ and a common $M > 0 : \forall i : M \geq M_i$ upper bound to replace all order specific M_i-s in the error bounds of Corollary 1.1:

field order	order $(k+1):0$ sample ratio		
$\|\boldsymbol{h}\|_1 = 0.1$	$\mathbb{R}$	$\mathbb{R}^2$	$\mathbb{R}^3$
$0 \to 1$	22.36%	5%	1.118%
$0 \to 2$	11.86%	1.41%	0.17%
$0 \to 3$	8.03%	0.65%	0.052%
$0 \to 4$	6.083%	0.37%	0.023%
$0 \to 5$	4.9%	0.24%	0.012%

If we take into account the increased per-sample scalar storage requirements, we get the following figures for the comparison of total scalar storage requirements between order 0 and $1-5$:

field order	order $(k+1):0$ storage ratio		
$\|\boldsymbol{h}\|_1 = 0.1$	$\mathbb{R}$	$\mathbb{R}^2$	$\mathbb{R}^3$
$0 \to 1$	44.72%	15.00%	4.47%
$0 \to 2$	35.57%	8.43%	1.67%
$0 \to 3$	32.14%	6.45%	1.04%
$0 \to 4$	30.42%	5.55%	0.79%
$0 \to 5$	29.42%	5.05%	0.66%

For example, the above table shows that even though per-sample payload sizes increase fourfold in $\mathbb{R}^3$ as the order increases from 0 to 1, the same precision can be attained with 4.47% of the original storage space. In practice, partial derivatives have different bounds, the numbers above only serve as a general outline of the expected accuracy and storage trends.

C Accuracy measurement data

Storage usage compared to trilinearly filtered order zero fields with matching accuracy.

		mean					median					max				
		O1c	O1F	O2q	O2F	O3s	O1c	O1F	O2q	O2F	O3s	O1c	O1F	O2q	O2F	O3s
Torus	64	105%	7%	19%	8%	13%	98%	3%	8%	2%	4%	21%	14%	31%	19%	31%
	128	105%	6%	10%	4%	8%	98%	2%	4%	1%	1%	21%	13%	26%	20%	26%
	192	—	4%	7%	3%	5%	108%	2%	2%	0%	1%	21%	14%	25%	17%	21%
Arma.	64	139%	65%	114%	67%	105%	139%	41%	125%	53%	119%	98%	50%	53%	41%	61%
	128	139%	60%	103%	60%	105%	121%	32%	88%	38%	99%	98%	55%	46%	50%	57%
	192	—	50%	96%	53%	97%	119%	25%	78%	31%	89%	79%	39%	37%	34%	42%
SDF1	64	149%	55%	103%	60%	105%	130%	14%	75%	26%	81%	27%	24%	41%	19%	44%
	128	149%	45%	88%	50%	93%	98%	6%	28%	7%	31%	41%	20%	41%	14%	26%
	192	—	47%	96%	48%	97%	88%	4%	14%	4%	16%	31%	12%	18%	15%	28%
SDF3	64	158%	55%	125%	67%	119%	202%	21%	125%	46%	150%	71%	55%	53%	46%	52%
	128	169%	60%	119%	63%	127%	139%	8%	43%	13%	52%	50%	35%	31%	28%	26%
	192	—	57%	125%	62%	124%	—	6%	24%	6%	26%	47%	57%	28%	29%	24%
Teapot	64	121%	33%	60%	31%	61%	98%	9%	31%	10%	31%	84%	65%	93%	46%	81%
	128	113%	22%	43%	20%	37%	91%	5%	16%	4%	13%	77%	48%	41%	43%	44%
	192	108%	18%	34%	16%	29%	88%	3%	9%	2%	7%	64%	33%	28%	35%	28%
Head	64	84%	16%	31%	13%	26%	65%	5%	13%	5%	10%	60%	31%	61%	98%	9%
	128	91%	12%	20%	9%	17%	65%	3%	6%	2%	5%	34%	16%	29%	88%	3%
	192	88%	10%	17%	7%	14%	64%	3%	4%	1%	3%	31%	13%	26%	65%	5%