First Order Signed Distance Fields

Róbert Bán and Gábor Valasek

Eötvös Loránd University, Hungary
{bundas,valasek}@inf.elte.hu

Abstract

This paper investigates a first order generalization of signed distance fields. We show that we can improve accuracy and storage efficiency by incorporating the spatial derivatives of the signed distance function into the distance field samples. We show that a representation in power basis remains invariant under barycentric combination, as such, it is interpolated exactly by the GPU. Our construction is applicable in any geometric setting where point-surface distances can be queried. To emphasize the practical advantages of this approach, we apply our results to signed distance field generation from triangular meshes. We propose storage optimization approaches and offer a theoretical and empirical accuracy analysis of our proposed distance field type in relation to traditional, zero order distance fields. We show that the proposed representation may offer an order of magnitude improvement in storage while retaining the same precision as a higher resolution distance field.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing;*** *Volumetric models;*

1. Introduction and previous work

Signed distance fields (SDF) are discretizations of continuous signed distance functions. They store the signed distance to the closest geometry at every sample. By applying a reconstruction filter, i.e. by blending the sampled values, they convey global geometric information about the scene at every point in space.

Distance fields mostly serve as auxiliary data structures in three-dimensional real-time computer graphics to provide effects like soft shadows from area lights, ambient occlusion, or sky visibility [Wri15]. In addition, they can be also used as a primary geometric representation on current generation hardware, as shown in [Aal]. SDF representations proved to be especially efficient for font rendering [Gre07]. Advances were also made in decreasing storage requirements [FPRJ00] and increasing accuracy [KDB16].

Our paper shows that both of the above improvements can be achieved by treating the reconstruction of a continuous signed distance function from samples as an approximation problem. In Section 2, we use Taylor's approximation theorem to justify the incorporation of higher order derivatives of the signed distance function in the distance field samples. The order of the highest derivative is referred to as the *order of the distance field* in this paper. Section 3 discusses how such representations can be filtered. Section 4 provides a practical construction algorithm for first order distance fields. In Section 5, we show that these are simultaneously more accurate and, as a result, more efficient in terms of storage than the classic, zero order uniform distance fields. Examples are presented that this storage gain can be over an order of magnitude, at the expense of an up to 10% performance hit in direct visualization.

2. Preliminaries

The approximation of a continuous function from samples has two main components: (i) the actual data to represent the samples and (ii) a reconstruction scheme that combines the samples to form a globally defined approximation.

In case of traditional distance fields, such as in [Gre07], a uniform grid stores the f_i signed distances of the grid points to the scene geometries. Our main insight is that we can consider these distance samples as zero order Taylor approximations to the global signed distance function $f : \mathbb{E}^3 \to \mathbb{R}$. This allows us to generalize the construction and to quantify the increase of accuracy due to storing higher order derivative data at sample points.

Let $\alpha = (\alpha_1, \dots, \alpha_n)$ denote a multi-index where $|\alpha| = \alpha_1 + \dots + \alpha_n$, $\boldsymbol{x}^\alpha = x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$, $\partial^\alpha f = \partial_1^{\alpha_1} \dots \partial_n^{\alpha_n} f$, $\alpha! = \alpha_1! \cdot \ldots \cdot \alpha_n!$. The degree k multivariate Taylor polynomial about $\boldsymbol{x}_0$ is

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

that is, indeed $T_{0,\boldsymbol{x}_0} = f(\boldsymbol{x}_0)$. In general, the approximation properties of $T_{k,\boldsymbol{x}_0}(\boldsymbol{x})$ are characterized by

Theorem 1 (Taylor's theorem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that $f \in C^{k+1}$ on an $S \subset \mathbb{R}^n$ convex set. Then $\forall \boldsymbol{x}_0 \in S : \forall \boldsymbol{x} \in S :$*

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

where $\tilde{\boldsymbol{x}} = (1 - c) \cdot \boldsymbol{x}_0 + c \cdot \boldsymbol{x}$ for some $c \in (0, 1)$.

If all derivatives are bounded by some constant $M > 0$, then

DOI: 10.2312/egs.20201011

$\left|f(\boldsymbol{x}) - T_{k,\boldsymbol{x}_0}(\boldsymbol{x})\right| \leq \frac{M}{(k+1)!}||\boldsymbol{x}-\boldsymbol{x}_0||_1^{k+1}$. Therefore, a single distance sample has an approximation order of 1 at regular regions, i.e. outside the medial axes of the represented geometries. Note that this approximation order is an actual bound on the maximum error, not just a polynomial precision property.

3. Approximation with first degree polynomials

In 3D, we can increase the local approximation order to 2 at every sample $\boldsymbol{x_i}$ by storing the $T_{1,\boldsymbol{x_i}}(\boldsymbol{x}) = f(\boldsymbol{x_i}) + \partial_x f(\boldsymbol{x_i})\cdot(x-x_i) + \partial_y f(\boldsymbol{x_i})\cdot(y-y_i) + \partial_z f(\boldsymbol{x_i})\cdot(z-z_i)$ first degree Taylor polynomial approximation of the signed distance function.

However, there are multiple bases to represent a polynomial such as (1). By stipulating that evaluation in the base should be invariant under trilinear filtering, we show that the global power basis is an adequate choice. Indeed, invariance under linear interpolation means that the order of evaluation and interpolation can be interchanged. It trivially holds for the power basis:

$$\begin{aligned}&(1-t)\cdot T_{1,\boldsymbol{x_i}}(\boldsymbol{x}) + t\cdot T_{1,\boldsymbol{x_j}}(\boldsymbol{x}) =\\ &\quad(1-t)\cdot\left(a_{\boldsymbol{i}}x+b_{\boldsymbol{i}}y+c_{\boldsymbol{i}}z+d_{\boldsymbol{i}}\right)+t\cdot\left(a_{\boldsymbol{j}}x+b_{\boldsymbol{j}}y+c_{\boldsymbol{j}}z+d_{\boldsymbol{j}}\right) =\\ &\quad\left((1-t)a_{\boldsymbol{i}}+ta_{\boldsymbol{j}}\right)x+\left((1-t)b_{\boldsymbol{i}}+tb_{\boldsymbol{j}}\right)y+\\ &\quad\left((1-t)c_{\boldsymbol{i}}+tc_{\boldsymbol{j}}\right)z+(1-t)d_{\boldsymbol{i}}+td_{\boldsymbol{j}}\end{aligned}$$

Invariance under bi- and trilinear interpolation follows immediately from the above. The $a_{\boldsymbol{i}}, b_{\boldsymbol{i}}, c_{\boldsymbol{i}}, d_{\boldsymbol{i}}$ coefficients are computed from the signed distance function as

$$\begin{aligned}&f(\boldsymbol{x_i}) + \partial_x f(\boldsymbol{x_i})\cdot(x-x_i) + \partial_y f(\boldsymbol{x_i})\cdot(y-y_i) + \partial_z f(\boldsymbol{x_i})\cdot(z-z_i) =\\ &\qquad\underbrace{\partial_x f(\boldsymbol{x_i})}_{a_i}\cdot x + \underbrace{\partial_y f(\boldsymbol{x_i})}_{b_i}\cdot y + \underbrace{\partial_z f(\boldsymbol{x_i})}_{c_i}\cdot z + \qquad (2)\\ &\qquad\underbrace{f(\boldsymbol{x_i}) - \partial_x f(\boldsymbol{x_i})\cdot x_i - \partial_y f(\boldsymbol{x_i})\cdot y_i - \partial_z f(\boldsymbol{x_i})\cdot z_i}_{d_i}\end{aligned}$$

Note that the above construction can approximate arbitrary functions, it is not restricted to signed distance functions. For our particular use case, the $a_{\boldsymbol{i}}, b_{\boldsymbol{i}}, c_{\boldsymbol{i}}$ coefficients of the linear Taylor approximation coincide with the partial derivatives of the signed distance function, i.e. they form a unit vector and only posses two scalar degrees of freedom.

4. First order SDF in practice

In this section, we investigate the problem of constructing a uniform grid of linear Taylor approximations $\boldsymbol{a}_{ijk} = \boldsymbol{a_i} = (a_{\boldsymbol{i}}, b_{\boldsymbol{i}}, c_{\boldsymbol{i}}, d_{\boldsymbol{i}}) \equiv T_{1,\boldsymbol{x}_{ijk}}(\boldsymbol{x}) = a_{\boldsymbol{i}}x + b_{\boldsymbol{i}}y + c_{\boldsymbol{i}}z + d_{\boldsymbol{i}}$, where the coefficients are as in Equation (2). Let $\Delta x, \Delta y, \Delta z > 0$ denote the grid sample spacing along the X, Y, Z axes, respectively, and let $\boldsymbol{o} \in \mathbb{E}^3$ be the origin of the distance field (in the sense that it contains the smallest x, y, z coordinates). Then the position of sample $\boldsymbol{i}$ is $\boldsymbol{x_i} = \boldsymbol{o} + [i\cdot\Delta x, j\cdot\Delta y, k\cdot\Delta z]^T$.

Since our input is assumed to be a piece-wise linear approximation of a smooth surface, we cannot rely on the analytic partial derivatives of the exact signed distance function. Even if such was obtainable for a particular geometry, it is still a debatable choice given that our construction has to adapt to arbitrary grid spacing. As such, merely taking the signed distance function of the closest triangle does not suffice, since it may convey locally too limited information about the SDF in relation to the grid size.

Instead, we propose to use a finer grid about $\boldsymbol{x_i}$, denoted by $\boldsymbol{s_{ij}} = \boldsymbol{x_i} + [i\cdot\Delta a, j\cdot\Delta b, k\cdot\Delta c]^T$, $\boldsymbol{j} = (i,j,k) \in \{-H,\dots,H\}^3$. Once we obtain an $\left\{f(\boldsymbol{s_{ij}}) \middle| \boldsymbol{j} \in \{-H,\dots,H\}^3\right\}$ set of distance samples, we fit a plane to them in the distance space as

$$\underbrace{\begin{bmatrix}1 & x_1 & y_1 & z_1\\ 1 & x_2 & y_2 & z_2\\ \vdots & \vdots & \vdots & \vdots\\ 1 & x_N & y_N & z_N\end{bmatrix}}_{\boldsymbol{X}} \cdot \underbrace{\begin{bmatrix}a\\ b\\ c\\ d\end{bmatrix}}_{\boldsymbol{a}} \approx \underbrace{\begin{bmatrix}f_1\\ f_2\\ \vdots\\ f_N\end{bmatrix}}_{\boldsymbol{f}} \qquad (3)$$

where we have flattened the three dimensional arrays $\boldsymbol{s_{ij}}$ and $f_{\boldsymbol{j}}$ into $[x_1,y_1,z_1],[x_2,y_2,z_2],\dots$ and $[f_1,f_2,\dots]$, respectively. The extent of the fine grid can be considered as a level-of-detail control.

The least-squares solution to (3) is $\boldsymbol{a} = \boldsymbol{X}^+\cdot\boldsymbol{f}$, where $\boldsymbol{X}^+$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{X}$. If the column vectors of $\boldsymbol{X}$ are linearly independent, it can be expressed as $\boldsymbol{X}^+ = (\boldsymbol{X}^T\cdot\boldsymbol{X})^{-1}\cdot\boldsymbol{X}^T$. In addition, one could also incorporate weighting into the fitting process to reduce the significance of far-away samples. Note that the samples need not be taken on a regular grid.

If the same fine grid structure is used for every $\boldsymbol{s_{ij}}$, $\boldsymbol{X}^+$ only needs to be pre-computed in the origin centered configuration, denoted by $\boldsymbol{X_o^+}$. Using $\boldsymbol{X_o^+}$ at $\boldsymbol{x_i}$ instead of the local matrices simply translates the world origin to $\boldsymbol{x_i}$ in Equation (3). To obtain the global world space coefficients of the Taylor polynomials we merely have to factor $a\cdot(x-x_i)+b\cdot(y-y_i)+c\cdot(z-z_i)+d$ in terms of the global power basis as $a\cdot x+b\cdot y+c\cdot z+d-a\cdot x_i-b\cdot y_i-c\cdot z_i$.

Algorithm 1 and Figure 1 summarize the above construction.

```
In  : Input geometry; $\boldsymbol{o} \in \mathbb{E}^3$ origin; $N,M,K \in \mathbb{N}^+$
      resolution; $\Delta x,\Delta y,\Delta z > 0$ grid sizes; $\boldsymbol{s_{oj}}$ origin
      centered fine grid with $\Delta a,\Delta b,\Delta c > 0$ sizes,
      $\boldsymbol{j} \in \{-H,\dots,H\}^3$; $H \in \mathbb{N}^+$; $\boldsymbol{X_o^+}$ pseudoinverse
Out: $(a_{\boldsymbol{i}},b_{\boldsymbol{i}},c_{\boldsymbol{i}},d_{\boldsymbol{i}})$, $\boldsymbol{i} \in [1,N]\times[1,M]\times[1,K]$ linear DF
for all $\boldsymbol{i} \in [1,N]\times[1,M]\times[1,K]$ do
    1. Compute sample location $\boldsymbol{x_i} \leftarrow \boldsymbol{o} + [i\cdot\Delta x, j\cdot\Delta y, k\cdot\Delta z]^T$
    2. Sample distance function on translated fine grid
         $f_{\boldsymbol{j}} \leftarrow f(\boldsymbol{s_{ij}}) = f(\boldsymbol{x_i}+\boldsymbol{s_{oj}})$, $\boldsymbol{j} \in \{-H,\dots,H\}^3$
    3. Flatten fine grid samples into $\boldsymbol{f} \leftarrow [f_1,\dots,f_{(2H+1)^3}]^T$
    4. Compute linear approximation $\tilde{\boldsymbol{a}} \leftarrow \boldsymbol{X_o^+}\cdot\boldsymbol{f}$
    5. Translate $\tilde{\boldsymbol{a}}$ to $\boldsymbol{x_i}$: $\tilde{\boldsymbol{a}}_{\boldsymbol{i}} \leftarrow (\tilde{a},\tilde{b},\tilde{c},\tilde{d}-[\tilde{a},\tilde{b},\tilde{c}]\cdot\boldsymbol{x_i})$
    6. Normalize for final sample $\boldsymbol{a_i} \leftarrow \tilde{\boldsymbol{a}}_{\boldsymbol{i}}/\sqrt{\tilde{a}_{\boldsymbol{i}}^2+\tilde{b}_{\boldsymbol{i}}^2+\tilde{c}_{\boldsymbol{i}}^2}$
end
```

Algorithm 1: *Uniform linear distance field construction.*

Trilinear filtering of a first order field first computes the trilinear interpolation between the $a_{\boldsymbol{i}}, b_{\boldsymbol{i}}, c_{\boldsymbol{i}}, d_{\boldsymbol{i}}$ coefficients of the 8 closest Taylor approximations, then evaluates the resulting polynomial using the coordinates of the query position $\boldsymbol{x}$ in (2). Nearest neighbor filtering fetches and evaluates the closest first order approximation.

5. Results

We implemented order zero and one distance fields in a C++ OpenGL framework. All distance fields consisted of equidistant

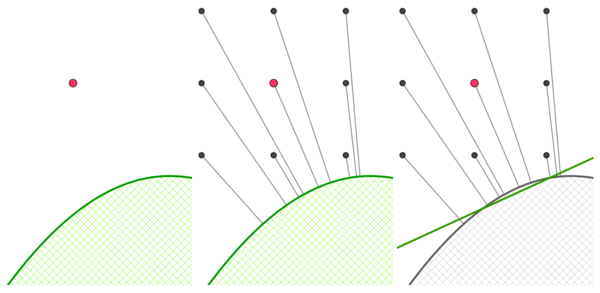

Figure 1: *The first order grid sample $\mathbf{x_i}$ (red point on left) uses distance samples on the fine grid (center) to fit a plane (right).*

samples. We refer to the first order distance fields as Taylor distance fields (TDF) in the tables below. The input geometries were normalized to the $[-1,1]^3$ region of space. The performance tests were carried out on an NVIDIA GeForce 1070 Ti and 2080 at Full HD resolution.

A direct visualization scenario was used for the distance field query. We rendered the given input distance field with sphere tracing followed by 10 binary search root refinement steps.

The tests were performed on an analytic test scene consisting of a sphere and two scenes of triangular meshes, a bunny and a dragon model, see Figure 2. Generating a sign-correct distance field from arbitrary triangular meshes is in itself a challenging problem. We used Krayer and Müller's [KM19] application to generate a high resolution ($256 \times 256 \times 256$), zero order distance field of the meshes for ground truth and a base for higher order constructions.

5.1. Fine grid resolution and extent

There are two degrees of freedom in the fine grid specification: (i) the $(2H+1)^3$ number of samples in the fine grid and (ii) the $\Delta a, \Delta b, \Delta c > 0$ fine grid spacing along the X, Y, and Z axes.

First, we measured how fine grid sample counts affect TDF accuracy. We varied samples from 2^3 to 9^3. Increasing sample counts had a very minor error mitigation effect. We have chosen 3^3 samples ($H = 1$) as a go-to fine grid resolution.

Then we measured how the extent of the fine grid alters the resulting TDF. Fine grid spacing was measured relative to the distance to the next sample, e.g. $\Delta a = 1$ means that the fine grid samples reach the neighboring sample position along the X axis. According to our tests, the larger the span ($\Delta a, \Delta b, \Delta c$), the larger the error becomes. We chose $\Delta := \Delta a = \Delta b = \Delta c = 0.1$ as our default, which resulted in a common local minimum error for all test scenes.

5.2. Accuracy

We compared the accuracy of zero and first order distance fields of the same resolution, the latter being always the more accurate.

SDF	sphere error		bunny error		dragon error	
dim	0th	1st	0th	1st	0th	1st
8^3	0.0299	0.0061	0.029	0.017	0.027	0.016
16^3	0.0149	0.0010	0.015	0.0037	0.014	0.0044
32^3	0.0074	0.0002	0.0074	0.0018	0.0074	0.0020
64^3	0.0036	0.0001	0.0036	0.0011	0.0037	0.0015
128^3	0.0017	0.00005	0.0017	0.00097	0.0017	0.0014

Accuracy tests resampled lower resolution (8^3 to 128^3, see table above) zero and first order SDFs to a 256^3 zero order distance field that coincided with our ground truth. Upscaling used nearest neighbor resampling. We computed the mean error on this high resolution grid. Errors are the absolute differences between the resampled distance values and the 256^3 ground truth field values.

The next set of tests focused on finding the zero order SDF sizes that matched the error of a given resolution first order Taylor DF. The first row of the following table shows the size of the first order distance field for the three models; the second to fourth the size of the smallest zero order SDF matching the error. Empty entries denote no matching zero order SDF up to 128^3. The TDF of resolution 51^3 was of higher accuracy than any 128^3 zero order SDF.

TDF	8^3	10^3	12^3	16^3	20^3	25^3	32^3	40^3
sphere	40^3	64^3	102^3					
bunny	16^3	40^3	51^3	64^3	81^3	102^3	128^3	
dragon	16^3	20^3	32^3	64^3	81^3	102^3	128^3	128^3

5.3. Storage

The naive approach for storing first order data is to use 4 floating point numbers for a_i, b_i, c_i, d_i, making a first order sample four times as expensive in storage. Nevertheless, when comparing zero and first order storage of the same accuracy, TDF can be up to an order of magnitude more efficient, as summarized below.

1st dim	scalars	0th dim	scalars	1st order storage %
8^3	2048	16^3	4096	50%
10^3	4000	20^3	8000	50%
12^3	6912	32^3	32768	21.09%
16^3	16384	64^3	262144	6.25%
20^3	32000	81^3	531441	6.02%
25^3	62500	102^3	1061208	5.89%
32^3	131072	128^3	2097152	6.25%

We used the worst case for the first order TDF, i.e. the row of the dragon. Even so, on the most relevant resolutions where fine details emerge (16^3 onwards), the first order TDF only requires about 6% of the scalar storage of the zero order SDF for the same accuracy.

Due to the relatively small metric extent of our test models, it did not matter whether we used 16 or 32 bit floats in the implementation. The precision of both the zero and first order distance fields remained the same statistically.

Storage can be further optimized for first order TDFs. Since the linear coefficients form a unit vector, we can encode them with octahedral mapping [MSS*10] and pack all the a_i, b_i, c_i, d_i in 32 bits. We tested several distributions of bits and a clear optimum for all three test models was the $10+10+12$ distribution, i.e. 10 bits for each of the two octahedral coordinates and 12 bits for the constant term in snorm. Packing introduces error to the approximation compared to the unpacked field; the error is at most 1% larger for smaller resolutions (up to 20^3) and grows to 28% at higher resolutions (128^3), due to the reduced floating point precision. In exchange, TDF storage requirements are halved by this technique.

5.4. Performance

The distance fields were trilinearly filtered for performance tests. Although our proposed construction is compatible with hardware

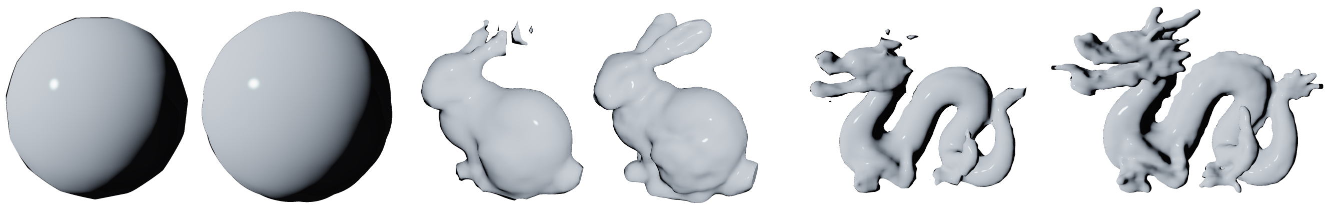
(a) *0th order* 8^3 **(b)** *1st order* 8^3 **(c)** *0th order* 16^3 **(d)** *1st order* 16^3 **(e)** *0th order* 32^3 **(f)** *1st order* 32^3

Figure 2: *Comparison between zero order and first order SDFs on different models. Note the additional detail of first order fields.*

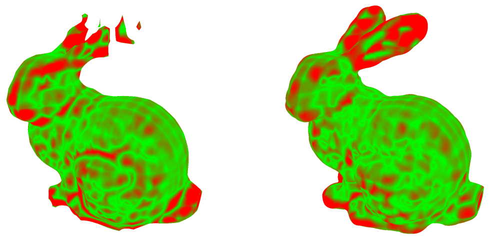
(a) *Zero order SDF* **(b)** *First order SDF*

Figure 3: *Error comparison between a zero order and a first order* 16^3 *distance field. Small absolute error in green, large in red.*

filtering, it must be noted that certain applications may not be able to take advantage of this due to the limited precision in which texture filtering is handled in hardware [NVI19]. In such cases, interpolation has to be implemented manually in the shaders. Similarly, packing also forces manual fetch, decode, and interpolation of data.

The following table summarizes the 100-frame average render times in milliseconds. The fourth column contains the relative increase of render times using the same resolution first order DF instead of the zero order field.

sphere	0th	1st	perf
1070 Ti	1.25614 ms	1.55074 ms	123.45%
2080	0.716773 ms	0.916037 ms	127.80%
bunny	0th	1st	perf
1070 Ti	1.16 ms	1.30086 ms	112.32%
2080	0.64 ms	0.718491 ms	112.71%
dragon	0th	1st	perf
1070 Ti	1.19 ms	1.41792 ms	118.93%
2080	0.617716 ms	0.717756 ms	116.20%

The standard deviation of render times was in the order of 10^{-5} for all configurations. In general, using the first order Taylor distance field adds 12-28% to the render times. However, using the same accuracy but lower resolution TDF, this difference shrinks to an up to 10% performance hit. The following table compares render times (ms) of zero and first order SDFs with similar accuracy.

0th	0.53 (16^3)	0.53 (20^3)	0.55 (32^3)	0.57 (64^3)	0.6 (81^3)
1st	0.56 (8^3)	0.58 (10^3)	0.59 (12^3)	0.62 (16^3)	0.66 (20^3)
perf	105.66%	109.43%	107.27%	108.77%	110.00%

Packing introduced a $2-5$ times overhead compared to non-packed, hardware interpolated figures.

6. Conclusions

This paper presented a first degree Taylor approximation based representation of signed distance fields. Storing higher order approximations proved to be a more efficient way to utilize the available bits of storage than the brute-force increase of lower order samples. These approximations have to be represented in a global basis, such as the global power basis to use GPU hardware interpolation.

We proposed a general first order signed distance field construction algorithm that can be applied in an arbitrary geometric context. Storage can be further optimized by using octahedral encoding.

Our empirical tests demonstrated that storage could be reduced down to 6% of the original zero order SDF size while retaining the same accuracy. The trade-off for this increase in storage efficiency is an increase of at most 10% in direct render times. Moreover, our Taylor-based construction can be applied to the approximation of arbitrary functions, it is not restricted to signed distance functions.

Acknowledgement The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

References

[Aal] Aaltonen S.: GPU-based clay simulation and ray-tracing tech in Claybook, Game Developers Conference 2018. 1

[FPRJ00] 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* (2000), SIGGRAPH, pp. 249–254. 1

[Gre07] Green C.: Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 Courses* (2007), SIGGRAPH '07, pp. 9–18. 1

[KDB16] Koschier D., Deul C., Bender J.: Hierarchical hp-adaptive signed distance fields. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016), SCA '16, pp. 189–198. 1

[KM19] Krayer B., Müller S.: Generating signed distance fields on the gpu with ray maps. *Vis. Comput. 35*, 6-8 (June 2019), 961–971. 3

[MSS*10] Meyer Q., Suessmuth J., Sussner G., Stamminger M., Greiner G.: On Floating-Point Normal Vectors. *Computer Graphics Forum* (2010). 3

[NVI19] NVIDIA Corporation: CUDA C++ Programming Guide; G.2. Linear Filtering, 2019. CUDA Toolkit v10.2.89. 4

[Wri15] Wright D.: Dynamic occlusion with signed distance fields. In *Advances in Real-Time Rendering in Games* (2015), Epic Games (Unreal Engine), SIGGRAPH. 1