





Hermite Interpolation of Scalar Fields in Computer Graphics

Róbert Bán¹ , Gábor Valasek² 

¹Eötvös Loránd University, Hungary, rob.ban@inf.elte.hu

²Eötvös Loránd University, Hungary, valasek@inf.elte.hu

Corresponding author: Róbert Bán, rob.ban@inf.elte.hu

Abstract. Scalar valued functions, such as height- and signed distance fields, are essential in real-time computer graphics. Depending on dimensionality, these are represented by function sample values stored on a regular grid that are bi- or trilinearly filtered, resulting in C^0 approximations. First, we propose to store the partial derivatives of the scalar valued function with the function values and use Hermite interpolation between the samples. This guarantees a globally C^1 -continuous result. For rendering applications, the surface normal vectors are often part of the discrete field of samples, as such, our technique does not necessarily require extra storage, merely a different basis to store the same data. The exact normals of the reconstructed cubic Hermite surface can be used as shading normals, resulting in a storage efficient replacement for normal mapping with richer visual appearance. We show that our method generalizes to arbitrary orders and dimensions. Moreover, we derive an approximation for mixed partial derivatives for three dimensional first order fields, akin to Adini's method for parametric surface patches. We demonstrate the applicability of Hermite interpolation in height field and signed distance field rendering.

Keywords: computer graphics, parallax mapping, geometric modeling, Hermite interpolation

DOI: <https://doi.org/10.14733/cadaps.2025.927-946>

1 INTRODUCTION

A common task in computer graphics is to store an arbitrary real valued continuous function. The usual discrete representation of such $f \in \mathbb{R}^n \rightarrow \mathbb{R}$ scalar functions consist of samples on a regular cubic grid inside an axis aligned bounding box. The samples contain the function value $f(x)$ at the sample position $x \in \mathcal{D}_f$. Upon sampling, we want to reconstruct the function value at an arbitrary position inside the described volume, which we can achieve by interpolating between the stored function values. The simplest interpolation is tensor product linear interpolation, i.e. linear, bilinear, trilinear interpolation in 1, 2, and 3 dimensions. This guarantees a C^0 -continuous reconstruction.

However, the smoothness of the stored functions is often important for the application. To obtain a sufficiently smooth interpolated output, one might need a grid with very high resolution. Higher order continuity may be achieved by incorporating a larger sampling footprint [14], however, this comes with an increased execution time. Our goal is reconstruct with higher order continuity by modifying the samples and without modifying the filtering footprint size. To achieve this, we store the $\nabla f(x)$ gradient in addition to the function value $f(x)$.

In terms of analytical properties, a higher order interpolation scheme also provides a more accurate approximation to an underlying continuous function. As such, a first order method that uses Hermite interpolation is capable of representing a shape with a given error using less data than a traditional, tensor product multilinear filtered field. Our paper focuses on defining the interpolation method with proven interpolation properties, and the qualitative properties of such reconstructions in the case of height field and signed distance field rendering, with a secondary focus on render efficiency.

In Section 2 we give an overview of the related literature. Section 3 derives the one dimensional Hermite polynomials for solving the interpolation problem, and proves their interpolation properties, which are then expanded to multiple dimensions in Section 4. Section 5 solves the problem of interpolation to a grid of higher order samples. Finally in Section 6 we show applied results in height map rendering and three-dimensional surface representations using signed distance fields, including an approximation for mixed partial derivatives in the three dimensional first order case.

2 PREVIOUS WORK

Texture filtering is an important step in computer graphics applications. Its main purpose is to define what values are read from a texture at a given sampling position. The two main sampling methods supported by GPU hardware acceleration are nearest neighbour filtering and linear (bilinear, trilinear) interpolation. The former results in discontinuous reconstruction, while the latter guarantees C^0 -continuity. Other reconstruction methods are achieved by fetching the samples and doing further arithmetic “by hand.” Combining multiple linearly filtered samples can yield higher continuity [14, 5]. The drawback of these higher order interpolation methods is the increased filtering footprint, i.e. they require more samples for a single evaluation. We aim to keep the footprint the same as it is with linear filtering – the samples at the vertices of the surrounding cell. The Hermite interpolation problem and its solution in one dimension is discussed in many places in literature, for example by Farin [8]. We derive the one-dimensional results using our notation, which we then generalize to higher dimensions.

Height maps are important tools for realistic rendering. They contain geometric information in the form of perturbation to a coarser surface representation, resulting in a high detail surface. There are many algorithms and techniques for rendering these surfaces, Szirmay-Kalos et al. give an overview [15]. Cone step mapping [6] and relaxed conemaps [13] present an efficient way of resolving the view ray–surface intersections using additional information stored besides the height values. Our higher order interpolation method can be used alongside these methods either by incorporating it in the intersection search, or only to the shading stage – calculating the normal vector –, resulting in very low additional computation cost with a great improvement in rendering quality.

Signed distance functions and fields are used in computer graphics as either an auxiliary data structure for graphical effects like ambient occlusion or soft shadows, or they can be the primary representation of the surface. Current hardware is able to render these surfaces using sphere tracing [11] and its variants [12, 2], leveraging the geometrical meaning of the function values. Signed distance fields are also used in two dimensions for scalable high quality font rendering [10]. For two and three-dimensional surfaces [9] gives an adaptive approach to minimizing storage using quadtree and octree structures. Three dimensional signed distance fields are used in many applications for real-time graphics [1, 7, 16].

3 HERMITE INTERPOLATION IN ONE DIMENSION

3.1 The Hermite Interpolation Problem

In one dimension, the order n Hermite interpolation problem ($n \in \mathbb{N}$) can be stated as follows. Given $f_0^0, f_1^0, f_0^1, f_1^1, \dots, f_0^n, f_1^n \in \mathbb{R}$, find a polynomial p of degree $2n + 1$ such that

$$\forall x \in \{0, 1\}: p(x) = f_x^0, \quad p'(x) = f_x^1, \quad p''(x) = f_x^2, \quad \dots, \quad p^{(n)}(x) = f_x^n. \quad (1)$$

The problem can be solved by looking for the solution as a polynomial with unknown coefficients, and calculating its derivatives.

$$\begin{aligned} p(x) &= a_0 + a_1x + a_2x^2 + \dots + a_{2n+1}x^{2n+1} \\ p'(x) &= a_1 + 2a_2x + 3a_3x^2 + \dots + (2n+1)a_{2n+1}x^{2n} \\ p''(x) &= 2a_2 + 6a_3x + 12a_4x^2 + \dots + (2n+1)(2n)a_{2n+1}x^{2n-1} \\ &\vdots \end{aligned}$$

Then we substitute these polynomials into the equations, resulting in a linear system of equations with $2n + 2$ unknowns and $2n + 2$ equations. For a fixed n , the matrix of the system is always the same, and it has full rank, meaning that there is a unique solution. Let $B \in \mathbb{R}^{(2n+2) \times (2n+2)}$ be the matrix of the linear system, $\mathbf{a} = [a_0, a_1, \dots, a_{2n+1}]^T$ the unknown coefficient vector, and $\mathbf{f} = [f_0^0, f_1^0, \dots, f_0^n, f_1^n]^T$ the vector of values to interpolate. Then the matrix equation and solution are

$$B\mathbf{a} = \mathbf{f} \quad \Rightarrow \quad \mathbf{a} = B^{-1}\mathbf{f}. \quad (2)$$

The problem can also be solved using the order n Hermite polynomials which we describe next.

3.2 One Dimensional Hermite Polynomials

Let $n \in \mathbb{N}$ be the maximal order. We define the order n Hermite polynomials through a set of equations.

Definition 1. The order n Hermite polynomial of base position x interpolating the k th derivative $k \in \{0, 1, \dots, n\}$ and $x \in \{0, 1\}$, is $\alpha_x^{k,n} : \mathbb{R} \rightarrow \mathbb{R}$. The defining equations for $\alpha_x^{k,n}$ are

$$\forall j \in \{0, 1, \dots, n\} : \forall y \in \{0, 1\} : (\alpha_x^{k,n})^{(j)}(y) = \delta_{k,j} \delta_{x,y}, \quad (3)$$

where $f^{(j)}$ is the j th derivative of f and

$$\delta_{a,b} = \begin{cases} 1, & a = b, \\ 0, & a \neq b. \end{cases} \quad (4)$$

This means that the values and the derivatives of $\alpha_x^{k,n}$ are zero at 0 and 1 except for the k th derivative at x , where it is 1. The number of order n Hermite polynomials is therefore $2n + 2$ – one for each interpolated position and derivative up to n . As there are $2n + 2$ equations for each polynomial, there is a unique polynomial of degree $2n + 1$ satisfying all requirements for each (k, n, x) triplet. We can calculate the polynomials the same way as we did for the general Hermite interpolation problem in Section 3.1.

For example, the order 0 Hermite polynomials are defined by the following equations.

$$\begin{aligned} \alpha_0^{0,0}(0) &= 1, & \alpha_1^{0,0}(0) &= 0, \\ \alpha_0^{0,0}(1) &= 0, & \alpha_1^{0,0}(1) &= 1. \end{aligned}$$

We look for the solution in the form of linear polynomials, i.e.

$$\alpha_0^{0,0}(x) = a_{00} + a_{01}x \quad \text{and} \quad \alpha_1^{0,0}(x) = a_{10} + a_{11}x$$

After substitution, the equations become

$$\begin{aligned} a_{00} + a_{01} \cdot 0 &= 1, & a_{10} + a_{11} \cdot 0 &= 0, \\ a_{00} + a_{01} \cdot 1 &= 0, & a_{10} + a_{11} \cdot 1 &= 1. \end{aligned}$$

Solving them we get the order 0 Hermite polynomials as

$$\alpha_0^{0,0}(x) = 1 - x \quad \text{and} \quad \alpha_1^{0,0}(x) = x.$$

The order 1 Hermite cubic polynomials are

$$\begin{aligned} \alpha_0^{0,1}(x) &= 1 - 3x^2 + 2x^3, & \alpha_1^{0,1}(x) &= 3x^2 - 2x^3, \\ \alpha_0^{1,1}(x) &= x - 2x^2 + x^3, & \alpha_1^{1,1}(x) &= -x^2 + x^3. \end{aligned}$$

The order 2 Hermite quintic polynomials are

$$\begin{aligned} \alpha_0^{0,2}(x) &= 1 - 10x^3 + 15x^4 - 6x^5, & \alpha_1^{0,2}(x) &= 10x^3 - 15x^4 + 6x^5, \\ \alpha_0^{1,2}(x) &= x - 6x^3 + 8x^4 - 3x^5, & \alpha_1^{1,2}(x) &= -4x^3 + 7x^4 - 3x^5, \\ \alpha_0^{2,2}(x) &= \frac{1}{2}x^2 - \frac{3}{2}x^3 + \frac{3}{2}x^4 - \frac{1}{2}x^5, & \alpha_1^{2,2}(x) &= \frac{1}{2}x^3 - x^4 + \frac{1}{2}x^5. \end{aligned}$$

3.3 Interpolation Using Hermite Polynomials

The order n Hermite polynomials may be used as a polynomial basis for solving the interpolation problem stated in Section 3.1. The basis function coefficients are exactly the given parameters – function values and derivatives – which makes the basis extremely useful.

Theorem 1. *The following polynomial solves the interpolation problem in Equation (1)*

$$p(x) = \sum_{i=0}^1 \sum_{k=0}^n f_i^k \cdot \alpha_i^{k,n}(x). \quad (5)$$

Proof. We only need to check whether p interpolates the given values and derivatives in Equation (1). This follows as

$$\begin{aligned} \forall j \in \{0, 1\}: \forall \ell \in \{0, 1, \dots, n\}: \\ p^{(\ell)}(j) &\stackrel{(5)}{=} \left(\sum_{i=0}^1 \sum_{k=0}^n f_i^k \cdot \alpha_i^{k,n}(j) \right)^{(\ell)} \\ &= \sum_{i=0}^1 \sum_{k=0}^n f_i^k \cdot \left(\alpha_i^{k,n} \right)^{(\ell)}(j) \\ &\stackrel{(3)}{=} \sum_{i=0}^1 \sum_{k=0}^n f_i^k \cdot \delta_{i,j} \cdot \delta_{k,\ell} \stackrel{(4)}{=} f_j^\ell \end{aligned}$$

In the last step the only non-zero term is when both δ 's are one, i.e. $i = j$ and $k = \ell$. □

3.4 Interpolation on a General Interval

The problems stated in Section 3.1 and 4.1 can be generalized to any interval, and their solution can be written in terms of the defined Hermite polynomials. Sadly, unlike the Bernstein-basis, Hermite polynomials are not invariant under affine parameter transformations. The problem is as follows. Let $n \in \mathbb{N}$, $a, b \in \mathbb{R}$, $a < b$, and given $f_a^0, f_b^0, f_a^1, f_b^1, \dots, f_a^n, f_b^n \in \mathbb{R}$, find a polynomial p of degree $2n + 1$ such that

$$\forall x \in \{a, b\}: \forall k \in \{0, 1, \dots, n\}: p^{(k)}(x) = f_x^k. \quad (6)$$

The solution is unique as before. The $a = 0, b = 1$ case is equivalent to the original wording. Let us now define the Hermite polynomials for the new interval.

Definition 2. Let $n \in \mathbb{N}, a, b \in \mathbb{R}, a < b, k \in \{0, 1, \dots, n\}, x \in \{a, b\}$.

$$\beta_{x,[a,b]}^{k,n}(y) := (b-a)^k \cdot \alpha_{\hat{x}}^{k,n}(s_a^b(y)), \quad (7)$$

where $\hat{x} = 0$ if $x = a$ and $\hat{x} = 1$ if $x = b$; and s_a^b is the linear mapping from $[a, b]$ to $[0, 1]$:

$$s_a^b(x) = \frac{x-a}{b-a}. \quad (8)$$

Note that $s_a^b(a) = 0$ and $s_a^b(b) = 1$, therefore $\hat{x} = s_a^b(x)$. These polynomials solve the interpolation problem similarly as before. To show that, first we verify the interpolation property of the polynomials.

Theorem 2. Let $n \in \mathbb{N}, a, b \in \mathbb{R}, a < b, k \in \{0, 1, \dots, n\}, x \in \{a, b\}$.

$$\forall y \in \{a, b\}, \forall \ell \in \{0, 1, \dots, n\}: \left(\beta_{x,[a,b]}^{k,n} \right)^{(\ell)}(y) = \delta_{x,y} \cdot \delta_{k,\ell}. \quad (9)$$

Proof. Let $y \in \{a, b\}, \ell \in \{0, 1, \dots, n\}, \hat{x} := s_a^b(x)$.

$$\begin{aligned} \left(\beta_{x,[a,b]}^{k,n} \right)^{(\ell)}(y) &\stackrel{(7)}{=} \left((b-a)^k \cdot \alpha_{\hat{x}}^{k,n}(s_a^b(y)) \right)^{(\ell)} \\ &= (b-a)^k \cdot \left(\alpha_{\hat{x}}^{k,n} \left(\frac{y-a}{b-a} \right) \right)^{(\ell)} \\ &= (b-a)^k \cdot \left(\frac{y-a}{b-a} \right)' \cdot \left(\left(\alpha_{\hat{x}}^{k,n} \right)' \left(\frac{y-a}{b-a} \right) \right)^{(\ell-1)} \\ &= (b-a)^{k-1} \cdot \left(\left(\alpha_{\hat{x}}^{k,n} \right)' \left(\frac{y-a}{b-a} \right) \right)^{(\ell-1)} \\ &= (b-a)^{k-2} \cdot \left(\left(\alpha_{\hat{x}}^{k,n} \right)'' \left(\frac{y-a}{b-a} \right) \right)^{(\ell-2)} = \dots \\ &= (b-a)^{k-\ell} \cdot \left(\alpha_{\hat{x}}^{k,n} \right)^{(\ell)} \left(\frac{y-a}{b-a} \right) \\ &\stackrel{(3)}{=} (b-a)^{k-\ell} \cdot \delta_{\hat{x}, s_a^b(y)} \cdot \delta_{k,\ell} \\ &\stackrel{(8)}{=} \delta_{x,y} \cdot \delta_{k,\ell}. \end{aligned}$$

□

Theorem 3. *The following polynomial solves the interpolation problem, i.e. it satisfies Equation (6),*

$$p(x) = \sum_{i \in \{a,b\}} \sum_{k=0}^n f_i^k \cdot \beta_{i,[a,b]}^{k,n}(x). \quad (10)$$

The proof is identical to the proof of Theorem 1, replacing $\alpha_i^{k,n}$ by $\beta_{i,[a,b]}^{k,n}$.

4 HIGHER DIMENSIONAL HERMITE INTERPOLATION

We define the solution to the multidimensional Hermite interpolation problem in terms of the one-dimensional solutions. The higher dimensional multivariate Hermite polynomials are written as the tensor product of the one-dimensional polynomials.

In the higher dimensional cases, the equations become very hard to read without additional notational aid, so let us define multi-indices. An $\mathbf{i} = (i_1, i_2, \dots, i_r) \in \mathbb{N}^r$ r -dimensional multi-index is a tuple with r elements. We define the following operations on multi-indices

$$\begin{aligned} |\mathbf{i}| &:= i_1 + i_2 + \dots + i_r, \\ \mathbf{i}! &:= i_1! \cdot i_2! \cdot \dots \cdot i_r!, \\ \mathbf{x}^{\mathbf{i}} &:= x_1^{i_1} \cdot x_2^{i_2} \cdot \dots \cdot x_r^{i_r}, \\ \partial^{\mathbf{i}} f &:= \partial_1^{i_1} \partial_2^{i_2} \dots \partial_r^{i_r} = \frac{\partial^{|\mathbf{i}|} f}{\partial x_1^{i_1} \partial x_2^{i_2} \dots \partial x_r^{i_r}}, \\ \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{1}} &:= \sum_{\mathbf{i} \in \{0,1\}^r}, \\ \sum_{|\mathbf{i}|=0}^n &:= \sum_{\mathbf{i} \in \{\ell \in \mathbb{N}^r : |\ell| \leq n\}}. \end{aligned}$$

4.1 The Interpolation Problem in r Dimensions

We generalize the interpolation problem from the one-dimensional $[0, 1]$ interval to the r -dimensional $[0, 1]^r$ hypercube. We state the problem as the following. Let $n \in \mathbb{N}$ be the maximal order, and given

$$\mathbf{i} \in \{0, 1\}^r, \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n: f_{\mathbf{i}}^{\mathbf{k}} \in \mathbb{R}$$

values, find a multivariate polynomial $\tilde{f}_n: \mathbb{R}^r \rightarrow \mathbb{R}$ such that

$$\forall \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n: \forall \mathbf{x} \in \{0, 1\}^r: \partial^{\mathbf{k}} \tilde{f}_n(\mathbf{x}) = f_{\mathbf{x}}^{\mathbf{k}}. \quad (11)$$

We give a constructive solution to the problem. The solution includes the one-dimensional Hermite polynomials, and it has a similar structure: we construct basis polynomials that generalize the property in Equation (3) to the multivariate case.

4.2 Multivariate Hermite Polynomials

Definition 3. *Let $n \in \mathbb{N}$, $\mathbf{k} = (k_1, k_2, \dots, k_n) \in \mathbb{N}^r$, $|\mathbf{k}| \leq n$, $\mathbf{j} = [j_1, j_2, \dots, j_n]^T \in \{0, 1\}^r$. The multivariate Hermite basis polynomials are*

$$\alpha_{\mathbf{j}}^{\mathbf{k},n}(\mathbf{x}) := \prod_{i=1}^r \alpha_{j_i}^{k_i,n}(x_i) \quad (\mathbf{x} = [x_1, x_2, \dots, x_r]^T \in \mathbb{R}^r). \quad (12)$$

Theorem 4. The multivariate Hermite polynomial $\alpha_x^{k,n}$ with $x \in \{0, 1\}^r$, $k \in \mathbb{N}^r$, $|k| \leq n$ has the following interpolation property:

$$\forall \ell \in \mathbb{N}^r, |\ell| \leq n, \forall y \in \{0, 1\}^r: \partial^\ell \alpha_x^{k,n}(y) = \delta_{k,\ell} \cdot \delta_{x,y}. \quad (13)$$

Proof. Let $k, \ell \in \mathbb{N}^r$, $|k|, |\ell| \leq n$, $x, y \in \{0, 1\}^r$.

$$\partial^\ell \alpha_x^{k,n}(y) \stackrel{(12)}{=} \partial^\ell \prod_{i=1}^r \alpha_{x_i}^{k_i,n}(y_i) = \prod_{i=1}^r (\alpha_{x_i}^{k_i,n})^{(\ell_i)}(y_i) \stackrel{(3)}{=} \prod_{i=1}^r \delta_{k_i,\ell_i} \cdot \delta_{x_i,y_i} = \delta_{k,\ell} \cdot \delta_{x,y}$$

□

4.3 Hermite Interpolation in r Dimensions

Let $n \in \mathbb{N}$ denote the fixed maximal order, and let us define the interpolating polynomials using the multivariate Hermite basis .

Theorem 5. Given the function values and derivatives in the vertices $f_y^k \in \mathbb{R}$, where $y \in \{0, 1\}^r$ and $k \in \mathbb{N}^r$, $|k| \leq n$, the following \tilde{f}_n polynomial

$$\tilde{f}_n(x) := \sum_{i=0}^1 \sum_{|k|=0}^n f_i^k \cdot \alpha_i^{k,n}(x) \quad (x \in [0, 1]^r) \quad (14)$$

solves the multivariate interpolation problem stated in Equation (11).

Proof. Let $k, \ell \in \mathbb{N}^r$, $|k|, |\ell| \leq n$, $i, j \in \{0, 1\}^r$.

$$\begin{aligned} \partial^\ell \tilde{f}_n(j) &\stackrel{(14)}{=} \partial^\ell \sum_{i=0}^1 \sum_{|k|=0}^n f_i^k \cdot \alpha_i^{k,n}(j) = \sum_{i=0}^1 \sum_{|k|=0}^n f_i^k \cdot \partial^\ell \alpha_i^{k,n}(j) \\ &\stackrel{(13)}{=} \sum_{i=0}^1 \sum_{|k|=0}^n f_i^k \cdot \delta_{k,\ell} \cdot \delta_{i,j} = f_j^\ell. \end{aligned}$$

□

Remark 1. The case $n = 0$ is also known as linear, bilinear, and trilinear interpolation in one, two, and three dimensions, respectively. Only the function value is given at the ends of the interval/vertices of the square/cube. Important note that in higher dimensions ($r > 1$), the resulting polynomial is not linear – despite what the names suggest.

The polynomial can be written as

$$\tilde{f}(x) = \sum_{i=0}^1 (1-x)^{1-i} x^i f_i = \sum_{i=0}^1 f_i \prod_{j=1}^r (1-x_j)^{1-i_j} x_j^{i_j}.$$

Here, \tilde{f} is linear in the sense that if we fix all but one of the variables, the resulting polynomial is linear with regards to the free variable. In each coefficient, either x_j or $1 - x_j$ appears exactly once because their exponents, $1 - i_j$ and i_j , are 0 and 1 in some order.

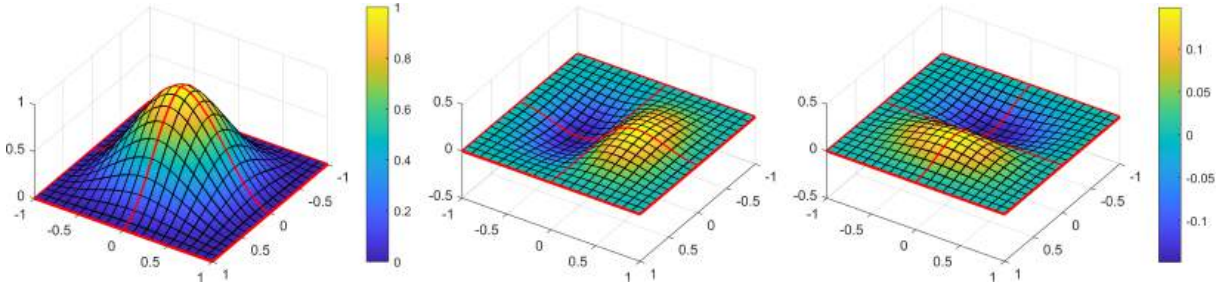


Figure 1: The 12 order 1 basis Hermite basis functions on $[0, 1]^2$ translated to the common interpolated value – left: function value, middle and right: partial derivative in the x and y directions.

Remark 2. As an example, the first order Hermite basis in the two-dimensional case has the following 12 polynomials. All of them are the products of two one-dimensional Hermite polynomial – the exact polynomials $\alpha_0^{0,1}, \alpha_1^{0,1}, \alpha_0^{1,1}$, and $\alpha_1^{1,1}$ are listed in Section 3.2. The functions are visualized in Figure 1.

The first four polynomials interpolate the function value at the vertices of $[0, 1]^2$:

$$\begin{aligned} \alpha_{(0,0)}^{(0,0),1}(x, y) &= \alpha_0^{0,1}(x) \cdot \alpha_0^{0,1}(y), & \alpha_{(1,0)}^{(0,0),1}(x, y) &= \alpha_1^{0,1}(x) \cdot \alpha_0^{0,1}(y), \\ \alpha_{(0,1)}^{(0,0),1}(x, y) &= \alpha_0^{0,1}(x) \cdot \alpha_1^{0,1}(y), & \alpha_{(1,1)}^{(0,0),1}(x, y) &= \alpha_1^{0,1}(x) \cdot \alpha_1^{0,1}(y). \end{aligned}$$

The rest interpolate the partial derivatives in the x or y directions.

$$\begin{aligned} \alpha_{(0,0)}^{(1,0),1}(x, y) &= \alpha_0^{1,1}(x) \cdot \alpha_0^{0,1}(y), & \alpha_{(1,0)}^{(1,0),1}(x, y) &= \alpha_1^{1,1}(x) \cdot \alpha_0^{0,1}(y), \\ \alpha_{(0,1)}^{(1,0),1}(x, y) &= \alpha_0^{1,1}(x) \cdot \alpha_1^{0,1}(y), & \alpha_{(1,1)}^{(1,0),1}(x, y) &= \alpha_1^{1,1}(x) \cdot \alpha_1^{0,1}(y), \\ \alpha_{(0,0)}^{(0,1),1}(x, y) &= \alpha_0^{0,1}(x) \cdot \alpha_0^{1,1}(y), & \alpha_{(1,0)}^{(0,1),1}(x, y) &= \alpha_1^{0,1}(x) \cdot \alpha_0^{1,1}(y), \\ \alpha_{(0,1)}^{(0,1),1}(x, y) &= \alpha_0^{0,1}(x) \cdot \alpha_1^{1,1}(y), & \alpha_{(1,1)}^{(0,1),1}(x, y) &= \alpha_1^{0,1}(x) \cdot \alpha_1^{1,1}(y). \end{aligned}$$

4.4 Interpolation on a General r -dimensional Interval

Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^r$ and $\mathbf{a} < \mathbf{b}$ (meaning, that $\forall i \in \{1, 2, \dots, r\}: a_i < b_i$). Let us define the r -dimensional interval $[\mathbf{a}, \mathbf{b}]$ and its vertices $V(\mathbf{a}, \mathbf{b})$ as

$$\begin{aligned} [\mathbf{a}, \mathbf{b}] &:= [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_r, b_r] \subset \mathbb{R}^r \quad \text{and} \\ V(\mathbf{a}, \mathbf{b}) &:= \{a_1, b_1\} \times \{a_2, b_2\} \times \dots \times \{a_r, b_r\} \subset \mathbb{R}^r. \end{aligned}$$

We define the general r -dimensional Hermite polynomials analogously to Definition 3. The interpolation property holds similarly.

Definition 4. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^r, \mathbf{a} < \mathbf{b}, n \in \mathbb{N}, \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n, \mathbf{j} \in V(\mathbf{a}, \mathbf{b})$.

$$\beta_{\mathbf{j}, [\mathbf{a}, \mathbf{b}]}^{\mathbf{k}, n}(\mathbf{x}) := \prod_{i=1}^r \beta_{j_i, [a_i, b_i]}^{k_i, n}(x_i) \quad (\mathbf{x} \in \mathbb{R}^r). \tag{15}$$

Theorem 6. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^r, \mathbf{a} < \mathbf{b}, n \in \mathbb{N}, \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n, \mathbf{x} \in V(\mathbf{a}, \mathbf{b})$.

$$\forall \ell \in \mathbb{N}^r, |\ell| \leq n, \forall \mathbf{y} \in V(\mathbf{a}, \mathbf{b}): \partial_{\mathbf{x}}^\ell \beta_{\mathbf{x}, [\mathbf{a}, \mathbf{b}]}^{\mathbf{k}, n}(\mathbf{y}) = \delta_{\mathbf{x}, \mathbf{y}} \cdot \delta_{\mathbf{k}, \ell}. \tag{16}$$

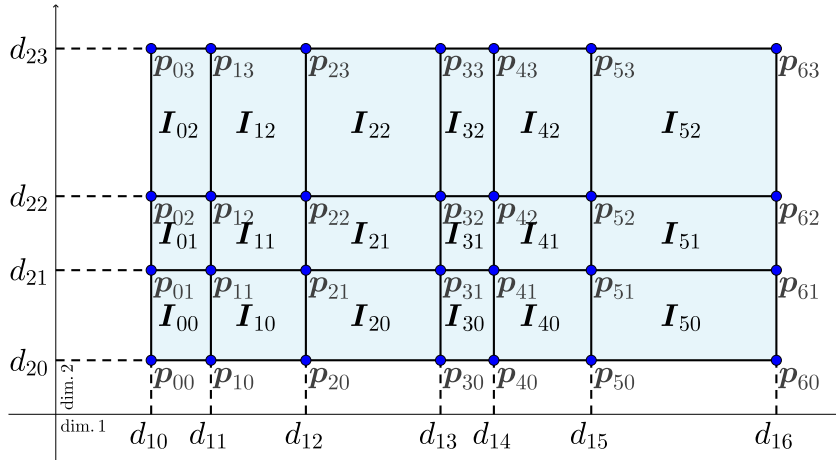


Figure 2: Example two-dimensional grid and notation

The proof is analogous to the proof of Theorem 4.

Theorem 7. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^r, \mathbf{a} < \mathbf{b}, n \in \mathbb{N}$. Given $f_i^{\mathbf{k}} \in \mathbb{R}$ for $\mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n, i \in V(\mathbf{a}, \mathbf{b})$, the multivariate polynomial

$$\tilde{f}_n(\mathbf{x}) := \sum_{i \in V(\mathbf{a}, \mathbf{b})} \sum_{|\mathbf{k}|=0}^n f_i^{\mathbf{k}} \cdot \beta_{i, [\mathbf{a}, \mathbf{b}]}^{\mathbf{k}, n}(\mathbf{x}) \quad (\mathbf{x} \in [\mathbf{a}, \mathbf{b}]) \quad (17)$$

has the following interpolation properties

$$\forall \mathbf{x} \in V(\mathbf{a}, \mathbf{b}), \forall \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n: \partial^{\mathbf{k}} \tilde{f}_n(\mathbf{x}) = f_{\mathbf{x}}^{\mathbf{k}}. \quad (18)$$

5 HERMITE INTERPOLATION ON A GRID OF SAMPLES

If we want to approximate a real valued function on a finite domain of an axis aligned bounding box, we may sample the values and the derivatives of the function on a grid of positions. Let the bounding box be $[\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^r$ and choose the subdivisions as follows. Let $\mathbf{m} \in \mathbb{N}^r$ the number of intervals in each direction and $d_{ij} \in \mathbb{R}$ ($i \in \{1, 2, \dots, r\}, j \in \{0, 1, \dots, m_i\}$) the subdivision points:

$$a_i = d_{i0} < d_{i1} < d_{i2} < \dots < d_{im_i} = b_i \quad (i \in \{1, 2, \dots, r\})$$

Then the $\mathbf{i} \in \mathbb{N}^r$ indexed vertex, enclosed interval (cell), and cell vertices are

$$\begin{aligned} \mathbf{p}_i &:= [d_{1i_1}, d_{2i_2}, \dots, d_{ri_r}]^T \in \mathbb{R}^r & (i \leq \mathbf{m}), \\ \mathbf{I}_i &:= [\mathbf{p}_i, \mathbf{p}_{i+1}] \subset \mathbb{R}^r & (i < \mathbf{m}), \\ X_i &:= V(\mathbf{p}_i, \mathbf{p}_{i+1}) \subset \mathbf{I}_i & (i < \mathbf{m}), \end{aligned}$$

where $\mathbf{1} = (1, 1, \dots, 1) \in \mathbb{N}^r$. See Figure 2 for a 2D example.

We define a global interpolating piece-wise polynomial function in $[\mathbf{a}, \mathbf{b}]$, given the function values and derivatives at the grid points.

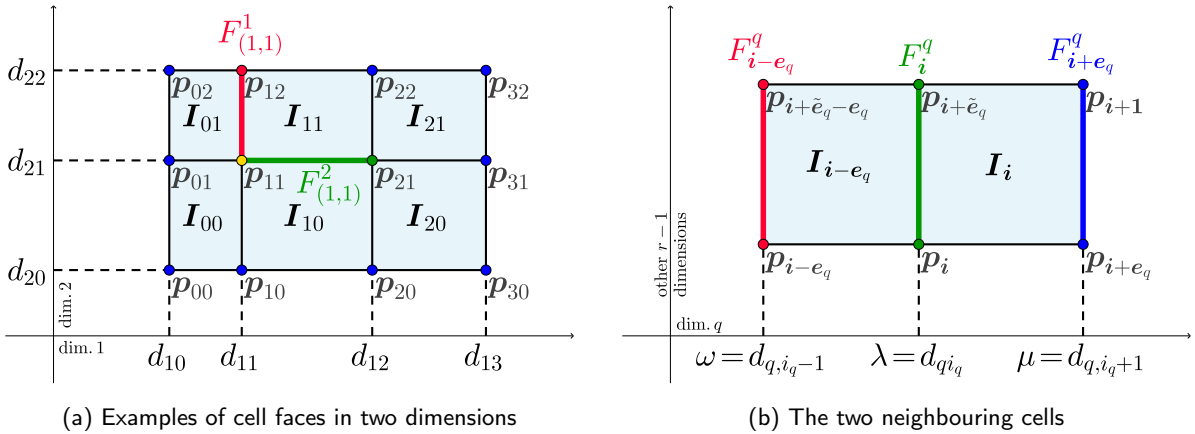


Figure 3: Illustrations for the notation used in the proof of Theorem 8.

Definition 5. Let $n \in \mathbb{N}$, $f_{\mathbf{p}_i}^{\mathbf{k}} \in \mathbb{R}$ ($\mathbf{k} \in \mathbb{N}^r$, $|\mathbf{k}| < n$, $\mathbf{i} \in \mathbb{N}^r$, $\mathbf{i} \leq \mathbf{m}$),

$$\tilde{f}_n(\mathbf{x}) := \sum_{j=0}^1 \sum_{|\mathbf{k}|=0}^n f_{\mathbf{p}_{i+j}}^{\mathbf{k}} \cdot \beta_{\mathbf{p}_{i+j}, I_i}^{\mathbf{k}, n}(\mathbf{x}) \quad (\mathbf{i} \in \mathbb{N}^r, \mathbf{i} < \mathbf{m}, \mathbf{x} \in I_i \subset [\mathbf{a}, \mathbf{b}]) \quad (19)$$

This definition is analogous to Theorem 7 in each interval. As such it has the same interpolation properties. The definition is, however, ambiguous as the intervals overlap. We show that \tilde{f}_n is in fact well defined – the overlapping intervals give the same definition to the function – and it is n times continuously differentiable on $[\mathbf{a}, \mathbf{b}]$.

Theorem 8. The interpolating piece-wise polynomial $\tilde{f}_n : [\mathbf{a}, \mathbf{b}] \rightarrow \mathbb{R}$ function is well defined and

$$\forall \mathbf{k} \in \mathbb{N}^r, |\mathbf{k}| \leq n: \partial^{\mathbf{k}} \tilde{f}_n \in C[\mathbf{a}, \mathbf{b}]. \quad (20)$$

Proof. Since \tilde{f}_n is a piece-wise polynomial, it is well defined and infinitely differentiable in the interior of all cells. We have to show that the neighbouring cells give the same definition for \tilde{f}_n on the boundary and that the desired derivatives match.

Let us now introduce notation for the faces of the cells and the set of grid points that lie on the vertices of this face.

$$\begin{aligned} F_i^q &:= [\mathbf{p}_i, \mathbf{p}_{i+\tilde{e}_q}] \subset \mathbb{R}^r, & (\mathbf{i} \in \mathbb{N}^r, \mathbf{i} \leq \mathbf{m}, q \in \{1, 2, \dots, r\}) \\ W_i^q &:= V(\mathbf{p}_i, \mathbf{p}_{i+\tilde{e}_q}) \subset F_i^q, & (\mathbf{i} \in \mathbb{N}^r, \mathbf{i} \leq \mathbf{m}, q \in \{1, 2, \dots, r\}) \end{aligned}$$

where $\tilde{e}_q = (1, 1, \dots, 1, 0, 1, \dots, 1) \in \mathbb{N}^r$ and the zero is at the q th coordinate. See Figure 3a for examples. We can also write the inner F_i^q faces as an intersection between neighbouring cells:

$$F_i^q = I_{i-e_q} \cap I_i, \quad (\mathbf{i} \in \mathbb{N}^r, \mathbf{0} < \mathbf{i} < \mathbf{m}, q \in \{1, 2, \dots, r\})$$

where $e_q = (0, 0, \dots, 0, 1, 0, \dots, 0) \in \mathbb{N}^r$ and the 1 is the q th coordinate.

Both F_i^q and W_i^q may be written out explicitly in terms of the d_{ij} subdivision points. Note that all points in the sets share a common q th coordinate.

$$\begin{aligned} F_i^q &= [d_{1i_1}, d_{1,i_1+1}] \times \dots \times \{d_{qi_q}\} \times \dots \times [d_{ri_r}, d_{r,i_r+1}], \\ W_i^q &= \{d_{1i_1}, d_{1,i_1+1}\} \times \dots \times \{d_{qi_q}\} \times \dots \times \{d_{ri_r}, d_{r,i_r+1}\}. \end{aligned}$$

We show, that the value and derivatives of \tilde{f}_n on the cell boundary F_i^q depend only on the values f_p^k , where $p \in W_i^q$ and $k \in \mathbb{N}^r, |k| \leq n$. For this let us begin with Equation (19) which can be written as

$$\tilde{f}_n(\mathbf{x}) = \sum_{p \in X_i} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_i}^{\mathbf{k}, n}(\mathbf{x}) \quad (\mathbf{i} \in \mathbb{N}^r, \mathbf{i} < \mathbf{m}, \mathbf{x} \in \mathbf{I}_i \subset [a, b]) \quad (21)$$

Since $X_i = W_i^q \cup W_{i+e_q}^q$ is a non-overlapping partition, we can break the sum into two parts. The partition corresponds to the two parallel cell faces F_i^q and $F_{i+e_q}^q$ shown in Figure 3b. Thus the above is expressed as

$$\sum_{p \in W_i^q} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_i}^{\mathbf{k}, n}(\mathbf{x}) + \sum_{p \in W_{i+e_q}^q} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_i}^{\mathbf{k}, n}(\mathbf{x}) =: S_1(\mathbf{x}) + S_2(\mathbf{x}) \quad (22)$$

Let us now inspect the partial derivatives $\ell \in \mathbb{N}^r, |\ell| \leq n$ at the cell boundary $\mathbf{y} \in F_i^q$. Note that $y_q = d_{q, i_q}$ for all $\mathbf{y} \in F_i^q$. Let us call this value $\lambda := d_{q, i_q}$. We will substitute this known coordinate in S_1 and S_2 . The two neighbouring subdivision values are $\omega := d_{q, i_q-1}$ and $\mu := d_{q, i_q+1}$. First we show, that $\partial^\ell S_2(\mathbf{y}) = 0$ ($\mathbf{y} \in F_i^q$).

$$\partial^\ell S_2(\mathbf{y}) = \sum_{p \in W_{i+e_q}^q} \sum_{|k|=0}^n f_p^k \cdot \partial^\ell \beta_{p, \mathbf{I}_i}^{\mathbf{k}, n}(\mathbf{y}) \quad (23)$$

$$\stackrel{(15)}{=} \sum_{p \in W_{i+e_q}^q} \sum_{|k|=0}^n f_p^k \cdot \prod_{j=1}^r \left(\beta_{p_j, [d_{j, i_j}, d_{j, i_j+1}]}^{k_j, n} \right)^{(\ell_j)}(y_j) \quad (24)$$

Let us now inspect the $j = q$ factor. Since $p \in W_{i+e_q}^q \Rightarrow p_q = \mu$, and $\mu \neq \lambda$:

$$\left(\beta_{p_q, [d_{q, i_q}, d_{q, i_q+1}]}^{k_q, n} \right)^{(\ell_q)}(y_q) = \left(\beta_{\mu, [\lambda, \mu]}^{k_q, n} \right)^{(\ell_q)}(\lambda) \stackrel{(9)}{=} \delta_{\mu, \lambda} \cdot \delta_{k_q, \ell_q} = 0 \quad (25)$$

Therefore $\forall \ell \in \mathbb{N}^r, |\ell| \leq n, \forall \mathbf{y} \in F_i^q: \partial^\ell S_2(\mathbf{y}) = 0$.

Let us continue with the inspection of the neighbouring cell \mathbf{I}_{i-e_q} . Similarly to Equation (21), we can write \tilde{f}_n on the interval as

$$\tilde{f}_n(\mathbf{x}) = \sum_{p \in X_{i-e_q}} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_{i-e_q}}^{\mathbf{k}, n}(\mathbf{x}) \quad (\mathbf{i} \in \mathbb{N}^r, \mathbf{e}_q \leq \mathbf{i} < \mathbf{m}, \mathbf{x} \in \mathbf{I}_{i-e_q}) \quad (26)$$

Here, $X_{i-e_q} = W_{i-e_q}^q \cup W_i^q$, which then splits the sum into two parts. The partition corresponds to the two parallel cell faces $F_{i-e_q}^q$ and F_i^q shown in Figure 3b. The above is expressed as

$$\sum_{p \in W_{i-e_q}^q} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_{i-e_q}}^{\mathbf{k}, n}(\mathbf{x}) + \sum_{p \in W_i^q} \sum_{|k|=0}^n f_p^k \cdot \beta_{p, \mathbf{I}_{i-e_q}}^{\mathbf{k}, n}(\mathbf{x}) =: Z_1(\mathbf{x}) + Z_2(\mathbf{x}) \quad (27)$$

We examine the sum on the cell face F_i^q as before. Similarly to S_2 , we will find that $\partial^\ell Z_1(\mathbf{y}) = 0$ ($\forall \ell \in$

$\mathbb{N}^r, |\ell| \leq n, \forall \mathbf{y} \in F_i^q$:

$$\partial^\ell Z_1(\mathbf{y}) = \sum_{\mathbf{p} \in W_{i-e_q}^q} \sum_{|\mathbf{k}|=0}^n f_{\mathbf{p}}^{\mathbf{k}} \cdot \partial^\ell \beta_{\mathbf{p}, \mathbf{I}_{i-e_q}}^{\mathbf{k}, n}(\mathbf{x}) \quad (28)$$

$$\stackrel{(15)}{=} \sum_{\mathbf{p} \in W_{i-e_q}^q} \sum_{|\mathbf{k}|=0}^n f_{\mathbf{p}}^{\mathbf{k}} \cdot \left(\prod_{\substack{j=1 \\ j \neq q}}^r \left(\beta_{\mathbf{p}_j, [d_{ji_j}, d_{j, i_j+1}]}^{k_j, n} \right)^{(\ell_j)}(y_j) \right) \left(\beta_{\mathbf{p}_q, [d_{q, i_q-1}, d_{q, i_q}]}^{k_q, n} \right)^{(\ell_q)}(y_q) \quad (29)$$

The factor for $j = q$ was written separately because the base interval differs from \mathbf{I}_i in the q th coordinate. Substituting the known values related to the q th dimension, the factor can be written as

$$\left(\beta_{\omega, [\omega, \lambda]}^{k_q, n} \right)^{(\ell_q)}(\lambda) \stackrel{(9)}{=} \delta_{\omega, \lambda} \cdot \delta_{k_q, \ell_q} = 0. \quad (30)$$

Finally, we need to prove that

$$\forall \ell \in \mathbb{N}^r, |\ell| \leq n, \forall \mathbf{y} \in F_i^q: \partial^\ell S_1(\mathbf{y}) = \partial^\ell Z_2(\mathbf{y}). \quad (31)$$

S_1 and Z_2 only differ in a single factor $\left(\beta_{\mathbf{p}_q, [*]}^{k_q, n} \right)$. We can reuse Equations (24) and (29), the only change needed in both is to replace the range of the outside sum by $\mathbf{p} \in W_i^q$. The q th factors of $\partial^\ell S_1(\mathbf{y})$ and $\partial^\ell Z_2(\mathbf{y})$ are as follows.

$$\begin{aligned} S_1: \quad & \left(\beta_{\mathbf{p}_q, [d_{qi_q}, d_{q, i_q+1}]}^{k_q, n} \right)^{(\ell_q)}(y_q) = \left(\beta_{\lambda, [\lambda, \mu]}^{k_q, n} \right)^{(\ell_q)}(\lambda) \stackrel{(9)}{=} \delta_{\lambda, \lambda} \cdot \delta_{k_q, \ell_q} = \delta_{k_q, \ell_q} \\ Z_2: \quad & \left(\beta_{\mathbf{p}_q, [d_{q, i_q-1}, d_{qi_q}]}^{k_q, n} \right)^{(\ell_q)}(y_q) = \left(\beta_{\lambda, [\omega, \lambda]}^{k_q, n} \right)^{(\ell_q)}(\lambda) \stackrel{(9)}{=} \delta_{\lambda, \lambda} \cdot \delta_{k_q, \ell_q} = \delta_{k_q, \ell_q} \end{aligned}$$

Substituting these into $\partial^\ell S_1(\mathbf{y})$ and $\partial^\ell Z_2(\mathbf{y})$ makes them identical. \square

Remark 3. Theorem 8 states that the interpolating multivariate polynomial \tilde{f}_n is globally n times continuously differentiable. This means that for practical purposes, the cell transitions are also sufficiently smooth.

Remark 4. The interpolation problem could be restated to use different interpolation orders in each dimension. The solution is similar in this case as well: in Equation (12), n would be replaced by an $\mathbf{n} \in \mathbb{N}^r$, and the i th factor would use n_i as its highest interpolated order.

Remark 5. In our definition, the higher dimensional order n problem interpolates derivatives up to the *total* order n . Instead, this may be changed to be the *maximal* order, in which case the multi-index $\mathbf{k} \in \mathbb{N}^r$ order goes up to $k_i \leq n$ instead of $|\mathbf{k}| \leq n$. In practice, we found that the additional terms enhance the quality of the reconstruction much less than the rest of the terms, and they cost significantly more storage. For example in the second order three-dimensional case, this means 27 coefficients per sample instead of 10.

Remark 6. In real-time practical applications the defined grid is often regular, which means that the basis polynomials are much simpler to calculate – they are merely translated to different positions.

6 APPLICATIONS OF HERMITE INTERPOLATION FILTERING

6.1 Adini Twist

As noted in the remarks of Section 5, for practical applications, we only store mixed partial derivatives up to a maximal *total* order. The rest of the values can be either implicitly assumed to be zero or approximated

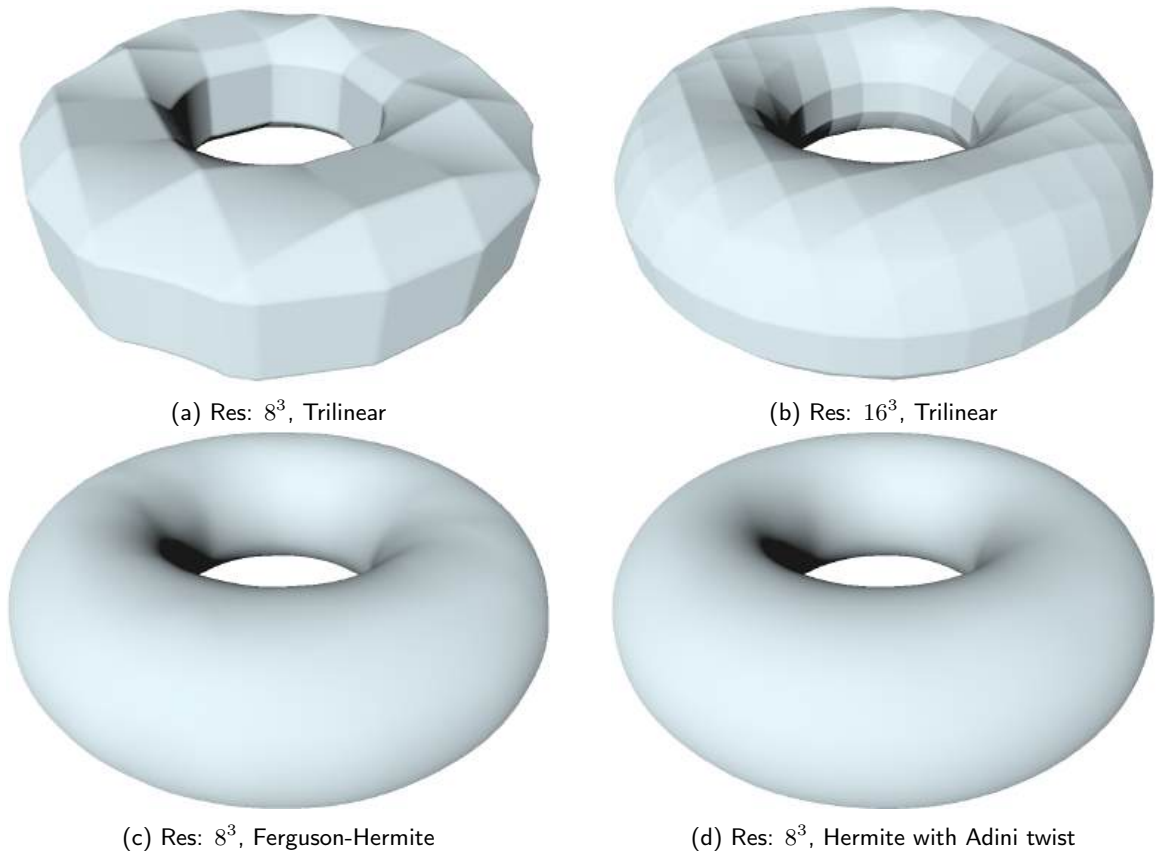


Figure 4: 3D example of an implicit surface stored as a discrete field. The trilinear case only interpolates the function value, while Hermite-interpolation interpolates the first derivatives as well. 4c) assumes zero twist, and 4d) calculates an approximated mixed partial derivatives similarly to the Adini method.

heuristically from the stored data. The first option is similar to how Ferguson patches [8] are constructed, and as such we call it Ferguson-Hermite interpolation. Although zeroing the mixed partial derivatives can be a sensible choice, it often results in visible flat spots in the interpolated result. See Fig. 4c and Fig. 4d for a comparison. Note the “wrinkles” and the slightly more angular silhouette in the first image where mixed partials are zeroed out.

For better approximation, and therefore better visual quality, the missing partial derivatives can be approximated from the rest of the stored data. We propose a particular instance of this approximation in three dimensions. Our method is similar to how the Adini twist vectors [8] are defined for parametric surfaces. For parametric surface patches, the Adini twist is defined by fitting a Coons patch [4] to the boundary of the four neighbouring patches, and evaluating the mixed partial derivative at the appropriate parameter value. For applying this to our scalar field in three dimensions, first we define the 3D implicit equivalent of the Coons patch, then we fit a Coons volume to the surrounding 8 cells of a sample, and finally evaluate the partial derivatives of the interpolating function in the middle point. These partial derivatives will then be used as the missing values for the Hermite interpolation.

For the ease of notation, let us denote linear interpolation and its derivative on the interval $[u_0, u_1]$ between

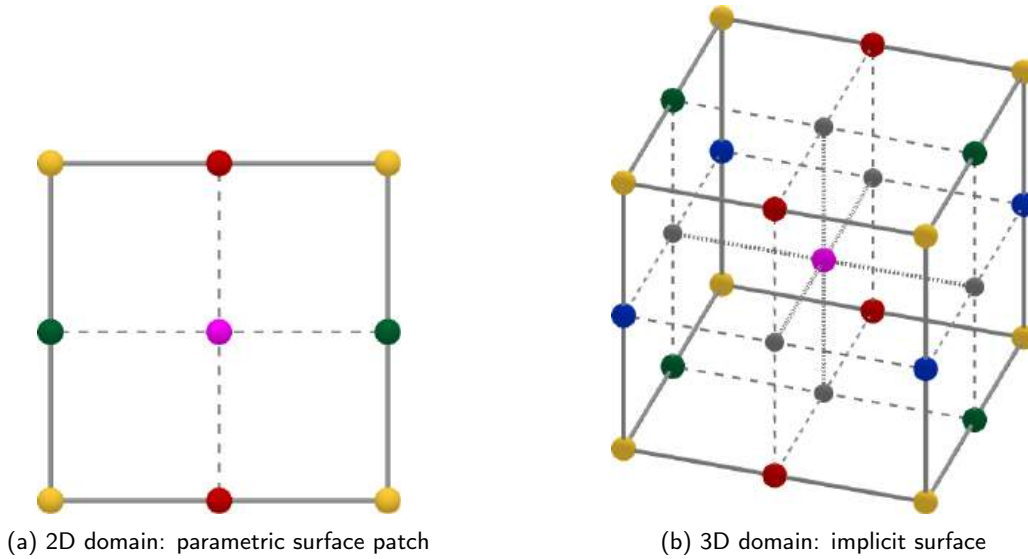


Figure 5: Values needed for the Adini twist calculation. To calculate the mixed partial derivatives at the center sample (magenta), we need to use the function value at the corner samples (yellow), and the function value and partial derivatives at the edge midpoint samples (red, green and blue).

two expressions a_0 and a_1 , that may not depend on the interpolation parameter as

$$\mathbb{L}_{u,i} a_i = \frac{u_1 - u}{u_1 - u_0} a_0 + \frac{u - u_0}{u_1 - u_0} a_1, \quad \text{and} \quad (32)$$

$$\mathbb{D}_{u,i} a_i = \frac{-1}{u_1 - u_0} a_0 + \frac{1}{u_1 - u_0} a_1. \quad (33)$$

If $[u_0, u_1]$ is the unit interval $[0, 1]$, then these simplify to

$$\mathbb{L}_{u,i} a_i = (1 - u) a_0 + u a_1, \quad \text{and} \quad (34)$$

$$\mathbb{D}_{u,i} a_i = a_1 - a_0. \quad (35)$$

We define the Coons-volume $s : [u_0, u_1] \times [v_0, v_1] \times [w_0, w_1] \rightarrow \mathbb{R}$ as

$$s(u, v, w) = \mathbb{L}_{v,j} \mathbb{L}_{w,k} s(u, v_j, w_k) + \mathbb{L}_{u,i} \mathbb{L}_{w,k} s(u_i, v, w_k) + \mathbb{L}_{u,i} \mathbb{L}_{v,j} s(u_i, v_j, w) - 2 \mathbb{L}_{u,i} \mathbb{L}_{v,j} \mathbb{L}_{w,k} s(u_i, v_j, w_k), \quad (36)$$

where $s(u, v_j, w_k)$, $s(u_i, v, w_k)$, $s(u_i, v_j, w)$ are the given functions to interpolate along the edges of the cube, and $s(u_i, v_j, w_k)$ are the values at the vertices of the cube. Similarly to the Coons patch, we first interpolate the parallel edges of the domain for the whole volume in each dimension, and subtract the interpolation of the vertices, so the final volume patch interpolates all given quantities.

We then use the Coons volume to derive the missing mixed partial derivatives for our Hermite interpolation, akin to the Adini twist method. For the calculation at each sample position, we fit a Coons volume to the

8 cubic cells surrounding the point – the point is a vertex of each cell as shown in Fig. 5b. The values at the vertices of the cube are given by the sample values, while the values along the edges are the Hermite interpolated function values. Each edge therefore consists of two cubic polynomial segments. First, we deduce the derivatives for a general Coons volume, then we evaluate them in the center point, assuming Hermite interpolation along the edges.

$$\begin{aligned} \partial_u s(u, v, w) = & \mathbb{L} \mathbb{L} s_u(u, v_j, w_k) + \mathbb{D} \mathbb{L} s(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{L} s(u_i, v_j, w) - 2 \mathbb{D} \mathbb{L} \mathbb{L} s(u_i, v_j, w_k) \end{aligned} \quad (37)$$

$$\begin{aligned} \partial_v s(u, v, w) = & \mathbb{D} \mathbb{L} s(u, v_j, w_k) + \mathbb{D} \mathbb{L} s(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{L} s(u_i, v_j, w) - 2 \mathbb{L} \mathbb{D} \mathbb{L} s(u_i, v_j, w_k) \end{aligned} \quad (38)$$

$$\begin{aligned} \partial_w s(u, v, w) = & \mathbb{L} \mathbb{D} s(u, v_j, w_k) + \mathbb{L} \mathbb{D} s(u_i, v, w_k) + \\ & + \mathbb{L} \mathbb{L} s_w(u_i, v_j, w) - 2 \mathbb{L} \mathbb{L} \mathbb{D} s(u_i, v_j, w_k) \end{aligned} \quad (39)$$

$$\begin{aligned} \partial_{uv} s(u, v, w) = & \mathbb{D} \mathbb{L} s_u(u, v_j, w_k) + \mathbb{D} \mathbb{L} s_v(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{D} s(u_i, v_j, w) - 2 \mathbb{D} \mathbb{D} \mathbb{L} s(u_i, v_j, w_k) \end{aligned} \quad (40)$$

$$\begin{aligned} \partial_{uw} s(u, v, w) = & \mathbb{L} \mathbb{D} s_u(u, v_j, w_k) + \mathbb{D} \mathbb{D} s(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{L} s_w(u_i, v_j, w) - 2 \mathbb{D} \mathbb{L} \mathbb{D} s(u_i, v_j, w_k) \end{aligned} \quad (41)$$

$$\begin{aligned} \partial_{vw} s(u, v, w) = & \mathbb{D} \mathbb{D} s(u, v_j, w_k) + \mathbb{D} \mathbb{D} s(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{L} s_w(u_i, v_j, w) - 2 \mathbb{D} \mathbb{L} \mathbb{D} s(u_i, v_j, w_k) \end{aligned} \quad (42)$$

$$\begin{aligned} \partial_{uvw} s(u, v, w) = & \mathbb{D} \mathbb{D} s_u(u, v_j, w_k) + \mathbb{D} \mathbb{D} s_v(u_i, v, w_k) + \\ & + \mathbb{D} \mathbb{D} s_w(u_i, v_j, w) - 2 \mathbb{D} \mathbb{D} \mathbb{D} s(u_i, v_j, w_k) \end{aligned} \quad (43)$$

For the rest of the derivation let us assume, that the grid is regular, and thus the the Coons volume consist of eight congruent cuboids as shown in Fig. 5b. If the spacing of the grid is $\Delta u, \Delta v$ and Δw in the three directions, then $u_1 = u_0 + 2\Delta u$ and the midpoint is at $u_m = u_0 + \Delta u$ in the first dimension. The linear interpolation and its derivative (Eq. (32) and Eq. (33)) evaluated at the midpoint simplifies to

$$\prod_{u,i} a_i(u) \Big|_{u=u_m} = \frac{1}{2}a_0(u_m) + \frac{1}{2}a_1(u_m), \text{ and} \quad (44)$$

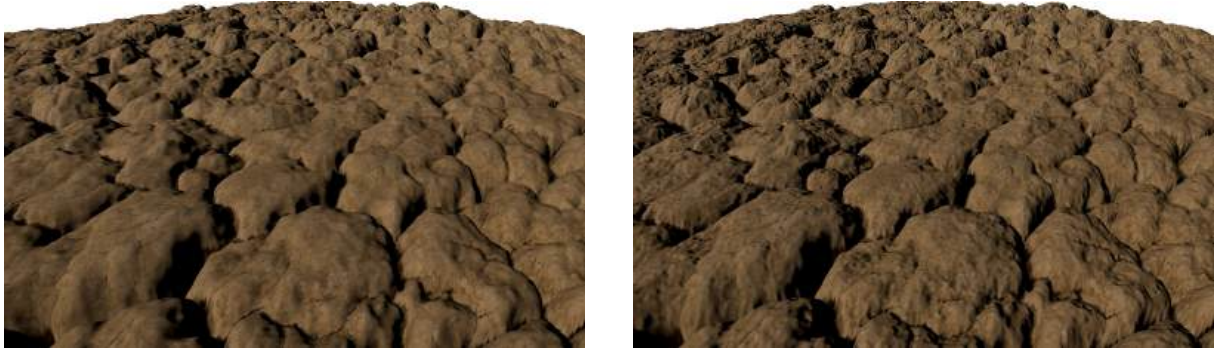
$$\prod_{u,i} a'_i(u) \Big|_{u=u_m} = \frac{1}{2\Delta u} (a'_1(u_m) - a'_0(u_m)). \quad (45)$$

The value along the edges of the Coons volume is defined by Hermite interpolation of the given grid values – two one dimensional cubic Hermite segments for each edge. The Hermite segments interpolate the function value and the derivative in the direction of the edge stored in the grid, at each endpoint and midpoint. The resulting partial derivatives are listed below with special notation to for better readability. A ‘ \odot ’ means alignment with the center point, ‘ \ominus ’ is one step towards the minimum point, and ‘ \oplus ’ is a step on the grid towards the positive direction along the given axes. For example $\mathbf{c} = [\odot \odot \odot]$ marks the center point, and $[\ominus \odot \oplus]$ has coordinates $[-\Delta u, 0, \Delta w]^T$ relative to the center.

$$\begin{aligned} \partial_{uv}s(\mathbf{c}) &= \frac{1}{4\Delta v} (s_u[\odot \oplus \oplus] + s_u[\odot \oplus \ominus] - s_u[\odot \ominus \oplus] - s_u[\odot \ominus \ominus]) \\ &+ \frac{1}{4\Delta u} (s_v[\oplus \odot \oplus] + s_v[\oplus \odot \ominus] - s_v[\ominus \odot \oplus] - s_v[\ominus \odot \ominus]) \\ &+ \frac{1}{4\Delta u \Delta v} (s[\oplus \oplus \odot] - s[\oplus \ominus \odot] - s[\ominus \oplus \odot] + s[\ominus \ominus \odot]) \\ &- \frac{1}{4\Delta u \Delta v} (s[\oplus \oplus \oplus] + s[\oplus \oplus \ominus] - s[\oplus \ominus \oplus] - s[\oplus \ominus \ominus] \\ &\quad - s[\ominus \oplus \oplus] - s[\ominus \oplus \ominus] + s[\ominus \ominus \oplus] + s[\ominus \ominus \ominus]) \end{aligned} \quad (46)$$

$$\begin{aligned} \partial_{uw}s(\mathbf{c}) &= \frac{1}{4\Delta w} (s_u[\odot \oplus \oplus] - s_u[\odot \oplus \ominus] + s_u[\odot \ominus \oplus] - s_u[\odot \ominus \ominus]) \\ &+ \frac{1}{4\Delta u \Delta w} (s[\oplus \odot \oplus] - s[\oplus \odot \ominus] - s[\ominus \odot \oplus] + s[\ominus \odot \ominus]) \\ &+ \frac{1}{4\Delta u} (s_w[\oplus \oplus \odot] + s_w[\oplus \ominus \odot] - s_w[\ominus \oplus \odot] - s_w[\ominus \ominus \odot]) \\ &- \frac{1}{4\Delta u \Delta w} (s[\oplus \oplus \oplus] - s[\oplus \oplus \ominus] + s[\oplus \ominus \oplus] - s[\oplus \ominus \ominus] \\ &\quad - s[\ominus \oplus \oplus] + s[\ominus \oplus \ominus] - s[\ominus \ominus \oplus] + s[\ominus \ominus \ominus]) \end{aligned} \quad (47)$$

$$\begin{aligned} \partial_{vw}s(\mathbf{c}) &= \frac{1}{4\Delta v \Delta w} (s[\odot \oplus \oplus] - s[\odot \oplus \ominus] - s[\odot \ominus \oplus] + s[\odot \ominus \ominus]) \\ &+ \frac{1}{4\Delta w} (s_v[\oplus \odot \oplus] - s_v[\oplus \odot \ominus] + s_v[\ominus \odot \oplus] - s_v[\ominus \odot \ominus]) \\ &+ \frac{1}{4\Delta v} (s_w[\oplus \oplus \odot] - s_w[\oplus \ominus \odot] + s_w[\ominus \oplus \odot] - s_w[\ominus \ominus \odot]) \\ &- \frac{1}{4\Delta v \Delta w} (s[\oplus \oplus \oplus] - s[\oplus \oplus \ominus] - s[\oplus \ominus \oplus] + s[\oplus \ominus \ominus] \\ &\quad + s[\ominus \oplus \oplus] - s[\ominus \oplus \ominus] - s[\ominus \ominus \oplus] + s[\ominus \ominus \ominus]) \end{aligned} \quad (48)$$



(a) Bilinear interpolation – 1 scalar/sample

(b) Hermite interpolation – 3 scalar/sample

Figure 6: Comparison of bilinear interpolation with Hermite interpolation on 256^2 height maps.

$$\begin{aligned}
 \partial_{uvw}s(\mathbf{c}) = & \frac{1}{4\Delta v\Delta w} (s_u[\odot \oplus \oplus] - s_u[\odot \oplus \ominus] - s_u[\odot \ominus \oplus] + s_u[\odot \ominus \ominus]) \\
 & + \frac{1}{4\Delta u\Delta w} (s_v[\oplus \odot \oplus] - s_v[\oplus \odot \ominus] - s_v[\ominus \odot \oplus] + s_v[\ominus \odot \ominus]) \\
 & + \frac{1}{4\Delta u\Delta v} (s_w[\oplus \oplus \odot] - s_w[\oplus \ominus \odot] - s_w[\ominus \oplus \odot] + s_w[\ominus \ominus \odot]) \\
 & - \frac{1}{4\Delta u\Delta v\Delta w} (s[\oplus \oplus \oplus] - s[\oplus \oplus \ominus] - s[\oplus \ominus \oplus] + s[\oplus \ominus \ominus] \\
 & \quad - s[\ominus \oplus \oplus] + s[\ominus \oplus \ominus] + s[\ominus \ominus \oplus] - s[\ominus \ominus \ominus])
 \end{aligned} \tag{49}$$

To summarize, for the final calculated mixed derivatives ($\partial_{uv}s, \partial_{uw}s, \partial_{vw}s, \partial_{uvw}s$), we need the function value at the vertices, and the function value and partial derivative in the edge direction at the edge middle points. The required function values and derivatives are shown in Figure 5b.

6.2 2D – Height Maps

Height maps are used in rendering detailed surfaces over lower resolution geometries [15]. An example rendering comparison is shown in Figure 6. We tried multiple different configurations for rendering height maps. Compared to a traditional bilinear filtering, Hermite interpolation is much slower due to the additional computation cost of interpolating by hand, while bilinear filtering is hardware accelerated. For the order 1 construction we store the gradient in the samples next to the function values, which is often done for normal mapping already. Calculating the interpolated value in each step of the intersection search proved to be expensive, as we expected. The performance measurements are in Table 1. We ran the tests on an AMD RX 5700 GPU at FullHD resolution.

The right column contains the performance test results for using Hermite interpolation for shading only. We did the intersection calculation on the traditional bilinearly filtered field, but calculated the surface normals of the Hermite interpolated reconstruction. The normals are exactly calculated since it is easy to differentiate the higher dimensional Hermite polynomials. This rendering method combines the higher quality visuals of Hermite interpolation with the performance of bilinear filtering. The visual quality of the Hermite-interpolated surface is similar to the quality of a bilinear surface of at least twice the resolution, as such we can lower the resolution with this technique without losing quality and even possibly gaining performance. The order 1 field representation uses three scalars per sample which is three times more compared to the traditional order 0 storage. However, as noted, this may be counteracted by lowering the resolution.

Resolution	Bilinear	Hermite	H. normal
128 × 128	0.17 ms	0.42 ms	0.21 ms
256 × 256	0.20 ms	0.44 ms	0.24 ms
512 × 512	0.28 ms	0.49 ms	0.31 ms

Table 1: Performance of height map rendering shown in Fig. 6 at FullHD resolution. The resolution of the height map impacts the performance through memory read efficiency and cache coherency. The three compared methods are *Bilinear* – storing only the height value, *Hermite* – storing the gradient as well and using Hermite interpolation, and *H. normal* – using bilinear interpolation for the intersection search and Hermite interpolation for the normal calculation. The Hermite interpolated fields store three times more data per sample compared to the bilinear case but they offer higher quality, allowing for reduced field resolutions.

6.3 3D – Signed Distance Fields

Signed distance fields (SDFs) are versatile implicit representations for surfaces. If the exact function is given the gradient can be calculated by automatic or numerical differentiation. In the case of triangle meshes the gradient is defined by the closest triangle. An example is shown using sphere tracing in Figure 7. Our measurements showed the expected performance hit: the order 1 construction (function value and gradient – 4 floating point values per sample) rendered in an average 1.02 ms, the order 2 construction (up to second order partial derivatives – 10 values in total) rendered in 1.94 ms. In comparison the hardware accelerated trilinear filtering on the function values takes about 0.30 ms.

Using Hermite interpolation for shading only in 3D is less appealing than using it for the intersection search too, since thin features will lose their silhouette. The higher order constructions give better precision than trilinear interpolation in all examined statistics – maximum, average, median, standard deviation, sign correctness, different percentiles of the absolute error. However, the second order construction does inhibit some additional artifacts due to the interpolation method – there are some “flat spots” near the sample position. As such the best visual quality is achieved by using order 1 Hermite interpolation.

Using the Adini twist method described in Section 6.1 for order 1 Hermite interpolation costs even more performance in practice, and as seen in Fig. 4 and Fig. 7, it barely changes the surface. As such it is useful for model storage compression purposes, but it does not provide enough benefits in real-time applications.

Compared to the traditional order 0 (trilinear) representation, the order 1 construction stores four times more data per sample. Similarly to height mapping, using Hermite interpolation for SDFs allows the usage of lower resolution fields at similar visual quality. The Adini twist values may be calculated during sampling but it is very expensive to calculate therefore we suggest to compute the values upon loading the field, and storing the extra four scalars per sample as a separate texture during runtime, doubling the effective storage cost.

6.4 Storage Optimization

In the two-dimensional case, the gradient can be compressed to the $[-1, 1]$ interval, which is favorable, as in practice normalized texture types are more efficient compared to floating point types. The compression is done by calculating the normalized 3D normal of the surface and then storing only the first two coordinates: $[1, 0, \partial_x \hat{f}]^T \times [0, 1, \partial_y \hat{f}]^T = [-\partial_x \hat{f}, -\partial_y \hat{f}, 1]^T$.

For the three-dimensional signed distance fields, the gradient is almost everywhere unit length, therefore it effectively has two degrees of freedom. It can be encoded into two values with for example octahedral encoding [3]. In both cases unpacking is needed and the stored data cannot be interpolated directly but we already use manual filtering in Hermite interpolation.

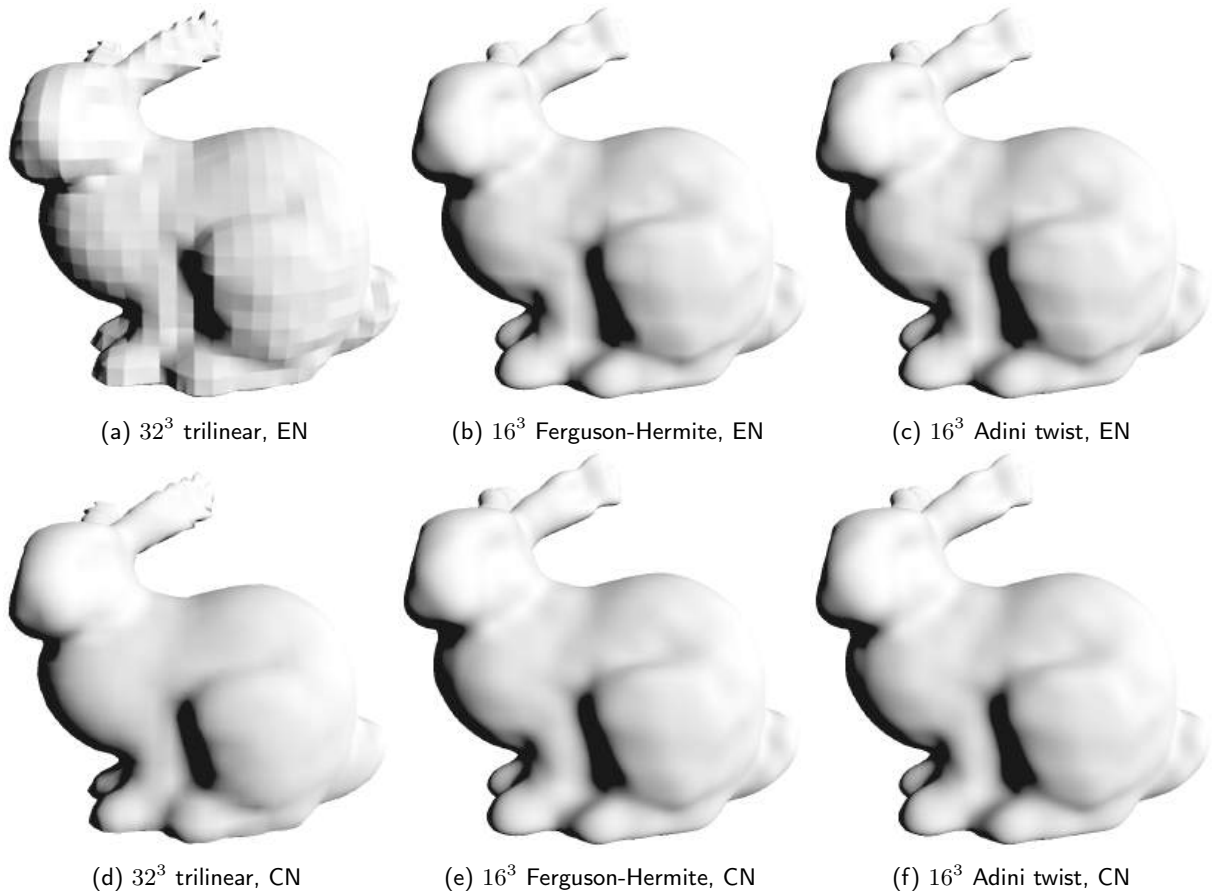


Figure 7: Comparison of trilinear interpolation with Hermite interpolation on signed distance fields rendered with sphere tracing. The top row shows the exact normals (EN) of the models, while the bottom row displays the same models with normals calculated using central differences (CN).

7 CONCLUSIONS

Hermite interpolation is a useful tool in computer graphics. We derived and proved the interpolation properties of Hermite interpolation for the general dimensional case and C^n -continuity. The filtering method may be used in two and three-dimensional problems for smooth surface reconstruction and shading. For hightmaps it is even usable without performance loss by using half the resolution compared to a traditional bilinearly filtered field, and calculating the shading normals from the Hermite interpolation. In three dimensions, our construction for signed distance field may be used if the main focus is on precision and performance is secondary.

ACKNOWLEDGEMENTS

Supported by the ÚNKP-23-4 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

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

ORCID

Róbert Bán, <http://orcid.org/0000-0002-8266-7444>

Gábor Valasek, <http://orcid.org/0000-0002-0007-8647>

REFERENCES

- [1] Aaltonen, S.: GPU-based clay simulation and ray-tracing tech in Claybook, 2018.
- [2] Bálint, C.; Valasek, G.: Accelerating Sphere Tracing. In EG 2018 - Short Papers, 4 pages. The Eurographics Association, Delft, Netherlands, 2018. <http://doi.org/10.2312/EGS.20181037>.
- [3] Cigolle, Z.H.; Donow, S.; Evangelakos, D.; Mara, M.; McGuire, M.; Meyer, Q.: A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques (JCGT)*, 3(2), 1–30, 2014. ISSN 2331-7418.
- [4] Coons, S.A.: SURFACES FOR COMPUTER-AIDED DESIGN OF SPACE FORMS:. Tech. rep., Defense Technical Information Center, Fort Belvoir, VA, 1967. <http://doi.org/10.21236/AD0663504>.
- [5] Csébfalvi, B.: Beyond trilinear interpolation: Higher quality for free. *ACM Transactions on Graphics*, 38(4), 1–8, 2019. ISSN 0730-0301, 1557-7368. <http://doi.org/10.1145/3306346.3323032>.
- [6] Dummer, J.: Cone Step Mapping: An Iterative Ray-Heightfield Intersection Algorithm, 2006. <http://www.lonesock.net/files/ConeStepMapping.pdf>.
- [7] 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*. SIGGRAPH, MediaMolecule, 2015.
- [8] Farin, G.E.: *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Computer Science and Scientific Computing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5 ed., 2001. ISBN 1-55860-737-4. <http://doi.org/10.5555/501891>.
- [9] 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 Conference on Computer Graphics and Interactive Techniques*, 249–254. ACM Press, SIGGRAPH, 2000. ISBN 978-1-58113-208-3. <http://doi.org/10.1145/344779.344899>.
- [10] Green, C.: Improved Alpha-tested Magnification for Vector Textures and Special Effects. In *ACM SIGGRAPH 2007 Courses on - SIGGRAPH '07, SIGGRAPH '07*, 9–18. ACM Press, San Diego, California, 2007. ISBN 978-1-4503-1823-5. <http://doi.org/10.1145/1281500.1281665>.
- [11] Hart, J.C.: Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer*, 12(10), 527–545, 1996. ISSN 01782789. <http://doi.org/10.1007/s003710050084>.
- [12] Keinert, B.; Schäfer, H.; Korndörfer, J.; Ganse, U.; Stamminger, M.: Enhanced Sphere Tracing. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, 8 pages. The Eurographics Association, 2014. ISBN 978-3-905674-72-9. <http://doi.org/10.2312/stag.20141233>.
- [13] Policarpo, F.; Oliveira, M.M.: Relaxed Cone Stepping for Relief Mapping. In *GPU Gems 3*, 2007. ISBN 0-321-51526-9.
- [14] Sigg, C.; Hadwiger, M.: Fast Third-Order Texture Filtering. In M. Pharr, ed., *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, 313–329. Addison-Wesley, Upper Saddle River, 2005. ISBN 0-321-33559-7.
- [15] Szirmay-Kalos, L.; Umenhoffer, T.: Displacement Mapping on the GPU - State of the Art. *Computer Graphics Forum*, 27(6), 1567–1592, 2008. ISSN 01677055, 14678659. <http://doi.org/10.1111/j.1467-8659.2007.01108.x>.
- [16] Wright, D.: Dynamic Occlusion with Signed Distance Fields, 2015.