Accurate Isosurface Interpolation with Hermite Data

Simon Fuhrmann
TU Darmstadt

Michael Kazhdan
Johns Hopkins University

Michael Goesele
TU Darmstadt

Abstract

In this work we study the interpolation problem in contouring methods such as Marching Cubes. Traditionally, linear interpolation is used to define the position of an isovertex along a zero-crossing edge, which is a suitable approach if the underlying implicit function is (approximately) piecewise linear along each edge. Non-linear implicit functions, however, are frequently encountered and linear interpolation leads to inaccurate isosurfaces with visible reconstruction artifacts. We instead utilize the gradient of the implicit function to generate more accurate isosurfaces by means of Hermite interpolation techniques. We propose and compare several interpolation methods and demonstrate clear quality improvements by using higher order interpolants. We further show the effectiveness of the approach even when Hermite data is not available and gradients are approximated using finite differences.

1. Introduction

Implicit functions are a popular surface representation for many reconstruction algorithms. As opposed to explicit representations, implicit functions are agnostic to topology and more easily support blending between shapes, boolean queries, and morphological operations such as erosion and dilation. Contouring of implicit functions, i.e., extracting an explicit surface from the implicit representation, is an important application in computer graphics.

An implicit function $F\colon \mathbb{R}^3 \rightarrow \mathbb{R}$ associates a scalar value $F(\mathbf{x})$ to every point in space $\mathbf{x}$. The function implicitly defines a surface S as the *level-set* $S = \{\mathbf{x} \mid F(\mathbf{x}) = d\}$ with respect to the *isovalue* d. The surface is guaranteed to be manifold if d is not a singular value of the function (i.e., the gradient ∇F does not vanish on the level-set). The level-set S is also called the *isosurface* of F. Without loss of generality, we assume $d = 0$ and note that choosing a different d is equivalent to subtracting d from F. A popular example for an implicit function is the *signed distance function* (SDF), which describes for every point in space the distance to the closest point on the shape, with the sign of the function indicating whether the point is interior or exterior to the surface. The surface is then defined as the zero level-set of the implicit function.

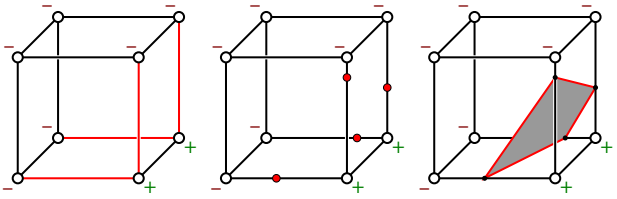

Figure 1. Marching Cubes: Four edges of the cube contain a sign change (left), interpolation of isovertices (middle) and the final polygonal surface (right).

An implicit function is often represented on a regular lattice, i.e., sampled at uniformly spaced positions, and Marching Cubes [15] is often the contouring algorithm of choice. As a regular sampling of F is unsuitable for the representation of large or multi-scale shapes, octrees and tetrahedralizations have been used, and the isosurface is obtained with more general algorithms [1, 19, 4, 24, 11].

For these representations, the implicit function is sampled at discrete positions and the *isovertex* positions are determined by interpolation along edges, and triangulations connecting the isovertices are computed per cell, see Figure 1. Traditionally, this interpolation is performed using linear approximation, i.e., by finding the zero-crossing of a linear function along each edge. If the implicit function is actually non-linear along the edges, the interpolated isovertex positions poorly estimate the actual position of the zero-crossing along the edge, see Figure 2. Depending on the type of implicit function, these inaccuracies often manifest themselves in structured patterns on the final surface, which can appear as ringing or undulating artifacts. This is caused by alternating between over- and underestimation of the correct zero-crossing.

A simple example of a non-linear implicit function for the sphere with radius r is $F_q(\mathbf{x}) = x^2 + y^2 + z^2 - r^2$. Contouring F_q with linear interpolation leads to a larger reconstruction error than contouring the signed distance function $F_\ell(\mathbf{x}) = \sqrt{x^2 + y^2 + z^2} - r$, although both functions have the same zero level-set. Note that near the isosurface, the signed distance function has small second derivatives and is approximately linear. Figure 3 visualizes this reconstruc-

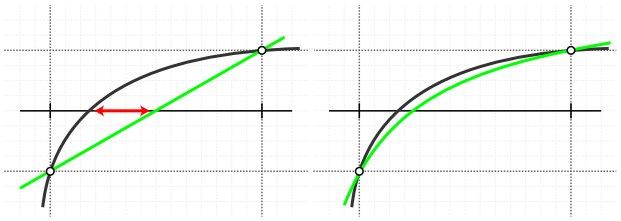

Figure 2. For non-linear functions (black), the linear interpolant (green) is often a poor fit (left). Interpolation with higher-order functions leads to much higher accuracy (right).

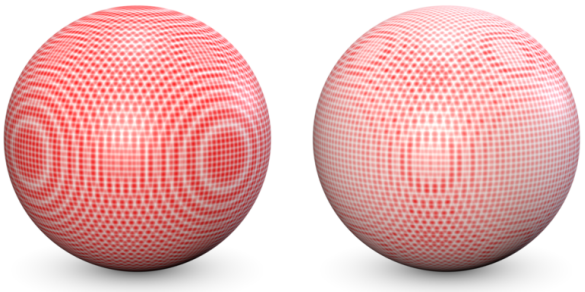

Figure 3. Visualization of the reconstruction error of the quadratic function F_q (left) and the signed distance function F_ℓ (right) on a $64 \times 64 \times 64$ voxel grid. Red corresponds to a larger error.

tion error. While both reconstructions have high-frequency errors due to discretization, reconstruction of F_q exhibits considerably more pronounced low-frequency ringing artifacts around the axes of the coordinate system.

A piecewise linear approximation to a smooth function has an approximation error that depends on the sampling density of the function. The approximation error decreases as $\mathcal{O}(h^2)$ with the sample spacing h, i.e., if the sampling spacing is halved, the approximation error is reduced by $1/4$. This quadratic behavior suggests that increasing the sampling density will be effective in reducing these artifacts. However, even in the cases where the continuous implicit function is available for re-sampling, such a refinement has a dramatic impact on memory consumption and runtime.

In this work we argue that contouring non-linear implicit functions requires additional data in order to obtain accurate, artifact-free results for high quality reconstruction. We advocate the use of Hermite data, i.e., utilizing the implicit function gradient $\nabla F(\mathbf{x})$ in addition to the values $F(\mathbf{x})$ at the sampling positions $\mathbf{x}$. Depending on the application, ∇F can be analytically computed or estimated using finite differences. We present several formulations for incorporating the derivatives in the interpolation scheme and evaluate the quality of the obtained results on higher-order implicit functions. Further, we incorporate all interpolation methods in *Poisson Surface Reconstruction* (PSR) [9, 10] and the more recent *Floating Scale Surface Reconstruction* (FSSR) [5], and analyze the impact of the different formulations on the reconstruction accuracy. We demonstrate clear surface quality improvements on synthetic and real-world data compared to linear interpolation. This improvement has motivated the incorporation of one Hermite interpolation technique in the early PSR code [18]. This work is the first to compare the different non-linear interpolation techniques and evaluate the practical implications on surface quality.

2. Related Work

The most popular method for contouring implicit functions is the *Marching Cubes* algorithm [15] which uses linear interpolation to place isovertices along the zero-crossing edges of a regular lattice and then defines a triangulation by connecting the isovertices within each cell, see Figure 1.

Although initially proposed for regular hexahedral grids, the Marching Cubes algorithm has been extended to adaptive space partitions including octrees [1, 19, 24, 23, 11] and (graded) tetrahedralizations of space [4]. To define the implicit surface, all these approaches require estimating the position of the isovertex along a zero-crossing edge, and linear interpolation is the technique most commonly used.

Many extensions of Marching Cubes have been proposed [20], including the reconstruction of bicubic spline surfaces [7] or continuous quadratic implicit functions for visualization purposes [17]. In contrast, our method does not compute a higher-order surface representation. It uses similar ideas for the purpose of reducing reconstruction artifacts, but is restricted to a one dimensional interpolation problem along the zero-crossing edges.

Hermite data has been used in the dual contouring method by Ju et al. [8], by Manson and Schaefer [16], and the primal/dual hybrid approach by Kobbelt et al. [12]. They use Hermite data to construct planes that are tangent to the surface and minimize a quadratic error function (QEF) to solve for an isovertex position in the interior of the cell. The minimizer is often a poor estimate of the actual function, and may require additional function evaluations [16]. There are numerical difficulties in solving the linear system of equations induced by the QEFs, e.g., the minimizer is not guaranteed to be in the interior of the cell (see [22, 8] for more details). These methods are different in that they focus on improving the reconstruction of sharp features and edges in the implicit function, not on more accurate isosurface interpolation.

In Lempitsky's work [14] a smooth implicit function is reconstructed from a binary volume by solving a constrained optimization problem minimizing the function curvature. This differs from our approach in that we do not use binary input and perform more accurate interpolation "on the fly" without having to solve a global optimization problem. As in Lempitsky's work, we also guarantee correctness in that we only place an isovertex along an edge whose endpoints have opposite signs.

3. Hermite Interpolation

The interpolation problem in primal contouring methods is one-dimensional because we are only interested in the root of the implicit function F along an edge $\mathbf{e}$. We call the restriction of F to this one-dimensional subspace $f = F\big|_{\mathbf{e}}$. To perform Hermite contouring, the values F and the gradient ∇F must be available at the sampling positions. For interpolation, however, we are only interested in the derivative $f' = \nabla F\big|_{\mathbf{e}}$ at the sampling positions along the direction of the edge $\mathbf{e}$. If the edges are axis-aligned, the derivative f' is just the corresponding component of the gradient ∇F. Otherwise, the directional gradient along $\mathbf{e}$ can be obtained with the dot product: $f' = \langle \nabla F, \mathbf{e} \rangle / \|\mathbf{e}\|^2$.

An edge $\mathbf{e}$ is represented by its two endpoints $\mathbf{x}_0$, $\mathbf{x}_1$, which are the sampling positions of F. To formulate the interpolation in a uniform setting, we scale the interval between $\mathbf{x}_0$, $\mathbf{x}_1$ to $[0, 1]$. This requires scaling the derivatives ∇F by a factor of $\|\mathbf{x}_0 - \mathbf{x}_1\|$. Given the function values and derivatives

$$\begin{aligned} f(0) &= v_0 \qquad f'(0) = d_0 \\ f(1) &= v_1 \qquad f'(1) = d_1 \end{aligned} \tag{1}$$

we describe several ways for using Hermite interpolation to obtain a more accurate isovertex position along the edge $\mathbf{e}$. In particular, we investigate cubic interpolation as well as two different types of quadratic interpolation.

3.1. Third Order Polynomial

A cubic function has four degrees of freedom, so it seems natural to use the two value and two derivative constraints to obtain a unique solution for the polynomial coefficients

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ p'(x) &= a_1 + 2a_2 x + 3a_3 x^2. \end{aligned} \tag{2}$$

Substituting the constraints from (1) into (2) leads to the linear system of equations (see, for example [13])

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ d_0 \\ d_1 \end{pmatrix} \tag{3}$$

with the unique solution

$$\begin{aligned} a_0 &= v_0 \qquad a_2 = 3v_1 - 3v_0 - 2d_0 - d_1 \\ a_1 &= d_0 \qquad a_3 = 2v_0 - 2v_1 + d_0 + d_1. \end{aligned} \tag{4}$$

Although straightforward, a cubic polynomial can have up to three real roots whose location and count may be sensitive to small perturbations of the function coefficients, see Figure 4. To uniquely define the position of an isovertex, we observe that there must be an odd number of roots along a zero-crossing edge, and we always use the "middle" root.

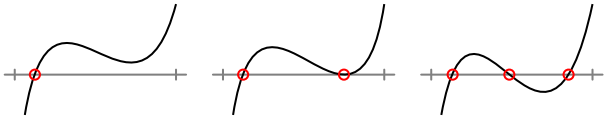

Figure 4. Different cases for the roots of a cubic function: One root (left), three roots counting multiplicity (middle), and three distinct roots (right).

This corresponds to using the single root in the case of linear and quadratic interpolation, and is also well-defined for higher-order interpolants.

3.2. Second Order Polynomials

Using a second order interpolant is the correct choice if the implicit function is known to be quadratic. Examples include PSR, where the implicit function is represented as a linear combination of second-order B-splines (or if the implicit function is regularized to have small third derivative). Since a quadratic function has three degrees of freedom, substituting the four constraints from (1) into (2) with $a_3 = 0$ yields an overdetermined linear system of equations of form $\mathtt{A}\mathbf{x} = \mathbf{b}$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ d_0 \\ d_1 \end{pmatrix}. \tag{5}$$

The coefficient matrix $\mathtt{A}$ has full rank, so there is no exact solution in general.

Least-Squares Solution: We can solve the linear system in (5) in a least-squares fashion using the normal equation, multiplying with $\mathtt{A}^\intercal$ on the left to get:

$$\mathtt{A}^\intercal\mathtt{A}\mathbf{x} = \mathtt{A}^\intercal\mathbf{b} \quad \Rightarrow \quad \mathbf{x} = (\mathtt{A}^\intercal\mathtt{A})^{-1}\mathtt{A}^\intercal\mathbf{b}. \tag{6}$$

The matrix $(\mathtt{A}^\intercal\mathtt{A})^{-1}\mathtt{A}^\intercal$ can be precomputed and the coefficients can be hard-coded as in (4). However, the least-squares solution produces a polynomial $p(x)$ where the constraints on the values are not exactly met. This can lead to the situation where, although $f(x)$ has a zero-crossing along the edge, $p(x)$ does not. For this reason we will not further consider this solution.

Least-Squares Derivatives: Instead, to guarantee that $p(x)$ always has a root along the edge if $f(x)$ also has a root, the quadratic function must interpolate the values of the implicit function $p(0) = v_0$ and $p(1) = v_1$. We now discuss a solution that is least-squares optimal for the derivatives, and interpolates the function values. For the value constraints, we have according to (5):

$$\begin{aligned} a_0 &= v_0 \\ a_0 + a_1 + a_2 &= v_1. \end{aligned} \tag{7}$$

The derivatives give rise to the constraints which must be met in a least-squares sense

$$\begin{aligned} a_1 &= d_0 \\ a_1 + 2a_2 &= d_1 \end{aligned} \tag{8}$$

which leads to the minimization problem

$$\arg\min_{a_1,a_2} : (a_1 - d_0)^2 + (a_1 + 2a_2 - d_1)^2. \tag{9}$$

From (7) we know that $a_2 = v_1 - a_1 - v_0$, and substituting a_2 in (9) yields a least-squares problem in a single variable:

$$\arg\min_{a_1} : (a_1 - d_0)^2 + (a_1 - 2v_1 + 2v_0 + d_1)^2 \tag{10}$$

Setting the derivative of (10) to zero and solving for a_1 leads to the polynomial coefficients

$$\begin{aligned} a_0 &= v_0 \\ a_1 &= \frac{d_0 - d_1}{2} + v_1 - v_0 \\ a_2 &= \frac{d_1 - d_0}{2}. \end{aligned} \tag{11}$$

Examining the coefficient a_2, we see that the second order term vanishes if the implicit function is locally linear, i.e., if $d_0 = d_1$.

Third Order Elimination: Finally, we discuss two possible second-order solutions that can be obtained from the third order solution in (4) by eliminating the third order coefficient a_3. This can be achieved by introducing an additional degree of freedom for the derivatives.

Scaling the derivatives by s and setting a_3 to zero gives

$$\begin{aligned} 2v_0 - 2v_1 + s \cdot d_0 + s \cdot d_1 &= 0 \\ s &= \frac{2v_1 - 2v_0}{d_0 + d_1}. \end{aligned} \tag{12}$$

If the derivatives d_0 and d_1 in (4) are scaled by s, the cubic term vanishes. However, the solution in (12) becomes unstable if the derivatives cancel each other out, i.e., $d_0 + d_1 \approx 0$. Whether this instability causes actual problems depends on the properties of the implicit function. For example, this is the method implemented in the PSR code for interpolating the indicator function, which has a steep gradient in the vicinity of the isosurface and is unlikely to have partial derivatives with opposite signs.

An alternative approach to scaling is to introduce an additive degree of freedom o

$$\begin{aligned} 2v_0 - 2v_1 + (d_0 + o) + (d_1 + o) &= 0 \\ o &= \frac{1}{2}(2v_1 - 2v_0 - d_0 - d_1). \end{aligned} \tag{13}$$

This solution has an interesting property. When adding the offset o to the derivatives d_0 and d_1 in (4), it can be shown that this solution is equivalent to the solution in (11).

4. Algebraic Surfaces

We now compare the interpolation methods on synthesized, non-linear implicit functions. A "ground truth" isosurface is generated by sampling a $512 \times 512 \times 512$ voxel grid and using linear interpolation to define the isovertex positions on zero-crossing edges. The test meshes are then extracted from a $64 \times 64 \times 64$ voxel grid and compared to the ground truth. The following interpolation methods are evaluated:

- `LINEAR`: Linear interpolation without derivatives
- `SCALING`: The quadratic method in (12) that scales the derivatives to eliminate the third order term
- `LSDERIV`: The quadratic method in (11) and (13) that interpolates the function values and least-squares fits the derivatives
- `CUBIC`: The cubic polynomial fit in (4)

For comparison to the ground truth, we use *Metro* [3], a tool for measuring distances between triangle meshes. Color-coding is used to visualize the distance between the ground truth and the test mesh directly on the surface, with red indicating larger distances. We also compare the impact of using the analytically computed gradient ∇F with a central differences approximation of ∇F. Note that, when using finite differences in conjunction with cubic interpolation, one obtains the standard Catmull-Rom interpolant [2].

Smooth Box: The implicit function of the *Smooth Box* dataset is given by

$$F(\mathbf{x}) = x^4 + y^4 + z^4 - 1. \tag{14}$$

This is a fourth-order function and cannot be exactly reconstructed with any of the interpolation methods in Section 3. Figure 5 visualizes the reconstruction error for all interpolation methods. Table 1 lists the maximum, mean and root-mean-square (RMS) distances to the ground truth mesh for the analytic and finite differences gradient.

Genus-2: The implicit function for the *Genus-2* dataset is a fifth-order polynomial with mixed terms

$$\begin{aligned} F(\mathbf{x}) = {} & 2y(y^2 - 3x^2)(1 - z^2) \\ & + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2). \end{aligned} \tag{15}$$

The error is visualized in Figure 6 and distances to the ground truth are given in Table 2. For both datasets the reconstruction errors of the non-linear interpolants are barely distinguishable and give nearly identical results regardless of whether analytic gradients or finite-differences are used.

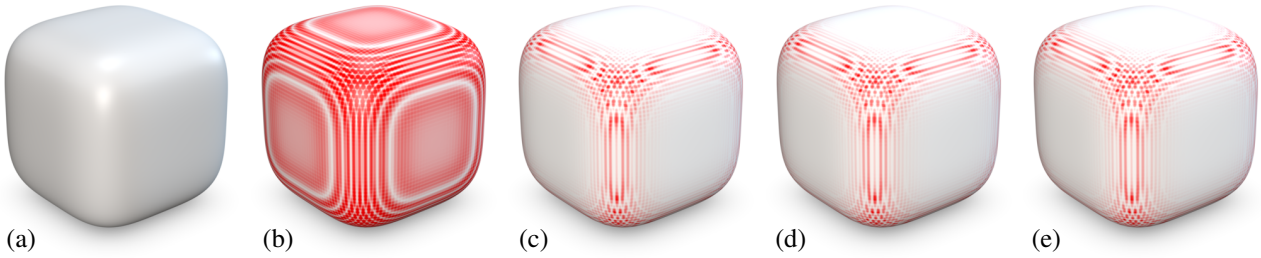

Figure 5. *Smooth Box*: (a) Ground truth, (b) `LINEAR` interpolation with color-coded reconstruction errors, (c) `SCALING` interpolation, (d) `LSDERIV` interpolation, and (e) `CUBIC` interpolation. All gradients have been computed analytically.

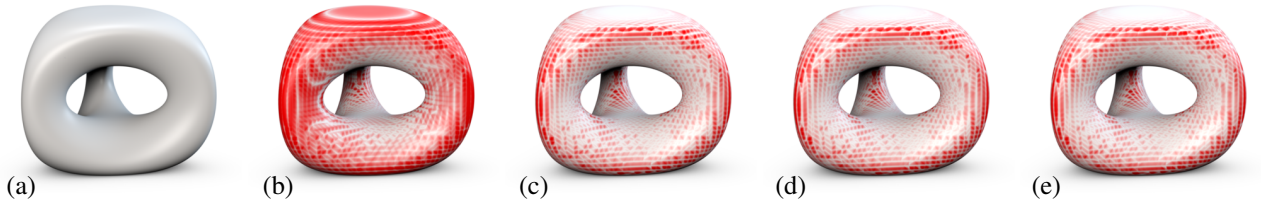

Figure 6. *Genus-2*: (a) Ground truth, (b) `LINEAR` interpolation, (c) `SCALING` interpolation with analytic gradients, (d) `LSDERIV` interpolation with analytic gradients, and (e) `LSDERIV` interpolation with approximate finite differences gradients.

Analytic ∇F	max	mean	RMS
`LINEAR`	5.200640	2.527520	2.655661
`SCALING`	4.576201	0.914086	1.225579
`LSDERIV`	4.576010	0.912602	1.224503
`CUBIC`	4.577743	0.912709	1.225184
Approx. ∇F	max	mean	RMS
`SCALING`	4.576851	0.916040	1.227008
`LSDERIV`	4.575684	0.911629	1.223793
`CUBIC`	4.581211	0.912659	1.225978

Table 1. *Smooth Box*: Distances to the ground truth mesh with analytic ∇F (top) and finite differences approximation (bottom). The distances are scaled for readability (factor 10^4).

Analytic ∇F	max	mean	RMS
`LINEAR`	2.0846366	0.4342808	0.5301991
`SCALING`	2.1002761	0.1964645	0.2915423
`LSDERIV`	2.0975643	0.1957446	0.2908659
`CUBIC`	2.0964227	0.1959128	0.2908642
Approx. ∇F	max	mean	RMS
`SCALING`	2.0997145	0.1974943	0.2925005
`LSDERIV`	2.0943636	0.1953266	0.2904807
`CUBIC`	2.0925705	0.1961537	0.2905572

Table 2. *Genus-2*: Distances to the ground truth mesh with analytic ∇F (top) and finite differences approximation (bottom). The distances are scaled for readability (factor 10^3).

5. Analytic and Discrete Surfaces

We implemented the interpolation methods in *Poisson Surface Reconstruction* (PSR) [9, 10] and the more recent *Floating Scale Surface Reconstruction* (FSSR) [5] to analyze the impact of Hermite interpolation on real surface reconstruction algorithms. Note that the `SCALING` method in (12) is already implemented in the PSR code [18]. In both, PSR and FSSR, the gradient of the implicit function can be computed analytically. Because the original weighting function in FSSR is not $\mathcal{C}^1$-continuous, we replace it with the weighting function $w(r) = \frac{1}{3^{12}}(r-3)^{12} \cdot (r+1)^4$.

We first consider synthetic data for which the ground truth is available and the reconstruction errors are easier to measure and visualize. Then, we demonstrate Hermite interpolation on real-word data from 3D scanners and Multi-View Stereo. Finally, we show an application to isosurface extraction from medical images.

5.1. Synthetic Data

We first evaluate the interpolation methods on two synthetic datasets, namely the *Sphere* and the *Blob*. To this end, we obtained high-resolution triangle meshes for both datasets and use these as ground truth. A point set is generated by computing per-vertex normals and, in the case of FSSR, also per-vertex scale values. The connectivity information is then discarded and the resulting point sets are used for reconstruction with PSR and FSSR.

Sphere Dataset: Because both PSR and FSSR use non-linear basis functions, estimation of isovertex positions using linear interpolation leads to artifacts, see Figure 7. Table 3 gives the distances of the reconstructed meshes from the ground truth. PSR produces essentially the same quality result for all non-linear interpolants while the FSSR error improves for most metrics with cubic interpolation.

Figure 7. *Sphere:* Visualization of the reconstruction error with PSR (top) and FSSR (bottom).

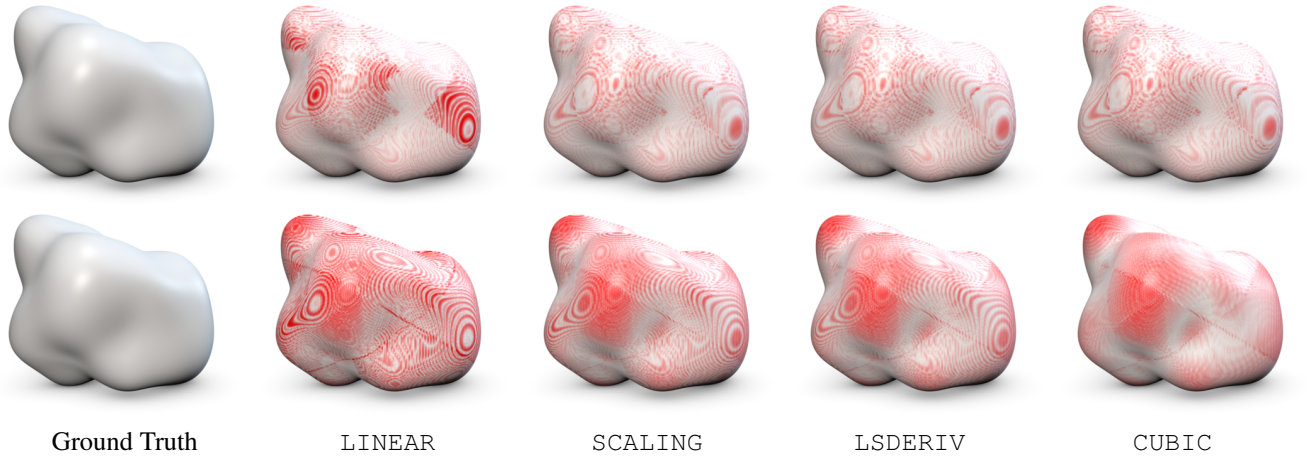

Figure 8. *Blob:* Visualization of the reconstruction error with PSR (top) and FSSR (bottom).

PSR [9]	max	mean	RMS
LINEAR	1.4475699	0.2142690	0.2911505
SCALING	0.8729671	0.1757755	0.2155248
LSDERIV	0.8729671	0.1757931	0.2155422
CUBIC	0.8729671	0.1758032	0.2155564
FSSR [5]	max	mean	RMS
LINEAR	1.5472957	0.3554895	0.3957082
SCALING	0.7770098	0.3520047	0.3692734
LSDERIV	0.7552927	0.3527912	0.3668787
CUBIC	0.6887877	0.3547093	0.3637461

Table 3. Reconstruction error on the *Sphere* dataset. The statistic shows the error between ground truth and the reconstruction using PSR (top) and FSSR (bottom). The distances have been obtained with *Metro* [3] and scaled for readability (factor 10^3).

PSR [9]	max	mean	RMS
LINEAR	3.837804	0.321320	0.467301
SCALING	3.137207	0.282015	0.369597
LSDERIV	3.137207	0.282022	0.369603
CUBIC	3.137207	0.281913	0.369466
FSSR [5]	max	mean	RMS
LINEAR	6.066898	0.659474	0.861424
SCALING	5.913879	0.485656	0.622890
LSDERIV	5.921924	0.434055	0.561907
CUBIC	6.012503	0.391485	0.513969

Table 4. Reconstruction error on the *Blob* dataset. The statistic shows the error between ground truth and the reconstruction using PSR (top) and FSSR (bottom). The distances have been obtained with *Metro* [3] and scaled for readability (factor 10^4).

Figure 9. *Stanford Bunny:* Geometric difference between LINEAR and CUBIC interpolation with PSR (left) and FSSR (right).

Blob Dataset: We also evaluate the different interpolation approaches on the *Blob* dataset, which exhibits more interesting curvature changes. The reconstruction errors are visualized in Figure 8. Similar to the *Sphere* dataset, linear interpolation leads to strong ringing artifacts and larger errors, see Table 4.

It is noteworthy that, with PSR, the quality improvement from LINEAR to non-linear interpolation is substantial. However, which higher-order interpolant is used barely makes a difference. This is because PSR represents the implicit function as the sum of second-order B-splines, so all interpolants reproduce the quadratic function along the edge. For FSSR, the CUBIC interpolation improves mean and RMS error as well as the visual appearance, although the maximum error can remain large.

5.2. Scanner and MVS Data

Next, we evaluate the interpolation methods on real-world scanner and MVS data. Because a ground truth model is not available for this data, we focus on a visual comparison between the LINEAR and the CUBIC interpolation. Note that visually, all higher-order methods produce results that are almost indistinguishable.

Stanford Bunny: We reconstructed the Stanford Bunny using PSR and FSSR with both, LINEAR and the CUBIC interpolation. The geometric difference between the two methods is visualized in Figure 9. This difference is presumably caused by the improved fitting with the CUBIC interpolant, and the ringing artifacts of the linear interpolation method become clearly visible.

Miniature City: The miniature city is a Multi-View Stereo dataset with 76 input images and has been reconstructed with the publicly available *Multi-View Environment* [6]. The resulting point cloud with 4,627,606 samples was then used as input for PSR and FSSR with the LINEAR and CUBIC interpolation method. The geometric improvement is visualized in Figure 10 and clearly visible even without color coding.

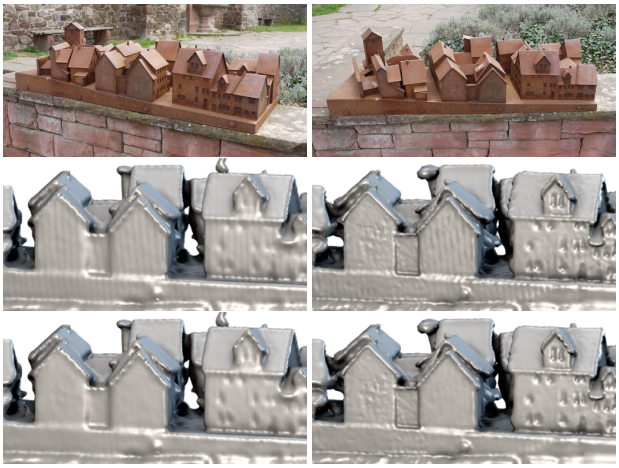

Figure 10. *Miniature City:* 2 out of 76 input images (top). Geometric difference between LINEAR (middle) and CUBIC (bottom) interpolation with PSR (left) and FSSR (right).

5.3. MRI Data

Brain: We compared the LINEAR and the CUBIC interpolation on an MRI scan of a brain obtained from the OASIS MRI database [21] (resolution $182 \times 218 \times 182$). Since the dataset comes without gradients, finite differences are used to estimate them. In Figure 11 we provide results including contrast-enhanced renderings to highlight the artifacts caused by the linear method, which are otherwise hard to visualize.

6. Conclusion

We presented Hermite interpolation for Marching Cubes-like algorithms to eliminate the majority of the artifacts that occur when contouring non-linear implicit functions with traditional linear interpolation. The extracted triangle meshes are guaranteed to have the same connectivity as the meshes extracted with traditional Marching-Cubes, but the accuracy of the isovertex positions is improved.

The proposed interpolation methods, particularly the quadratic ones, are simple to implement and can be applied to a wide range of surface extraction algorithms. The computational overhead of the quadratic methods is insignificant and in fact barely measurable. The cubic interpolation increases the total surface extraction time by about 3% with our implementation.

We have demonstrated the applicability of Hermite interpolation on PSR and FSSR and show that when gradients cannot be computed analytically, the finite differences approximation is still successful in removing the artifacts. Any of the non-linear interpolation methods substantially

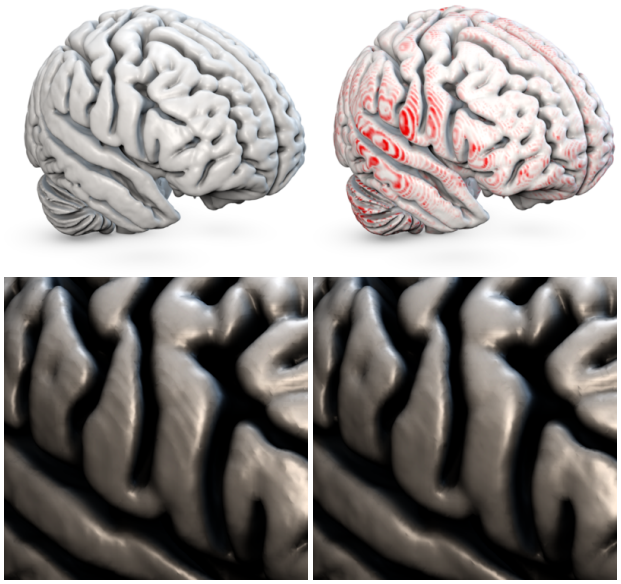

Figure 11. *Brain*: The brain isosurface with `CUBIC` interpolation (top left) and the error distance compared to linear interpolation (top right). High-contrast close-ups (bottom) of the linear method (left) and the cubic method (right) show the ringing caused by linear method.

increases surface accuracy, but "the right" method depends on the application: For example, in PSR, the quadratic methods provide sufficient accuracy while in FSSR, cubic interpolation leads to further improvement.

Acknowledgements

Part of the research leading to these results has received funding from the European Commission's FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D) and ICT-611089 (CR-PLAY).

References

[1] J. Bloomenthal. Polygonization of Implicit Surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988. 1, 2

[2] E. Catmull and R. Rom. A Class of Local Interpolating Splines. In *Computer Aided Geometric Design*, pages 317 – 326. 1974. 4

[3] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring Error on Simplfied Surfaces. *Compututer Graphics Forum*, 17(2):167–174, 1998. 4, 6

[4] A. Doi and A. Koide. An Efficient Method of Triangulating Equi-Valued Surfaces by using Tetrahedral Cells. *IEICE Transactions on Information and Systems*, 74(1):214–224, 1991. 1, 2

[5] S. Fuhrmann and M. Goesele. Floating Scale Surface Reconstruction. In *Proceedings of ACM SIGGRAPH*, 2014. 2, 5, 6

[6] S. Fuhrmann, F. Langguth, and M. Goesele. MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, 2014. 7

[7] R. S. Gallagher and J. C. Nagtegaal. An Efficient 3-D Visualization Technique for Finite Element Models and Other Coarse Volumes. In *Proceedings of ACM SIGGRAPH*, pages 185–194, 1989. 2

[8] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual Contouring of Hermite Data. *ACM Transactions on Graphics*, 21(3), July 2002. 2

[9] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, New York, New York, USA, 2006. 2, 5, 6

[10] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, June 2013. 2, 5

[11] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe. Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Eurographics Symposium on Geometry Processing*, pages 125–133, 2007. 1, 2

[12] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature Sensitive Surface Extraction from Volume Data. *Proceedings of ACM SIGGRAPH*, D:57–66, 2001. 2

[13] D. H. U. Kochanek and R. H. Bartels. Interpolating Splines with Local Tension, Continuity, and Bias Control. In *Proceedings of ACM SIGGRAPH*, pages 33–41, 1984. 3

[14] V. Lempitsky. Surface Extraction from Binary Volumes with Higher-Order Smoothness. *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 1197–1204, 2010. 2

[15] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH*, pages 163–169, 1987. 1, 2

[16] J. Manson and S. Schaefer. Isosurfaces Over Simplicial Partitions of Multiresolution Grids. *Computer Graphics Forum*, 29(2):377–385, 2010. 2

[17] A. Marinc, T. Kalbe, M. Rhein, and M. Goesele. Interactive Isosurfaces with quadratic C^1 Splines on Truncated Octahedral Partitions. *Information Visualization*, 11(1):60–70, 2012. 2

[18] Michael Kazhdan. Poisson Surface Reconstruction Code. http://www.cs.jhu.edu/ misha/Code/PoissonRecon/. 2, 5

[19] H. Mueller and M. Stark. Adaptive generation of surfaces in volume data. Technical report, 1991. 1, 2

[20] T. S. Newman and H. Yi. A Survey of the Marching Cubes Algorithm. *Computers and Graphics*, 30(5):854–879, 2006. 2

[21] OASIS. Open Access Series of Imaging Studies. http://www.oasis-brains.org/. 7

[22] S. Schaefer, T. Ju, and J. Warren. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):610–619, May 2007. 2

[23] S. Schaefer and J. Warren. Dual Marching Cubes: Primal Contouring of Dual Grids. *Computer Graphics Forum*, 24(2):195–201, June 2005. 2

[24] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha. Topology Preserving Surface Extraction using Adaptive Subdivision. *Proceedings of Symposium on Geometry Processing*, page 235, 2004. 1, 2