# Isosurface Orientation Estimation in Sparse Grids Using Tetrahedral Splines

Brian A. Wood
University of Alabama in Huntsville
Huntsville, Alabama 35899
bwood@cs.uah.edu

Timothy S. Newman
University of Alabama in Huntsville
Huntsville, Alabama 35899
tnewman@cs.uah.edu

## ABSTRACT

One challenge in applying standard marching isosurfacing methods to sparse rectilinear grid data is addressed. This challenge, the problem of finding approximating gradients adjacent to locations with data dropouts, is addressed here by a new approach that utilizes a tetrahedral spline fitting-based strategy for gradient approximation. The new approach offers improved robustness in certain scenarios (compared to the current state-of-the-art approach for sparse grid isosurfacing). Comparative studies of the new approach's accuracy and computational performance are also presented.

## Keywords

Orientation Estimation, Volume Visualization, Isosurfaces

## 1 INTRODUCTION

One common means for visualizing scalar volumetric data is isosurfacing, which involves finding the set of locations in space where the phenomenon recorded in the dataset achieves a particular value, called the isovalue, denoted herein as $\alpha$. Isosurface visualization is a powerful approach for observing and studying the behavior of volumetric data. Isosurfacing can promote discovery in disparate applications areas, such as medical diagnosis, fluid flow studies, etc.

Well-known isosurfacing methods exist for volumetric data organized on a number of grid types [10]. Focus here is on scalar data organized on rectilinear grids, which is very common, and on isosurfacing methods applied to such grids that produce triangle meshes approximating the isosurface and assume data values are available at each grid point. However, in some applications, the data is sparse; there is not a data value available at every grid point. (Here, we will use the term sparse grid to mean a 3D rectilinear grid dataset with some missing values.) For example, data collected from sensor arrays may have missing data values when data cannot be collected at every grid point due to physical limitations. Popular isosurfacing methods for rectilinear grid data, such as the standard, marching methods of Marching Cubes and Marching Tetrahedra, require determination of a local gradient at each mesh vertex to estimate the isosurface orientation, and then use that in rendering to produce a shading that is harmonious with local data trends. When data is sparse, the schemes these methods use for estimating orientation can fail at certain locations. Thus, sparseness can make well-known isosurfacing rendering methods unable to be applied. Here, we introduce a new solution to the challenge of isosurfacing on sparse grids.



Sparse grids may be produced from a variety of sensing modalities and volume data generation methods. Data from sensor arrays, particularly ones that measure physical phenomena, has the potential to have missing data values due to sensor faults. For example, wireless 3D sensor arrays, such as those used to capture data underground [1] and underwater [17], operate under harsh conditions and can be particularly vulnerable to sensor faults. Low batteries, bad calibration, high noise, or environmental hazards can all contribute to faults in sensor arrays [11]. Conversion of 3D mesh geometry to volume data via voxelization algorithms [16] can produce datasets with data values only at grid points necessary to reproduce the original mesh. Additionally, volume data derived from point clouds or signed distance functions may not contain sufficient data to estimate data gradients at all isosurface locations, in particular the mesh vertices [12].

One prior work has proposed a work-around to the gradient (orientation) determination challenge in Marching Cubes on sparse grids. The new approach we describe here offers improved results in certain scenarios.

The paper is organized as follows. Section 2 discusses background material and related work. Section 3 de-

scribes the new approach for estimating isosurface orientation on sparse grid datasets. Section 4 provides details on rendering isosurfaces extracted from sparse grids. Section 5 provides results from experiments and comparisons to prior orientation (or normal) estimation approaches. Section 6 contains the paper's conclusion.

## 2 BACKGROUND AND RELATED WORK

The most common method [10] for isosurfacing on scalar data on rectilinear grids is the Marching Cubes (MC) algorithm. MC has been adapted by Nielson et al. [12] to allow application to rectilinear grids with missing data values (i.e., sparse grids). We describe that adaptation in Section 3.2. First, though, we describe the basic steps of MC and illustrate its failings for sparse grids.

Marching Cubes isosurfacing produces a triangle mesh representation of the isosurface by advancing cell-by-cell through the volume. In each cell, it follows three major steps. In the first step, the general topological arrangement of the isosurface mesh in the cell is determined. (Each general topological arrangement is called a "case" in this paper, reflecting the typical nomenclature of the MC literature.) Second, for topologies containing isosurface mesh facets, the mesh vertex locations in the cell are found. Third, the triangle mesh is formed by connecting vertex locations into the determined topology. An orientation vector is also determined for each vertex location.

In MC, mesh vertices are located on grid lines, with positions there found via linear interpolation. At each vertex, an orientation vector is ultimately used in rendering the produced mesh. These vectors are determined by linearly interpolating the gradients of the grid point locations bounding the grid segment containing each mesh vertex. These gradients are computed using central differencing; for grid point $(x_i, y_i, z_i)$, MC finds the gradient $\nabla f$ as:

$$\nabla f(x_i,y_i,z_i) = \left\{ \begin{array}{c} \frac{f(x_{i+1},y_i,z_i) - f(x_{i-1},y_i,z_i)}{2} \\ \frac{f(x_i,y_{i+1},z_i) - f(x_i,y_{i-1},z_i)}{2} \\ \frac{f(x_i,y_i,z_{i+1}) - f(x_i,y_i,z_{i-1})}{2} \end{array} \right\}, \quad (1)$$

where $f(x_i, y_i, z_i)$ is the scalar value at $(x_i, y_i, z_i)$.

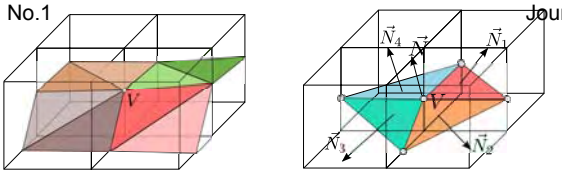
Since the central-difference gradient uses the values of adjacent grid points, if there is a missing data value preceding or following a grid point in any axial direction, central-differencing will be undefined. As a result, MC is unable to estimate the orientation vector for any mesh vertex on a grid segment whose endpoint has an undefined gradient value. Data sets with missing or undefined data thus require an alternative orientation estimator. One option could be use of ad-hoc alternatives for those grid points where central differencing is undefined. For example, a mix of methods could be used (e.g., forward-differencing and reverse-differencing, as suitable) at a cost of consistency.

Other works have considered the issue of estimating orientation in volume data without relying on differencing techniques. For example, Möller et al. [15] have used a two-step approach for shading raytraced isosurface renderings. Hossain et al. [8] have proposed reconstruction filters for gradient estimation derived from methods using Taylor series and Hilbert spaces. They evaluated the accuracy of their filters on both Cartesian and Body-Centered Cubic lattices. Correa et al. [4] have studied averaging-based and regression-based orientation estimation approaches for use in volume raycasting on unstructured grids. Their study recommended the use of a hybrid approach that selects the gradient estimator to use based on local properties of the unstructured grid. Neumann et al. [9] have estimated orientation by fitting a hyperplane on points nearby to a grid point and then taking a linear regression result on data points on the hyperplane. However, while these orientation estimation approaches do not rely on differencing, they assume that data is available at all grid points and thus cannot be used with sparse grids.

Other methods for producing visualizations of sparse grid volume data have also been described. For example, Djurcilov and Pang [6] have described some techniques for visualizing weather data when sample points are missing due to sensor failures. Their techniques require resampling data to produce a fully populated grid prior to isosurface extraction.

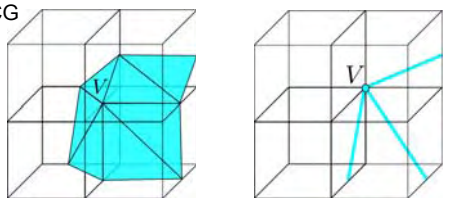
### 2.1 Quadratic and Quintic Splines

Rössl et al. [14] have proposed a technique for volume reconstruction by fitting a spline model to regular, rectilinear volumetric data. Their technique first partitions the volume's grid into uniform tetrahedra and then fits super splines on each partition. Super splines are a class of splines in which smoothness is preserved on vertices between adjacent tetrahedra. Each fitting uses Bézier splines with constants drawn from the values at the vertices of each tetrahedron, ensuring that the super spline condition is not violated. Details of their process are described later, in Section 3. Awanou and Lai [2] have presented an approach using quintic splines to interpolate a volume. Their approach is similar to that of Rössl et al., but it does not require a regular grid and uses a higher order spline to model the volume data.




(a) Marching Cubes output (b) LSN virtual local mesh

Figure 1: LSN perp vector estimation




(a) Non-degenerate mesh (b) Degenerate mesh used by LSN

Figure 2: LSN summed average estimation

## 2.2 Locally Supported Normals (LSN)

One methodology for determining isosurface orientation in sparse rectilinear grid data is the Locally Supported Normal (LSN) approach described by Nielson et al. [12]. It considers isosurfacing in a Marching Cubes context, resolving the undefined orientation problem by an estimation process that uses a virtual mesh constructed on the vertices of the MC isosurface. Orientation vectors computed in this manner tend to exhibit sharper shading color transitions at triangle edges than if central-differencing could be used, resulting in a surface with a more faceted appearance. However, central-differencing cannot be applied where grid values are missing or undefined.

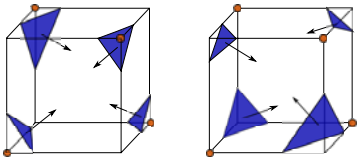
The estimation used in LSN is integrated into MC-style isosurfacing; it produces orientation estimates as vertex locations are calculated. The LSN approach relies on a temporary virtual local mesh that it defines about the point for which an orientation vector is needed. This virtual mesh is *not* the Marching Cubes output mesh; Figure 1 demonstrates the difference between a mesh produced by MC and the virtual mesh used by LSN for a point $V$. The LSN approach first computes perpendicular (perp) vectors for each face in the virtual mesh; these vertex perp operations are done independent of the MC topology determination. Each perp vector is found as the cross-product of edge vectors of the virtual mesh face. For each of the MC internal vertices shared by multiple triangles, all perp vectors of faces incident to it are averaged to form a *master perp vector* at the vertex. The master perp vector becomes the LSN's estimate of the isosurface orientation at that vertex. Figure 1(b) shows the LSN's estimation of the orientation for a location $V$ in a volume. Four perp vectors, $\vec{N}_1, \vec{N}_2, \vec{N}_3$, and $\vec{N}_4$, are shown. The average of these is the master perp vector $\vec{N}$ ; here, $\vec{N}$ is $\frac{1}{4}\sum_{i=1}^{4}\vec{N}_i$.

The LSN's estimation can produce erroneous results when certain data characteristics are encountered. The first, and most pronounced, of these errors occurs when degenerate triangles are encountered during orientation estimation. A degenerate triangle with two or more coincident vertices will yield a cross-product of zero, resulting in a zero vector (because the triangle does not lie on a unique plane in space). MC produces degenerate triangles when the isovalue is identical to a grid point value [13]. If a vertex is associated with only degenerate triangles, the orientation vector computed using LSN at that vertex has length zero. The rendered isosurface can contain artifacts at pixel locations affected by the zero length orientation vector.

In Figure 2(a), a mesh containing no degenerate triangles is shown. In contrast, Figure 2(b) displays an LSN mesh corresponding to the same topology, but with vertex $V$ located at a cube corner, resulting in four degenerate triangles (one triangle degenerating to a point and three to a line). The result from LSN is a zero length orientation vector.

Additionally, the LSN approach makes assumptions about what have been called ambiguous faces [10] of cells. These assumptions can lead to inaccurate orientation vectors. One example cube where this incorrect assumption is a problem is shown in Figure 3. The cube has the Case 13 base topology of the MC [12], shown in Figure 3(a). However, the LSN estimation uses the virtual mesh shown in Figure 3(b) to compute orientation vectors in corners of the cube opposite to those defined by MC. We refer to triangles used in the LSN virtual mesh that do not appear in the MC mesh as *illusory triangles*. The normals (i.e., perp vectors) associated with these triangles may differ greatly from the orientation vectors that would result if the actual MC isosurface facets had been used. In particular, each vector found using an illusory triangle will contribute errors to the orientation vector estimation at vertices of illusory triangles. For such situations, the orientations can be estimated incorrectly and yield incorrectly shaded renderings.

The illusory triangle problem in LSN is not just limited to cubes with ambiguous faces. For example, in Marching Cubes Case 5 LSN uses an illusory triangle to compute a perp vector. The topology used by MC for the Case 5 topology is shown at the top of Figure 4 (labeled "C5"). The five virtual mesh triangles used by LSN are shown in the rest of the figure. While most of the virtual mesh triangles should produce reasonable results, the one used for $V_4$ is illusory and its orientation is not consistent with the actual mesh properties at $V_4$. Other MC cases also exhibit illusory triangles yielding orientations that differ markedly from that of the MC isosurface mesh. An example of the incorrect orientation from illusory triangles is provided later in this work.



(a) Topology of cell as defined by Marching Cubes

(b) Topology of cell assumed by the LSN approach

Figure 3: Comparison of cell topologies used by MC and LSN

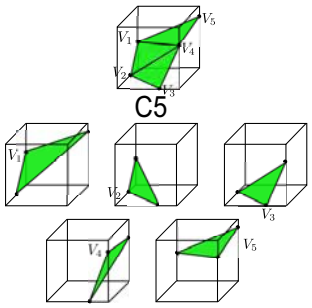



Figure 4: Case 5 MC and LSN topologies

## 3 NEW GRADIENT ESTIMATION APPROACH FOR SPARSE GRIDS

Next, we describe our new approach for determining orientation vectors in MC for sparse grids. The approach is guaranteed to produce orientation vectors at any location for which it is possible to find a Marching Cubes isosurface vertex. That is, the scheme introduced here can handle any rectilinear sparse grid configuration satisfying the condition that the isosurface vertices can be computed. (I.e., like LSN, our approach assumes there is local support for the isosurface.) For some scenarios, it also offers improved performance over prior approaches for computing MC isosurface orientation vectors on sparse grids.

### 3.1 Using Quadratic Splines

Our work is motivated by Rössl et al.'s modeling of volumetric data variation using quadratic Bézier-Bernstein super splines (2BBSS) in tetrahedral regions. A tetrahedron allows for the use of an interpolating volumetric spline using a barycentric coordinate system given a sufficient number of data points on the tetrahedron. Specifically, given four points $v_0, v_1, v_2, v_3$ defining the four vertices of a tetrahedron, a quadratic trivariate spline $p$ is composed in the Bézier-Bernstein form:

$$p(\lambda) = \sum_{i+j+k+l=2} a_{ijkl} B_{ijkl}(\lambda), \qquad (2)$$

where the parameter $\lambda$ is the location within the spline (in barycentric coordinates with $\lambda = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$), the coefficients $a_{ijkl}$ are the control points of the spline,

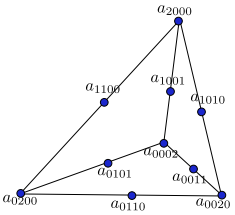



Figure 5: Spline control points

and the $B_{ijkl}$'s are Bernstein polynomials. The control points are calculated as linear combinations of the vertices of the tetrahedron:

$$a_{ijkl} = \frac{i}{2}v_0 + \frac{j}{2}v_1 + \frac{k}{2}v_2 + \frac{l}{2}v_3, \qquad (3)$$

as depicted in Figure 5. The Bernstein polynomials $B_{ijkl}$ are defined as

$$B_{ijkl}(\lambda) = \frac{2!}{i!j!k!l!}\lambda_0^i\lambda_1^j\lambda_2^k\lambda_3^l,\ i+j+k+l=2, \qquad (4)$$

where each $\lambda = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ is a barycentric coordinate with respect to the tetrahedron.

Numerous schemes exist for partitioning rectilinear grids into collections of tetrahedra. We employ one such scheme here to enable the use of tetrahedral splines in the estimation of orientation vectors. Tetrahedral partitions also alleviate the problem of missing data because only 4 grid values are needed to model isosurface behavior within a tetrahedral partition, as opposed to the 6 necessary for a central differencing. By partitioning rectilinear dataset cells into tetrahedra, we can calculate an orientation in any cell intersected by the isosurface. In the 2BBSS model, each tetrahedron must have associated data values at each tetrahedral vertex. Given such, a spline is formulated that approximates the surface within the tetrahedron.

Our approach finds the approximating spline in cells intersected by the isosurface by partitioning the cell into tetrahedra and then evaluating the spline constructed on those tetrahedra to determine orientation vectors (i.e., spline normals) at any barycentric coordinate $(\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ within each tetrahedron of interest. For each of them, our approach uses de Casteljau's algorithm [3] [5] to determine the spline's partial derivative [14] by applying the algorithm in the direction of tetrahedron edges. The usage of de Casteljau's algorithm to compute the derivative of a curve is well understood [7].

For any point on a spline, the formulation of de Casteljau's algorithm enables finding the directional derivatives at $q$ as follows. First, given a spline with control points of the form $a^0_{ijkl}$, for a point $q$ having the

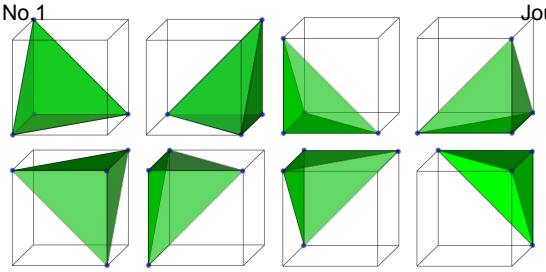
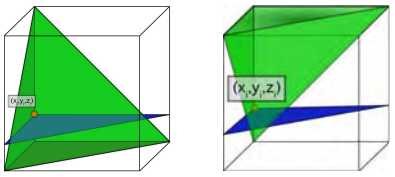


Figure 6: Tetrahedral partitions




(a) Tetrahedral partition 1

(b) Tetrahedral partition 2

Figure 7: Two choices of tetrahedral partition of the cell

barycentric coordinate $\lambda_{\mathbf{q}} = (\lambda_{0_q}, \lambda_{1_q}, \lambda_{2_q}, \lambda_{3_q})$, new control points denoted by $a^1_{ijkl}$ are computed as:

$$a^1_{ijkl} = \lambda_{0_q} a^0_{i+1,j,k,l} + \lambda_{1_q} a^0_{i,j+1,k,l} + \lambda_{2_q} a^0_{i,j,k+1,l} + \lambda_{3_q} a^0_{i,j,k,l+1}, \quad (5)$$

which define a subdivision of the original spline. (Another application of the formula would produce the value at $q$, however we need just the control points $a^1_{ijkl}$ of the spline subdivision because they define partial derivatives for the spline.) Since the normal at any point on a surface $s(x,y,z)$ can be defined as

$$\nabla s(x,y,z) = (\frac{\partial s}{\partial x}, \frac{\partial s}{\partial y}, \frac{\partial s}{\partial z}), \quad (6)$$

we compute the orientation by finding the partial derivatives in the directions parallel to the coordinate system axes. The formulation of this partial derivative is given in Section 4.

## 4 ISOSURFACE RENDERING WITH SPARSE GRIDS

Our approach defines 2BBSS splines for tetrahedral subregions of each active cell. We consider eight candidate tetrahedral partitions of each cell (shown in Figure 6) and choose from these the one that enables the most accurate estimate of the orientation vector. The choice is described shortly. This orientation estimation is based on a 2BBSS approximation of the volume within that tetrahedron. The eight tetrahedra were chosen because they share the property that three tetrehdron faces are coplanar with faces of the cell which helped simplify the construction of the spline.

For each isosurface mesh vertex, there are two candidate tetrahedra from which the orientation at that vertex could be computed. Next, how our approach decides on the one to use is described. An example situation is shown in Figure 7. In it, the vertex shown in red is located on the rear edge of the cell. One candidate tetrahedron is shown in Figure 7(a) and the other is shown in Figure 7(b). For the case where the vertex lies on an isosurface mesh triangle completely located within a tetrahedron, that tetrahedron is chosen. However, a triangle's surface may span both possible choices of tetrahedra. For such cases, tetrahedron selection is done instead by considering the total number of isosurface mesh triangle edges; we select the tetrahedron containing the greatest number of triangle edges. We have found that selection using this criterion provides more accurate orientation vectors than using a static tetrahedral partition that is ignorant of the triangles's location in the cell. Our approach uses an adaptation of the MC topological case lookup table to record the tetrahedral selections, allowing fast determination of the tetrahedron as well as supporting orientation vector determination coincident with mesh determination (i.e., within an extended MC context).

Next, we describe the orientation determination procedure. The partial derivative of the spline $p(\lambda)$ in the direction $\xi_\phi$ of a tetrahedron edge $v_\phi - v$ is given by

$$\frac{\partial p}{\partial \xi_\phi} = 2 \sum_{i+j+k+l=1} (a_{i,j+b,k+c,l+d} - a_{i+1,j,k,l}) \lambda_0^i \lambda_1^j \lambda_2^k \lambda_3^l, \quad (7)$$

where $(\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ are the barycentric coordinate variables of the spline equation and $(b,c,d)$ is used to define an offset to a tetrahedral vertex in direction $\xi_\phi$.

Figure 8 illustrates the vector calculations when finding the partial derivative in the $x$ direction. The arrows on tetrahedron edges indicate a forward difference calculation using the tetrahedron vertices of that edge. The partial derivative is a linear combination of the differences, with weights for each component dependent on the particular tetrahedral partition being used within the cell. Similar vectors are computed for partial derivatives in the $y$ and $z$ directions.

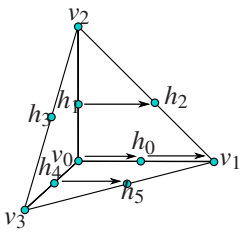



Figure 8: Computing the orientation from sample points

Since cell vertices are located on grid edges, each mesh vertex is guaranteed to have at least two zero-valued components of its barycentric coordinate. By determining which edge type the vertex is located on, we can choose the most appropriate equation to minimize the number of calculations required. For instance, for the tetrahedron shown in Figure 8, the gradient for a vertex on any edge parallel to the base is given by:

$$\nabla F = \left\{ \begin{array}{l} \gamma_0 \sum_{i+j=1} (a_{i,j+1,0,0} - a_{i+1,j,0,0}) \lambda_0^i \lambda_1^j, \\ \gamma_1 \sum_{i+k=1} (a_{i,0,k+1,0} - a_{i+1,0,k,0}) \lambda_0^i \lambda_2^k, \\ \gamma_2 \sum_{i+l=1} (a_{i,0,0,l+1} - a_{i+1,0,0,l}) \lambda_0^i \lambda_3^l \end{array} \right\}, \quad (8)$$

where $\gamma_v$, $v = 0,1,2, \gamma = \pm 1$, is an orienting coefficient. For the example in Figure 8, formulation of the spline assumes a tetrahedron oriented as in Figure 8, however the tetrahedral partition used may be a reflection or rotation (or combination of both) of this orientation. The $\gamma$ coefficient, which corrects for reflected or rotated instances, allows correcting the directions the components of the orientation vector.

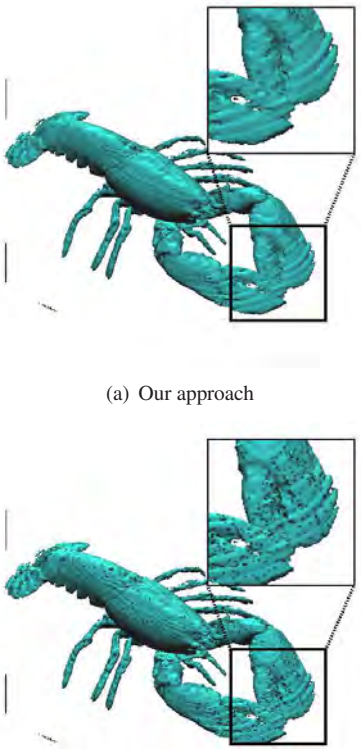
For each tetrahedron the mesh vertex is located in, a gradient vector is produced by evaluating Equation 9. Vertices will be shared among up to four tetrahedra, resulting in as many as four separate vectors per vertex. The orientation vector ultimately assigned to the vertex is the mean of these four gradient vectors.

## 5 RESULTS

In this section we present results of experiments to evaluate our approach versus LSN. These experiments consider accuracy of orientation vectors and the run times to compute them. We also report a qualitative evaluation of rendered images to determine the impact of degenerate triangles on each approach.

Accuracy was tested by comparing orientation vectors computed using our spline-derived orientations against the orientations using the LSN approach, then comparing these against orientations computed using central-differencing. Eight well-known real (sensed) volume datasets and five mathematically-defined datasets were used in testing. Additionally, we performed visual comparisons of the rendered images to determine if there was a difference between renderings made using the two orientation estimation approaches.

The datasets were converted to sparse grid representations by removing all grid values that were not required by MC to extract the isosurface with marker values. Specifically, grid points that were not on grid edges containing a mesh vertex were set to marker values. By removing all data points that do not contribute to the isosurface extraction, we could operate on volumes with the least possible number of defined values and thus the least favorable datasets for the classic central difference orientation estimation approach used in MC.



(a) Our approach

(b) LSN approach

Figure 9: Renderings performed using both orientation estimation approaches.

### 5.1 Measurement of Orientation Estimation Accuracy

Isosurfaces were extracted using Marching Cubes for ranges of isovalues on the eight sensed datasets. The range was made large so that results would not be biased against a particular sub-range of isovalues. A root mean square (RMS) error for each isosurface was calculated by comparing the angular difference (in radians) of all orientation vectors produced by both estimation approaches against the central-difference estimate. The central-difference is the baseline in this error comparison because it is equivalent to computing the gradient of a second-order data fitting at each grid point. The mean RMS error of each dataset at all tested isovalues is shown in Table 1. Inspection of individual isovalues on some datasets showed that LSN was sometimes more accurate than our approach, but on average ours appears to be the superior approach. The spline orientation estimation produced more accurate orientation estimates (on average) than the LSN approach in all datasets except for the Engine dataset.

Table 2 shows the RMS errors for 9 isosurfaces extracted on the sensed datasets. LSN does occasionally produce more accurate results, however our orientation

| Dataset | Ours | LSN |
|---|---|---|
| Foot | 0.531 | 0.547 |
| Frog | 0.569 | 0.589 |
| Lobster | 0.369 | 0.375 |
| MRA | 0.639 | 0.653 |
| Piggy bank | 0.876 | 0.898 |
| Backpack | 0.561 | 0.568 |
| Sheep heart | 0.313 | 0.315 |
| Engine | 0.204 | 0.187 |

Table 1: Average RMS error of approaches vs. central-difference

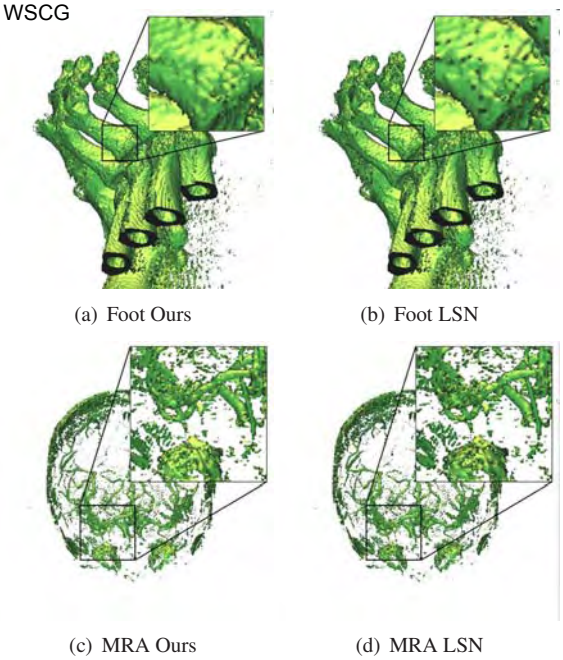
| Dataset | Isovalue | Ours | LSN |
|---|---|---|---|
| MRA | 65 | 0.740 | 0.764 |
| | 75 | 0.684 | 0.714 |
| | 80 | 0.775 | 0.783 |
| Foot | 80 | 0.555 | 0.572 |
| | 90 | 0.502 | 0.518 |
| | 100 | 0.472 | 0.322 |
| Frog | 40 | 0.512 | 0.524 |
| | 45 | 0.513 | 0.523 |
| | 80 | 0.545 | 0.640 |
| Lobster | 50 | 0.318 | 0.316 |
| | 65 | 0.329 | 0.332 |
| | 80 | 0.336 | 0.338 |

Table 2: RMS error of approaches vs. central-difference

estimation produces more accurate results in the majority of cases we tested. Figure 10 shows MRA and Foot isosurfaces (for $\alpha = 65$ and 90, respectively). The magnified callouts show subtle differences in the two renderings, but both are very similar to the baseline images produced using central-difference gradient estimates.

## 5.2 Accuracy using Mathematically Defined Data

Experiments were also performed to measure the accuracy of the orientation estimation approaches versus exact orientation vector values. These experiments tested scalar fields generated using five mathematically defined fields. The isosurfaces were generated corresponding to level sets (i.e., implicit surfaces) of these fields. Orientation vectors were estimated using our spline-based estimation, the LSN estimation, and the standard MC central-difference approaches. Orientation vectors at each location were compared against the exact orientation vector values computed at the isosurface intersection locations. Table 3 reports the RMS error with respect to the exact orientation vectors for isosurfaces of the zero level set. Excepting the Marschner-Lobb dataset, the central-difference estimates are superior to both LSN and our orientation estimations. But LSN estimates are sometimes better than ours. Thus, empirical evidence suggests that, for mathematically



(a) Foot Ours (b) Foot LSN

(c) MRA Ours (d) MRA LSN

Figure 10: Zoomed comparison of isosurface images

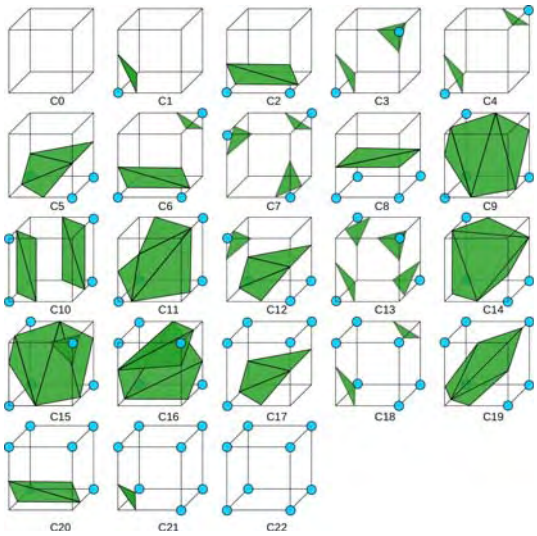



Figure 11: MC lookup table base topologies

defined, noise-free data, LSN estimation may be quite suitable; LSN estimation may provide more accurate normal estimation than our approach for many mathematically defined scalar fields.

## 5.3 Individual MC Topologies

Since results for the mathematically defined datasets were incongruous with those observed for sensed data (where our approach appears to be better than LSN), we performed an analysis of occurrences of MC base topologies defined in the MC lookup table [12] to determine if one estimation approach produced more accurate orientation vectors for particular base topologies.

| Dataset | Ours | LSN | Central Diff |
|---|---|---|---|
| Mar-Lobb | 0.0545 | 0.0616 | 0.0718 |
| Six Peaks | 0.0139 | 0.0534 | 0.0179 |
| Genus_3 | 0.00622 | 0.00506 | 0.000265 |
| Flower | 0.0261 | 0.0261 | 0.0183 |
| Peaks | 0.0372 | 0.0235 | 0.0218 |

Table 3: RMS error calculated versus exact orientations

The base topologies are shown in Figure 11 with labels $C_i$ ($C_i$ means "Case i", as used throughout this section). The isosurfaces extracted from mathematically defined datasets showed no occurrences of the Case 4, 7, 12, and 15 topologies. Additionally, very low occurences were observed for Cases 6, 10, 11, 12, 14, 15, 18 and 19. Many of these topologies consist of disconnected triangles within a cell. Due to the nature of the level sets MC produced for these datasets it is not unexpected that occurrences of these topologies would be rare. The sensed data contained far more examples of these topologies. While for some isovalues, there were no instances of a few topological cases, such situations were observed less frequently than for the synthesized datasets. For one dataset (MRA), some isovalues did not give rise to any cells of the type Case 15 or 18. For one dataset (the Engine dataset), the majority of isovalues did not give rise to any of Case 4, 7, 13, or 15 cells. This may be a result of the engine structure in the dataset being manufactured from a CAD model that had a limited number of basic surface types.

To determine the effect that particular base topologies had on orientation estimation accuracy, we considered RMS error of orientation vectors on a topological basis for sensed data isosurfaces. The Case 7, 10 12, 13, 15, and 19 topologies demonstrated much lower RMS errors for our estimation than for LSN estimation. LSN estimation produced consistently more accurate orientation vectors for the Case 8 topology. These results suggest that LSN estimation be considered for isosurfaces likely having low occurrences of topologies beneficial to our approach; mathematically defined datasets similar to ones tested here may be good for LSN.

The LSN's orientation vectors can differ substantially from true orientation vectors and from orientation vectors calculated using central-differencing, as demonstrated in Figure 3. We also analyzed the degree each case should be considered "at-risk" of exhibiting errors due to the incorrect topology assumption, focusing on error-prone vertices. Our criterion for this analysis was if angular divergence in the vector was 90 degrees or more from the central-difference orientation vector. We considered only vertices at the midpoint of cell edges. The analysis showed that 146 of the 256 possible MC cases were potentially problematic. One to five vertices demonstrated angular divergence greater than 90 degrees in these cases. Error comparisons of orientation vectors for just the problematic cases are reported in Table 4 over an average of 100 isovalues for each dataset. Our approach produces orientations that are closer to the central-difference than LSN when these cases are encountered. Figure 9 shows isosurface renderings for the Lobster dataset using both approaches. However, the incidence of orientations that meet the angular divergence criterion in sensed and simulation data is likely much smaller since the triangle vertex locations in the analysis were chosen to highlight the problematic cases and the severity in angular difference is lessened when vertices are located closer to grid point locations.

| Dataset | Ours | LSN |
|---|---|---|
| MRA | 0.639 | 0.651 |
| Foot | 0.922 | 0.933 |
| Frog | 0.652 | 0.689 |
| Lobster | 0.479 | 0.478 |

Table 4: RMS error for problematic cases

| Dataset | Isoval. | # undef. | Total | % |
|---|---|---|---|---|
| Foot | 40 | 12204 | 278894 | 4.36 |
| Frog | 40 | 1263 | 101841 | 1.24 |
| Lobster | 40 | 2946 | 149250 | 1.97 |
| Engine | 40 | 4704 | 637854 | 0.74 |
| Mar-Lobb | 0 | 0 | 603343 | 0 |
| Six Peaks | 0 | 8 | 2004650 | 0 |

Table 5: Undefined orientations using LSN approach

| Dataset | Ours (secs) | LSN (secs) |
|---|---|---|
| Flower | 1.077 | 2.873 |
| Six Peaks | 0.926 | 2.428 |
| Mar-Lobb | 2.959 | 8.018 |

Table 6: Orientation estimation times

Rendering artifacts at degenerate triangles in the isosurface mesh can be observed in Figure 9(b). They manifest here as dark spots and are a result of using a vector cross product to compute orientation vectors on degenerate triangles in the virtual mesh. (MC produces a triangle with three coincident vertices when a grid value is identical to the isovalue.) Here, the orientation vector computed for this triangle has length equal to zero. The zero-length vector leads to a zero vector for the Phong illumination diffuse and specular components . Our method does not exhibit this phenomenon, as is illustrated in Figure 9(a), since our orientation vector relies on the result of a fitting to four data values within the cell rather than on any mesh triangles.

In Table 5, we show the number of undefined orientation vectors recorded using the LSN estimation. Volumes with 8 bit integers had more undefined orientations than did those with floating point values. Far fewer undefined orientations were present in the syn-

thetic datasets, which all used 32 bit floating point numbers to store the volume's sample values. The number of undefined orientation vectors using LSN estimation appeared to correspond to the data type used to store the volume's data values.

Finally, in Table 6 execution times for calculating orientations for three of the larger datasets are shown. The LSN estimation requires over twice the computation of our approach. The LSN approach is not as fast as ours.

## 6 CONCLUSION

We have presented a new approach for estimating isosurface orientation vectors on sparse grid datasets. The typical approach for orientation estimations, central-differencing, cannot be used universally in sparse grids due to undefined data at some grid locations. Our approach can produce isosurface orientations anywhere that MC can produce triangles. Further, the approach is not affected by the presence of degenerate triangles, which produce shading errors in other approaches as a result of undefined orientations. Thus, the new approach has certain advantages even over MC's orientation estimation. Our approach has, on average, a smaller RMS error than a competing approach (using the baseline of central-difference estimations) on real world data. For synthetic data, advantages were less clear. Computation times for our approach were markedly faster. Further, the new approach guarantees orientation vectors to be defined at all vertex locations, making it applicable to a wider variety of data.

An area for further investigation is using spline fittings that observe the continuity properties of super splines in producing more accurate orientation estimations. Also, other isosurfacing algorithms could be investigated with our approach to estimate orientations to determine what increases in accuracy and error tolerance occur. Another area of further investigation is removing random data grid values to simulate random sensor failures. Lastly, we will evaluate the impact of increasing noise levels on the new approach's accuracy.

## 7 REFERENCES


[1] T. Abdoun, V. Bennett, L. Danisch, T. Shantz, and D. Jang. Field installation details of a wireless shape-acceleration array system for geotechnical applications. In *14th Int'l Symp. on: Smart Structures and Materials & Nondestructive Evaluation and Health Monitoring*. Int'l Society for Optics and Photonics, 2007.

[2] G. Awanou and M.-J. Lai. $C^1$ quintic spline interpolation over tetrahedral partitions. *Approximation Theory X: Wavelets, Splines, and Apps.*, pages 1–16, 2002.

[3] W. Boehm and A. Müller. On de Casteljau's algorithm. *Computer Aided Geometric Design*, 16:587–605, 1999.

[4] C. D. Correa, R. Hero, and K.-L. Ma. A comparison of gradient estimation methods for volume rendering on unstructured meshes. *IEEE Trans. on Visualization and Computer Graphics*, 17(3):305–319, 2011.

[5] P. de Casteljau. Courbes et surfaces á poles. In *Andres Citröen, Automobiles SA*, Paris, 1963.

[6] S. Djurcilov and A. Pang. Visualizing sparse gridded data sets. *IEEE Computer Graphics and Apps.*, 20:52–57, 2000.

[7] G. Farin. *Curves and Surfaces for CAGD*. Morgan Kaufmann, 5 edition, 2001.

[8] Z. Hossain, U. R. Alim, and T. Möller. Toward high-quality gradient estimation on regular lattices. *IEEE Trans. on Visualization and Computer Graphics*, 17(4):426–439, 2011.

[9] L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4d linear regression. *Computer Graphics Forum*, 19(3):351–358, 2000.

[10] T. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, October 2006.

[11] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor network data fault types. *ACM Trans. on Sensor Networks (TOSN)*, 5(3):25, 2009.

[12] G. M. Nielson, A. Huang, and S. Sylvester. Approximating normals for marching cubes applied to locally supported isosurfaces. In *Proc., Visualization '02*, pages 459–466, 2002.

[13] S. Raman and R. Wenger. Quality isosurface mesh generation using an extended marching cubes lookup table. *Computer Graphics Forum*, 27(3):791–798, May 2008.

[14] C. Rössl, F. Zielfelder, G. Nurnberger, and H.-P. Siedel. Spline approximation of general volumetric data. In *Proc., Symp. on Solid Modeling and Apps.*, pages 71–82, 2004.

[15] K. M. T. Möller, R. Machiraju and R. Yagel. A comparison of normal estimation schemes. In *Proc., Visualization '97*, pages 19–26, 1997.

[16] S. W. Wang and A. E. Kaufman. Volume sampled voxelization of geometric primitives. In *Proc., Visualization '93*, pages 78–84, 1993.

[17] Y. Wang, Y. Liu, and Z. Guo. Three-dimensional ocean sensor networks: a survey. *Journal of Ocean University of China*, 11(4):436–450, 2012.