SHREC: SHarp REConstruction of isosurfaces

Arindam Bhattacharya, Ross Vasko and Rephael Wenger

ABSTRACT

We present an algorithm, called SHREC, for constructing isosurfaces with sharp edges and corners. Algorithm SHREC computes isosurface vertex locations on sharp features, selects a well-spaced subset of these vertices, and merges isosurface vertices in the neighborhood of each selected vertex. The resulting isosurfaces have good representations of sharp features. Algorithm SHREC is similar to a previous algorithm, MergeSharp, but improves on MergeSharp by better generation of isosurface vertices, better selections of vertices on sharp features and better choices in merging. We also define a function to measure the closeness of the normals between two surfaces and use this function to evaluate the quality of reconstructions of surfaces with sharp features.

Keywords: Isosurface, sharp features, industrial CT.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

1 INTRODUCTION

Given a regular grid sampling of a scalar field, $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, and a scalar value σ, we are interested in constructing a good mesh representation of the level set $f^{-1}(\sigma) = \{x : f(x) = \sigma\}$. Such a representation is called an *isosurface* and the scalar value σ is called an *isovalue*.

When f is smooth, i.e., when f is continuous and its derivatives are continuous, a number of algorithms do an excellent job of isosurface construction. However, if the gradient field of f is discontinuous, then the problem of isosurface construction is more challenging. If the level set $f^{-1}(\sigma)$ intersects a gradient discontinuity, then the level set $f^{-1}(\sigma)$ will have a "sharp corner" (a 0-dimensional feature) or a "sharp edge" (a 1-dimensional feature) at that discontinuity. We are interested in constructing isosurfaces which do a good job of representing those sharp features.

Isosurfaces are represented by piecewise linear or piecewise smooth meshes, usually composed of triangles or quadrilaterals. This underlying mesh should model the sharp features of the isosurface Σ. A 1-dimensional feature of Σ with dihedral angle α should be represented by a single, connected sequence of mesh edges with dihedral angle near α. A 0-dimensional feature of Σ with solid angle α should be represented by a single isosurface vertex with solid angle near α. On the other hand, mesh edges and vertices representing smooth, low curvature portions of Σ should have dihedral angles near 180 degrees and solid angles near 2π.

A number of algorithms have been proposed for constructing isosurfaces with sharp features [2, 17, 18, 20, 22, 25, 34, 32, 37]. Unfortunately, these algorithms create "notches" along sharp edges, degenerate, zero area triangles or quadrilaterals, and "folds" in the mesh with "flipped" triangles. (See Figure 1 for illustrations of these problems.)

In [1], we described an algorithm, MergeSharp, for constructing isosurfaces with sharp features based on merging grid cubes around sharp features. By placing a single isosurface vertex in these merged regions, MergeSharp significantly reduces problems of "notches", degenerate mesh polygons and "folds" in the mesh. While MergeSharp reduced the mesh problems, it did not eliminate them, with many meshes still having one or two locations with degenerate mesh polygons or folds in the mesh.

In this paper, we present an algorithm, SHREC, which almost completely eliminates the mesh problems listed above. The algorithm is based on the MergeSharp paradigm, where grid cubes around sharp features are merged into a region containing a single isosurface vertex. However, it differs from MergeSharp by better generation of isosurface vertices on sharp features, better selection of isosurface vertices on the sharp features, and more controlled merging of grid cubes. The algorithm performs measurably better than MergeSharp or than other algorithms for which we had implementations.

There is extensive work on construction of surfaces with sharp features from point cloud data, e.g. [4, 8, 10, 12, 14, 28, 29, 35] and many other articles. In [28] and [35], the final construction of the surface mesh is accomplished by Marching Cubes [24] or some other isosurface reconstruction algorithm. The algorithms in this and similar papers can be used by algorithms in those papers to construct a surface mesh that better represents sharp surface features than the Marching Cubes isosurface.

In [8, 12] and [29], the final construction of the surface mesh is accomplished by Voronoi based algorithms described in [8] and [12]. These algorithms could be applied to isosurface reconstruction from regular grid scalar data by constructing point clouds from the grid data and then applying the algorithms to the point clouds. We compare isosurfaces with sharp features constructed using the algorithm from [12] with isosurfaces constructed by our algorithm.

Few of the isosurface or point cloud reconstruction papers provide quantitative analysis of the resulting surfaces. Some provide a distance measure (Hausdorff or root-mean squared) between the constructed surface and an ideal surface, but such measures fail to capture errors in surface normals of the mesh polygons. Instead of quantitative measures, most papers provide a few images for visual inspection of the quality of reconstructed features. Needless to say, the lack of quantitative measures makes it difficult evaluate the quality of the algorithms results, to compare different algorithms, or to comprehensively test an algorithm on a large number of data sets.

In contrast, the Hausdorff metric and variants are excellent quantitative measures of the difference between reconstructed and ideal smooth surfaces. Software tools such as Metro [9] and MESH [3] are commonly used to measure the quality of surface reconstructions and provide quantitative evaluations of reported results. However, these tools completely ignore surface orientation and are not suitable for evaluating reconstructed features.

In this paper, we present the angular distance as a measure of the difference between the surface orientation of two meshes. Essentially, the angular distance measures the difference between the surface normal at a point on one mesh with the surface normals on nearby points on the other mesh. The angular distance is defined and discussed in Section 12. We also provide a software tool which measures the angular distance.

In this paper, we are using the term "feature" to refer to a "significant" discontinuity in surface normals, not a region with higher curvature. Most of the papers on reconstruction from point clouds don't distinguish between discontinuities in surface normals and surface points with high curvature. When isosurfaces are reconstructed from scalar data on a regular grid, the sampling density is

fixed and it is impossible to determine if the original surface has a discontinuity in surface normals or very high curvature. Thus, we assume that features in our ideal surface are points or curves where the surface normal is discontinuous.

Contributions

The major contribution of our work is an isosurface generation algorithm called SHREC which does measurable better than previous algorithms in constructing isosurfaces with sharp features. In particular, SHREC does better at generating and selecting vertices on sharp features and in avoiding constructing degenerate or small angle triangles and flipping triangles. A second contribution is a definition of the angular distance between two surfaces and the application of the angular distance to evaluate the quality of surface reconstruction algorithms.

2 Definitions

A grid edge, facet or cube is *active* if it contains at least one vertex with scalar value greater than or equal to the isovalue and at least one vertex with scalar value less than the isovalue.

A fold in a mesh is a mesh edge **e** with dihedral angle near zero, so that the two mesh triangles incident on **e** are almost positioned on top of one another. (See Figure 1(d).)

3 Related Work

The well-known Marching Cubes algorithm [24] by Lorensen and Cline constructs isosurface patches within active grid cubes. The isosurface patches align along their common boundaries. Because Marching Cubes positions vertices only on grid edges, never inside grid cubes, it does a poor job at representing sharp features on isosurfaces.

Dual contouring algorithms construct an isosurface using quadrilaterals which are dual to active grid edges. The isosurface vertices are located within the grid cubes, not on grid edges. Gibson [15, 16] gave the first dual contouring algorithm which she called surface nets. Because Gibson's algorithm placed only one isosurface vertex in each active cube, it produced many isosurface edges contained in four quadrilaterals. Thus, the resulting isosurface was not a manifold. Nielson [26] modified Gibson's algorithm to allow multiple isosurface vertices in active cubes. The number of vertices in a grid cube **c** corresponds to the number of isosurface patches in **c** produced by the Marching Cubes algorithm.

Nielson's algorithm eliminates most, but not all, of the non-manifold problems in dual contouring. A dual contouring algorithm for constructing an isosurface which is a always a manifold is contained in [36]. The algorithm is essentially Nielson's algorithm but the number and connectivity of isosurface vertices in grid cubes is sometimes modified.

In [23], Lindstrom gave an algorithm for locating a point on a surface from a set of n tangent planes. The n tangent give a set of n equations in three unknowns, described by $Mx = b$ where M is an $n \times 3$ matrix and b is a column vector of length n. Lindstrom uses the singular valued decomposition (SVD) of M to determine a point close to all the tangent planes. The SVD of M also indicates whether the point is on a 0-dimensional (corner) or 1-dimensional (edge) surface feature or on a smooth portion of the surface. Lindstrom's algorithm is described in more detail in Section 6.1. All the papers for isosurface construction with sharp features use Lindstrom's algorithm or some variation to locate points on sharp features. Most of them also use Lindstrom's algorithm to identify sharp features and classify them as 0 or 1 dimensional.

Kobbelt et al. [22] modified the Marching Cubes algorithm to position vertices inside grid cubes when those grid cubes intersect sharp features. They called their algorithm Extended Marching Cubes. In addition to scalar values at regular grid vertices, Extended Marching Cubes requires a directed distance field representing the distance along each axis to the modeled surface. It also requires the surface normals at the intersection of the grid edges and the surface. Extended Marching Cubes uses the directed distance field to locate points on the intersection of grid edges and the isosurface. It uses the surface normals to construct tangent planes intersecting a grid cube and computes a point on the sharp features from those tangent planes.

Ju et al. [20, 31] present a dual contouring algorithm for constructing isosurfaces with sharp features. Vertices of dual contouring isosurfaces lie inside grid cubes while isosurface quadrilaterals are dual to grid edges. In addition to scalar values at regular grid vertices, the algorithm by Ju et al. requires surface normals at the intersection of the grid edges and the surface. As in [22], the surface normals are used to construct tangent planes intersecting a grid cube. Ju et al. apply Lindstrom's algorithm to the tangent planes to locate an isosurface vertex in each grid cube and determine if the vertex lies on a sharp 0 or 1 dimensional feature. Variations on [20] are given in [37, 34].

Ashida and Bandler [2], Ho et al. [18] and Greß and Klein [17] give algorithms for constructing multiresolution isosurface with sharp features using oct-trees or kd-trees. The isosurface mesh is constructed by first constructing polygonal curves representing the intersection of the isosurface and oct-tree or kd-tree cells, and then connecting an isosurface vertex with those polygonal curves. As in [20], isosurface locations are computed from input surface normals using Lindstrom's algorithm.

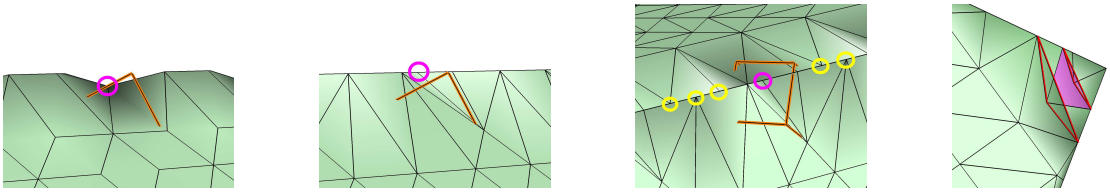
The Dual Marching Cubes[1] algorithm by Schaefer and Warren [32] constructs a dual mesh which aligns with sharp features and then extracts the isosurface from that mesh using Marching Cubes. Vertices of the dual mesh are positioned on sharp features using Lindstrom's algorithm. Because the isosurface is extracted using Marching Cubes, the isosurface will have many "sliver" triangles with small angles. Dual Marching Cubes reduces the number of "sliver" triangles by positioning the vertices of the dual grid to lie on the isosurface, whenever possible.

With the exception of [18] and [22], none of the reconstruction papers listed above give any quantitative analysis of the reconstructed features. Kobbelt et al. [22] give the "approximation error" of their reconstructions to two original models. Ho et al. [18] measured the geometric distance between the boundary of the union of three random tetrahedra and the reconstructed surface produced by their algorithm. Unfortunately, neither Kobbelt et al. nor Ho et al. explain exactly what they mean by "approximation error" or "geometric distance", but these values are probably the Hausdorff distance or some distance averaged over the surface. In any case, they do not measure the correspondence between surface normals.

The algorithms described above produce "sliver" triangles with very small angles along the 1-dimensional features. A small perturbation of a vertex of such triangles has a large effect on the triangle normal direction, so the normal direction of such triangles is almost arbitrary. The triangle angles may be so small that the triangles are effectively degenerate with zero area.

If grid cube **c** is on or near a sharp feature, Extended Marching Cubes and the dual contouring algorithms generate an isosurface vertex for cube **c** by computing a point close to a set of the tangent planes. What happens if the point is not inside **c**? The problem is discussed in [31] where various modifications are proposed for increasing the likelihood of generating a location inside **c**. However, a sharp edge (1-dimensional feature) may intersect an inactive grid cube and a sharp corner (0-dimensional feature) may lie in an inactive grid cube. In such cases, if isosurface vertices are required

[1] Nielson also calls his dual contouring algorithm "Dual Marching Cubes" [26], but it is totally different from Schaefer and Warren's algorithm.



(a) Notch in 1D feature (b) Not clamped to cube (c) Degenerate triangles (d) Folds in mesh

Figure 1: Reconstruction problems. (a) The orange cube generates the vertex in the magenta circle. Clamping the vertex to the orange cube creates a notch in the 1-dimensional feature. (b) Same mesh as (a) but vertex in the magenta circle is not clamped to the orange cube. (c) Different view of mesh, vertex and cube in (b). The vertex in the magenta circle is part of a degenerate triangles which lies on the 1D feature. Vertices in the yellow circles are also part of degenerate triangles which lie on the 1D feature. Note that the mesh is fully triangulated so that the apparent mesh quadrilaterals are actually mesh triangles adjacent to degenerate triangles. (d) Mesh folds produced by Extended Marching Cubes near a cube corner. The fold edges are at highlighted in red. The magenta triangle overlaps six other mesh triangles.

to lie within active cubes, the isosurfaces will contain notches or truncated corners. (See Figure 1(a).)

Zhang et al. in [37] state that the isosurface vertex generated by cube $\mathbf{c}$ may be placed outside cube $\mathbf{c}$. This solves the problem of notches (Figure 1(b)) or truncated corners, but at the cost of exacerbating the problem of sliver and degenerate triangles (Figure 1(c)). As noted in [31] placing vertices in inactive grid cubes also introduces a new problem of folds in the mesh (Figure 1(d)).

The MergeSharp algorithm by Bhattacharya and Wenger [1, 6] attacks the problem of sliver triangles, notches, and folds by merging grid cubes around features so that isosurface vertices on features are well-separated from each other and from non-feature vertices. Isosurface vertices are permitted to be placed in inactive grid cubes.

Bhattacharya and Wenger analyzed their algorithm by extracting all "sharp" mesh edges with dihedral angle below a threshold (140°), forming a graph (1-skeleton) from those edges and counting the number of vertices with degrees other than two. For instance, the 1-skeleton from the sharp mesh edges in the reconstruction of a cube should have eight vertices with degree three and no vertices with degrees other than two or three. The 1-skeleton from the sharp mesh edges in the reconstruction of a thickened annulus should have no vertices with degree other than two. By counting the difference between the expected and the actual degree counts, Bhattacharya and Wenger gave a quantitative measure of the performance of their algorithm.

We mention only a few papers from the large literature on surface and feature reconstruction from point sets. Point set data is inherently noisy, so much of the literature focuses on finding the true position of points on surfaces. Daniels et al. [10], Fleishman et al. [14] and Oztireli et al. [28] construct local surface patches fitted to local sets of points and project points onto these surface patches. Wang et al. [35] construct approximations of the tangent planes at the sample points and project points onto these tangent planes. Avron et al. [4] estimate surface normals at sample points using convex optimization, and then reposition the points, again using convex optimization.

The papers cited above focus on correct positioning of surface points in the presence of sharp features. The actual construction of the surface mesh is left to preexisting algorithms. For constructing the surface mesh, Oztireli et al. and Wang et al. use Marching Cubes, Daniels et al. use the advancing front algorithm from [33], and Avron et al. use the Ball Pivoting algorithm from [5]. (Wang uses Poisson Surface Reconstruction described in [21] but that algorithm uses a variation of Marching Cubes.) Neither Marching Cubes nor the Ball Pivoting algorithm is particularly well-adapted to constructing meshes with good representations of sharp features. By starting from the sharp features, Schreiner et al. claim that their advancing front method can do a good job of representing sharp features.

Two algorithms, one by Dey et al. [12] and one by Salman et al. [29], first identify and select points on sharp features and then reconstruct the surface mesh from the selected feature points and a subset of the smooth points. The algorithms differ in how they identify points on sharp features. Dey et al. use the graph Laplacian to identify points on sharp features while Salman et al. use an analysis of the shape of Voronoi cells.

Both algorithms use the "protecting ball" technique from [8] to construct the surface mesh from the weighted Delaunay triangulation of the selected feature points and a subset of the smooth points. The meshing algorithm ensures that surface vertices are well-spaced along feature curves and that surface vertices which are not on feature curves are suitably far from those curves. The method requires constructing and updating the weighted Delaunay triangulation of a set of points. Dey et al.'s algorithm works on non-manifold surfaces and handles three or more smooth pieces joined at a single curve.

The papers [4, 12, 14, 28] do not give any quantitative analysis of the reconstructed surfaces. Salman et al. [29] present the Hausdorff distance between their reconstruction and the ideal surface. The Hausdorff distance does not contain information about the match between surface normals. Daniels et al. [10] report quantitative results of a data compression algorithm [27] which was modified using the ideas of their paper. However, they do not report any quantitative results of their algorithm. Wang et al. [35] report the difference between normals estimated by their algorithm and the true surface normals. Their analysis is the closest we found to a measure of the quality of the feature reconstruction. However, even this analysis does not measuring the actual reconstructed surface and its features. It measures an intermediate quantity, the estimated surface normals, computed by the algorithm.

4 More Definitions

For each active grid edge $\mathbf{e}$, let $w_\mathbf{e} = (1-\alpha)v_a + \alpha v_b$ where $\alpha = (\sigma - s_a)/(s_a - s_b)$ and s_a and s_b are the scalar values at vertices v_a and v_b. Point $w_\mathbf{e}$ approximates the intersection of $\mathbf{e}$ and the isosurface.

For each active grid cube $\mathbf{c}$, let $\mathbf{c}$.centroid be the centroid of the points $w_\mathbf{e}$ over all the active edges of $\mathbf{c}$. If the isosurface is smooth around $\mathbf{c}$ and intersects $\mathbf{c}$ in a single connected component, then point $\mathbf{c}$.centroid will lie close to the isosurface.

A grid cube with grid indices (x_0, x_1, x_2) is in column x_0, row x_1 and z-plane x_2 of the grid. Let $\mathbf{c}$ and $\mathbf{c}'$ be grid cubes with grid indices (x_0, x_1, x_2) and (x_0', x_1', x_2'), respectively. The *cube distance* between the cubes is $\max(|x_0 - x_0'|, |x_1 - x_1'|, |x_2 - x_2'|)$. The *distance vector* between the cubes is $(|x_0 - x_0'|, |x_1 - x_1'|, |x_2 - x_2'|)$.

For a grid cube $\mathbf{c}$, subgrid $R_\mathbf{c}^{3\times3\times3}$ is the $3 \times 3 \times 3$ subgrid of 27 grid cubes centered at $\mathbf{c}$. Subgrid $R_\mathbf{c}^{5\times5\times5}$ is the $5 \times 5 \times 5$ subgrid of 125 grid cubes centered at $\mathbf{c}$.

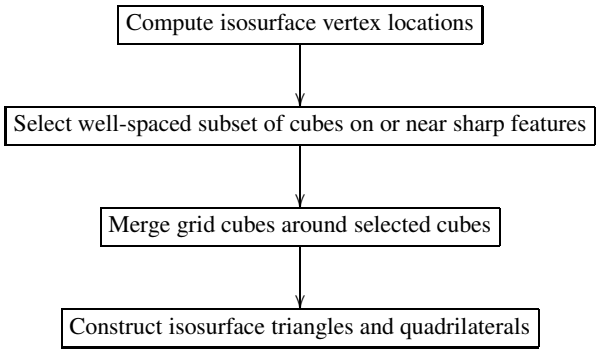


Figure 2: Algorithm SHREC.

```
/* C_d = cubes generating vertices on d-dimensional features */
1  Sort C_0 and C_1 by increasing |c.isov − c.centroid|;
2  Mark all cubes as uncovered;
3  foreach cube c of C_0 do
4      if (cube c is uncovered) then
5          if (c.isov does not create a large angle triangle with
           vertices from previously selected cubes) then
6              Select c;
7              foreach cube c' sharing a vertex with c do
8                  Mark c' as covered;
9              end
10         end
11     end
12 end
13 Repeat steps 3-12 on C_1;
```

Algorithm 1. Selection of feature cubes in MergeSharp.

5 MergeSharp

Algorithm SHREC has four parts: computation of isosurface vertex locations, selection of a well-spaced subset of grid cubes on or near sharp features, merging of grid cubes around selected cubes and construction of isosurface triangles and quadrilaterals (Figure 2). Algorithm MergeSharp from [1] has a similar four parts, but the first three parts are substantially different in SHREC. To motivate the description of Algorithm SHREC, we briefly describe the first three parts of MergeSharp and the associated problems.

In the first part, MergeSharp computes the location of an isosurface vertex, **c**.isov, in or near each active grid cube **c**. MergeSharp computes this location using gradients at the vertices of **c** and cubes adjacent to **c**. When cube **c** is near a sharp feature, the isosurface vertex will lie on that feature. MergeSharp also identifies whether the feature is 0-dimensional or 1-dimensional or if the vertex lies on a smooth region of the isosurface.

Let C_0 and C_1 be the set of grid cubes whose isosurface vertices lie on 0-dimensional features or 1-dimensional features, respectively. In the second part, MergeSharp selects a well-spaced subset of $C_0 \cup C_1$. Cubes which generate such vertices are called *selected* cubes. The vertices in selected cubes will form the sharp features in the isosurface mesh. MergeSharp chooses selected cubes as follows.

Mark all the cubes as "uncovered". Sort C_0 and C_1 by increasing distance of **c**.isov from **c**.centroid. Select the next uncovered cube **c** in C_0 where $p_{\mathbf{c}}$ does not form a large angle triangle with vertices in previously selected cubes. Mark all cubes which share a vertex with **c** as covered. After processing list C_0, apply the same procedure to list C_1. Pseudocode is given in Figure 1.

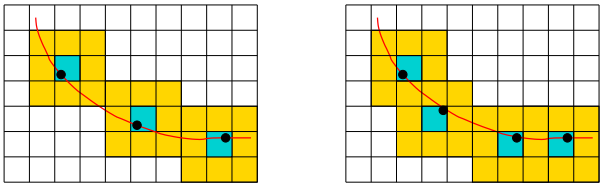


Figure 3: 2D illustration of vertex and cube selection. (a) Selection which gives poor cover of the red curve. The curve intersects an uncovered square. (b) Selection which gives better cover of the red curve. The curve is far away from any uncovered square.

The third part is merging covered cubes with selected cubes. MergeSharp simply merges each covered cube with the first selected cube which covers it. In the description of MergeSharp in [1], the selection and merging steps are combined so that a cube is merged as soon as it is covered, but the outcome of the algorithm is exactly the same.

There are problems with every one of the first three parts of MergeSharp. First, MergeSharp uses gradients from **c** or its immediate neighbors to compute the location of **c**.isov. As discussed in [7] computing correct gradients near sharp features is difficult, if not impossible. Thus, the gradients in **c** and its neighbors may not be known. Algorithm SHREC selects gradients from a larger area than just the immediate neighbors of **c**.

Second, the isosurface vertex **c**.isov generated by cube **c** may lie outside **c**. Moreover, if cube **c** is near a 1-dimensional feature, point **c**.isov may lie outside of **c** even if this 1-dimensional feature intersects **c**. In addition, because of curvature, noise and numerical instability, point **c**.isov could lie in some cube $\mathbf{c}'$ adjacent to **c** while point $\mathbf{c}'$.isov lies inside **c**.isov. Algorithm SHREC locates **c**.isov inside **c** whenever possible.

Third, the selection step chooses cubes based on the proximity of **c**.isov to **c**.centroid. If a point **c**.isov is near **c**.centroid, it probably is located in cube **c** and is a good approximation of the vertex location. While this is reasonable, it ignores the interaction between selected cubes. MergeSharp does better when 1-dimensional features are well-covered by the selected and covered cubes. For instance, the selected squares in Figure 3(b), are packed together more closely than the ones in Figure 3(a), and their 3×3 regions do a better job of covering the given curve. Algorithm SHREC selects cubes so that they are packed closely together.

Finally, the merging step merges cube **c** with the the first selected cube adjacent to **c**. Doing so sometimes distorts triangles, creating extremely thin triangles and sometimes creating "folds" in the surface mesh. Algorithm SHREC prefers merging cubes which are facet adjacent over merging edge adjacent or vertex adjacent cubes. It also avoids merges which creates small thin triangles or creates folds. It extends the merging by one more cube in certain regions to ensure good covering of 1D features.

6 Generating Points on Sharp Features

As noted in Section 5, when a cube **c** intersects a 1-dimensional feature, we would like to choose a point **c**.isov inside that feature whenever possible. We would also like that point to be "near" the center of the intersection of the isosurface and cube **c**. Finally, if the 1-dimensional feature is near **c** but does not intersect **c**, we would like to choose a point that is "closest" to **c**.

SHREC uses a number of techniques to compute points inside cubes. First, when a cube **c** is near a 1-dimensional feature, SHREC computes a line **c**.Lsharp approximating the feature near **c**. SHREC computes the points on **c**.Lsharp closest to **c**.centroid and

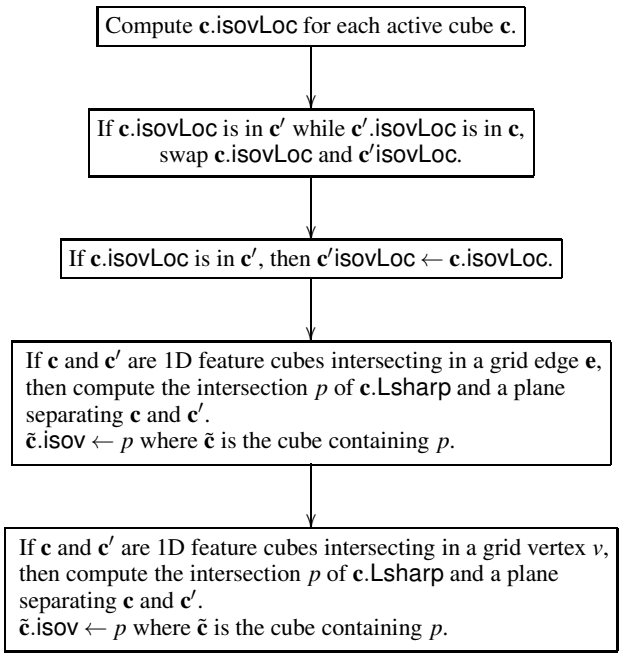


Figure 4: SHREC algorithm for generating points on sharp features.

c.center. If one of those points is inside **c**, then SHREC sets **c**.isov to that point. Otherwise, SHREC computes the point on **c**.Lsharp which is closest to **c**.center under the L_∞ metric. If **c**.Lsharp intersects **c**, then that point will lie in **c**.

Second, SHREC compares isosurface vertex locations computed by neighboring cubes and swaps or sets isosurface vertex locations based on those comparisons. Third, SHREC computes isosurface vertex locations on planes separating cubes intersecting 1D features. SHREC stores these isosurface vertex locations in the cubes containing them.

The second and third steps of the algorithm are outlined in Figure 4. Details of all three steps follow.

6.1 Computing Isosurface Vertex Locations

At the heart of any algorithm to compute a surface with sharp features is an algorithm to compute the locations of mesh vertices on those features. SHREC uses the algorithm from MergeSharp [1] to compute an initial vertex location **c**.isovLoc for each active cube **c**. The algorithm computes the vertex locations from gradients at grid vertices. The MergeSharp algorithm is a variation of the algorithm in [20] for computing vertex locations from surface normals. Similar algorithms are in [22, 31, 37].

SHREC computes one isosurface vertex location **c**.isovLoc for each active grid cube **c**. The algorithm computes an isosurface vertex location for **c** by using the gradients at vertices of **c** and nearby cubes. A grid vertex v with scalar s_v and gradient g_v gives a plane

$$h_v = \{p : (p-v)\cdot g_v + s_v = \sigma\}. \tag{1}$$

This plane is an "approximation" to the tangent planes to the isosurface at isosurface points in the neighborhood of v.

A set of k vertices and gradients gives a set of k equations in three variables $Mp = b$ where M is a $k\times 3$ matrix and b is a column vector with k rows and the unknown p is a column vector with three rows. Of course, if M has more than three rows, the system $Mp = b$ is over-determined and has no exact solution. The least squares approximation to $Mp = b$ is the solution to $M^T Mp = M^T b$.

Let A be the 3×3 matrix $M^T M$ and let b' equal the column vector $M^T b$. SHREC uses singular valued decomposition (SVD) as described in [20, 22, 23] to compute an isosurface vertex location from the equation $Ap = b'$.

Let σ_1, σ_2 and σ_3 be the singular values of A sorted in decreasing order. A singular value σ_i is *large*, if $\sigma_i/\sigma_1 \geq \varepsilon$ for some predefined parameter ε. There are three possible cases based on the number of large singular values of A. In the first case A has three large singular values. In this case, the tangent planes around **c** have normals in three or more very distinct directions. The isosurface has some 0-dimensional feature near cube **c**. We call a cube **c** a *0D feature cube* if the matrix A associated with **c** has three large singular values. The solution to $Ap = b'$ approximates the 0-dimensional feature near cube **c**.

In the second case, matrix A has only two large singular values. In this case, the tangent plane normals are close to two different directions. The isosurface has some 1-dimensional feature near cube **c**. We call a cube **c** a *1D feature cube* if the matrix A associated with **c** has two large singular values.

To compute the 1-dimensional feature near **c**, we use singular valued decomposition to remove the small singular value from A. The singular valued decomposition of A is $A = U\Sigma V^T$ where

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}.$$

When A has only two large singular values, σ_1 and σ_2, MergeSharp replaces Σ by a new diagonal matrix Σ' with diagonal entries σ_1, σ_2, 0. Let A' equal $U\Sigma' V^T$. Matrix A' has rank two. The solution to $A'p = b'$ is a set of points on a line **c**.Lsharp. Line **c**.Lsharp represents a line tangent to the 1-dimensional feature.

In the last case, matrix A has only one large singular value. In this case, the tangent plane normals are close to a single direction. The isosurface is smooth around cube **c**. Replace Σ by a new diagonal matrix Σ' with diagonal entries σ_1, 0, 0. Let A' equal $U\Sigma' V^T$. Matrix A' has rank one. The solution to $A'p = b'$ is a set of points on a plane. That plane represents a tangent plane to the isosurface in cube **c**.

In the case that A has only one or two large singular values, the solution to $A'p = b'$ is a line or plane. As suggested in [31], SHREC selects the point on the line or plane that is closest to **c**.centroid.

More precisely, SHREC defines the diagonal matrix Σ^+ with diagonal entries:

$$\begin{array}{ll} 1/\sigma_1, 1/\sigma_2, 1/\sigma_3, & \text{if } A \text{ has three singular values,} \\ 1/\sigma_1, 1/\sigma_2, 0, & \text{if } A \text{ has two singular values,} \\ 1/\sigma_1, 0, 0, & \text{if } A \text{ has one singular value.} \end{array}$$

The point

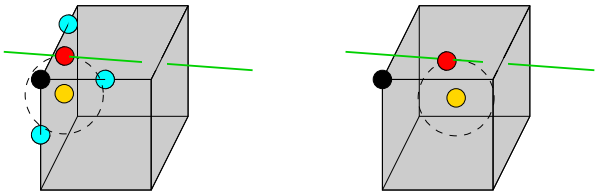
$$\phi_{\mathbf{c}}(q) = q + V\Sigma^+ U^T (b' - Aq) \tag{2}$$

is the point on $\{p : A'p = b'\}$ closest to q. SHREC selects point $\phi_{\mathbf{c}}(\mathbf{c}.\text{centroid})$ on the line or plane $\{p : A'p = b'\}$.

The number of large singular values of A determines whether the computed isosurface vertex location lies on a 0-dimensional feature, a 1-dimensional feature, or a smooth portion of the isosurface. This information is used in subsequent steps in the SHREC algorithm.

When $A'p = b'$ is a line, the direction of that line is given by the equation $(I - V\Sigma^+ U^T)w$ for any vector w which is not in the kernel of $(I - V\Sigma^+ U^T)w$. Substituting $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ for w and using $(I - V\Sigma^+ U^T)w$ with largest magnitude gives the direction. The direction and point $\phi_{\mathbf{c}}(\text{centroid})$ defines the line **c**.Lsharp.

Instead of using $A = M^T M$, we could have used the singular value decomposition of M. Using M is preferable for numerical



(a) (b)

Figure 5: Green line is **c**.Lsharp. Black cube vertex has scalar value above the isovalue. (a) Red point is point on **c**.Lsharp closest to **c**.centroid (yellow). Green line **c**.Lsharp intersects cube **c** but red point is outside cube **c**. Blue points are the intersection points of the isosurface and the cube edges, computed using linear interpolation. (b) Red point is point on **c**.Lsharp closest to **c**.center (yellow). Green line **c**.Lsharp intersects cube **c** but red point is outside cube **c**.

stability, since the condition number of A is the square of the condition number of M. However, because we remove small eigenvalues, the numerical stability is not an issue. We use the singular value decomposition of the 3×3 matrix A because it is simpler and faster to compute than the singular valued decomposition of the $k \times 3$ matrix M.

6.2 Computing Vertex Locations on 1-Dimensional Features

Consider a 1D feature cube **c**. By definition, cube **c** is near some 1-dimensional feature. That 1D feature is approximated by the line **c**.Lsharp. The location **c**.isovLoc of the isosurface vertex associated with **c** should be on **c**.Lsharp. However, that condition still gives one degree of freedom in selecting the location of **c**.isovLoc. One obvious additional condition is that if **c**.Lsharp intersects **c**, then **c**.isovLoc should be in **c**. An additional condition is that if **c**.Lsharp does not intersect **c**, then **c**.isovLoc should be "close to" **c** under some suitably defined metric.

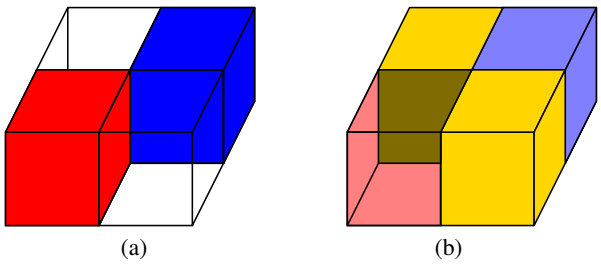
Algorithm SHREC computes three different possible isosurface vertex locations. First, Algorithm SHREC computes the point $p_{\mathbf{c}}^0 = \phi_{\mathbf{c}}(\mathbf{c}.\mathsf{centroid})$ (Equation 2), the point on **c**.Lsharp which is closest to **c**.centroid. The idea is that **c**.centroid is a good approximation for the intersection of **c** and the isosurface, and so one should choose a point near **c**.centroid. If $p_{\mathbf{c}}^0$ lies in **c**, then SHREC sets **c**.isovLoc to $p_{\mathbf{c}}^0$.

In most cases, if line **c**.Lsharp intersects cube **c**. then the point $p_{\mathbf{c}}^0$ will lie in **c**. However, if line **c**.Lsharp is "near" some facet or edge of **c**, then it is possible for **c**.Lsharp to intersect **c** but $p_{\mathbf{c}}^0$ to lie outside **c**. (See Figure 5(a).)

If $p_{\mathbf{c}}^0$ lies outside **c**, then SHREC computes $p_{\mathbf{c}}^1 = \phi_{\mathbf{c}}(\mathbf{c}.\mathsf{center})$, the point on **c**.Lsharp which lies closest to the center **c**.center of cube **c**. If $p_{\mathbf{c}}^1$ is in **c** while $p_{\mathbf{c}}^0$ is not, SHREC sets **c**.isovLoc to $p_{\mathbf{c}}^1$.

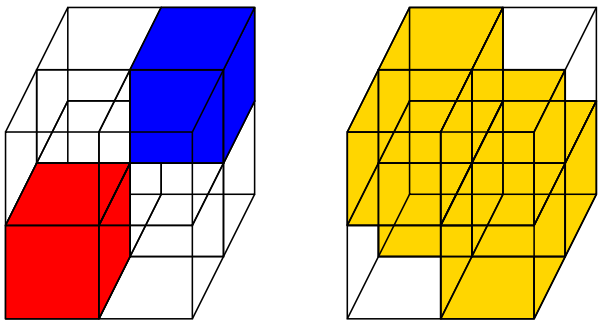
Unfortunately, it is possible that both $p_{\mathbf{c}}^0$ and $p_{\mathbf{c}}^1$ are not in **c** even though **c**.Lsharp intersects **c**. (See Figure 5(b).) As a final step, SHREC computes a point $p_{\mathbf{c}}^2$ on **c**.Lsharp which is closest to the center **c**.center of **c** under the L_∞ metric. If **c**.Lsharp intersects **c**, then this point is guaranteed to lie in **c**. Details for computing $p_{\mathbf{c}}^2$ are in Appendix A. If $p_{\mathbf{c}}^2$ is in **c** while $p_{\mathbf{c}}^1$ and $p_{\mathbf{c}}^2$ are not, SHREC sets **c**.isovLoc to $p_{\mathbf{c}}^2$.

Instead of computing $p_{\mathbf{c}}^0$, $p_{\mathbf{c}}^1$ and $p_{\mathbf{c}}^2$, we could compute and use only $p_{\mathbf{c}}^2$. However, the computations of $p_{\mathbf{c}}^0$ and $p_{\mathbf{c}}^1$ and $p_{\mathbf{c}}^2$ are much faster, and their locations are preferable to $p_{\mathbf{c}}^2$ when they are contained in cube **c**. Algorithm MergeSharp computes only $\phi_{\mathbf{c}}^0$.



(a) (b)

Figure 6: (a) Two grids cubes sharing an edge **e**. (b) The two other cubes (yellow) which also contain **e**.



(a) (b)

Figure 7: (a) Two grids cubes sharing a vertex v. (b) The six other cubes (yellow) which also contain v.

6.3 Swapping Locations

Many of the problems in Algorithm SHREC (and MergeSharp) occur when isosurface vertex location **c**.isovLoc lies outside of cube **c**. Thus, it is always preferable that **c**.isovLoc be in **c**. This is not always possible since the sharp features near **c** may not intersect **c**. However, in cases where a sharp feature intersects **c**, we would like **c**.isovLoc to be in **c**.

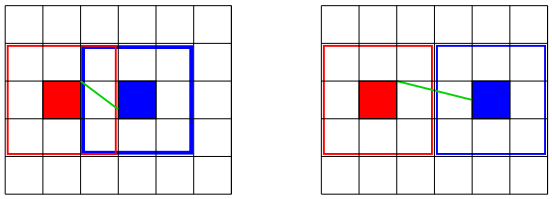
The computation of a sharp feature near **c** depends upon gradients in the neighborhood of **c**. The set of such gradients changes for each cube **c**. Because of the inaccuracy in computing sharp features, it is possible that **c**.isovLoc is in a cube $\mathbf{c}'$ adjacent to **c** while $\mathbf{c}'$.isovLoc is not contained in **c**. In some cases, location **c**.isovLoc is in $\mathbf{c}'$ while $\mathbf{c}'$.isovLoc is in **c**. In those cases, we simply swap **c**.isovLoc and $\mathbf{c}'$.isovLoc. In other cases, location $\mathbf{c}'$.isovLoc lies in some third cube $\mathbf{c}''$. In that case, we simply set $\mathbf{c}'$.isovLoc to **c**.isovLoc.

The setting of isosurface vertex locations from adjacent cubes increases the number of cubes **c** containing their associated vertex locations **c**.isovLoc. Note that if the initial location **c**.isovLoc lies in **c**, then we never change **c**.isovLoc.

6.4 Locations on Planes

Consider two grid cubes, **c** and $\mathbf{c}'$, which intersect in an edge **e** but not in any facet. (See Figure 6.) Two other grid cubes, $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$ share edge **e**. A 1-dimensional feature which passes through **c** and $\mathbf{c}'$ must intersect either $\tilde{\mathbf{c}}$ or $\tilde{\mathbf{c}}'$. (In the exceptional case, the 1-dimensional feature passes through **e**, in which case it intersects both $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$.) Since the 1D feature intersects either $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$, either $\tilde{\mathbf{c}}$.isovLoc should be in $\tilde{\mathbf{c}}$ or $\tilde{\mathbf{c}}'$.isovLoc should be in $\tilde{\mathbf{c}}'$. However, because of inaccuracies in computing points on 1D features, neither condition may hold.

To ensure that either $\tilde{\mathbf{c}}$.isovLoc lies in $\tilde{\mathbf{c}}$ or $\tilde{\mathbf{c}}'$.isovLoc lies in $\tilde{\mathbf{c}}'$, Algorithm SHREC computes the plane h containing **e** and perpendicular to the line from **c**.center to $\mathbf{c}'$.center. SHREC then



(a) Configuration (2,0) (b) Configuration (3,0)

(c) Configuration (2,1) (d) Configuration (3,1)

Figure 8: Tightly packed 2D configurations of selected squares. Selected squares and 3×3 subgrid around each square. Line segments with endpoints on the two selected squares (e.g., the green line segments) are contained within the union of the two 3×3 subgrids.

computes the intersection of the line $\mathbf{c}$.Lsharp and plane h. This intersections point, p_I, lies in either $\tilde{\mathbf{c}}$ or $\tilde{\mathbf{c}}'$. If p_I lies in $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}$.isovLoc is not in $\tilde{\mathbf{c}}$, SHREC sets $\tilde{\mathbf{c}}$.isovLoc to p_I. If p_I lies in $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}'$.isovLoc is not in $\tilde{\mathbf{c}}'$, SHREC sets $\tilde{\mathbf{c}}'$.isovLoc to p_I.

Next, consider the case of two grid cubes, $\mathbf{c}$ and $\mathbf{c}'$, which intersect in a vertex v but not in any edge. (See Figure 7.) Six other grid cubes share vertex v with $\mathbf{c}$ and $\mathbf{c}'$. A 1-dimensional feature which passes through $\mathbf{c}$ and $\mathbf{c}'$ must intersect one of these six other grid cubes. (In the exceptional case, the 1D feature passes through v, in which case all six.) Since the 1D feature intersects one of the six grid cubes, point $\tilde{\mathbf{c}}$.isovLoc should lie in $\tilde{\mathbf{c}}$ for one of the six grid cubes $\tilde{\mathbf{c}}$. Again because of inaccuracies in computing points on 1D features, point $\tilde{\mathbf{c}}$.isovLoc may not be in $\tilde{\mathbf{c}}$ for any of the six grid cubes $\tilde{\mathbf{c}}$.

To ensure that $p_{\tilde{\mathbf{c}}}$ lies in $\tilde{\mathbf{c}}$ for one of the six grid cubes, Algorithm SHREC computes the plane h containing v and perpendicular to the line from $\mathbf{c}$.center to $\mathbf{c}'$.center. SHREC then computes the intersection of the line $\mathbf{c}$.Lsharp and plane h. This intersections point, p_I, lies in at least one of the six grid cubes. If p_I lies in cube $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}$.isovLoc is not in $\tilde{\mathbf{c}}$, SHREC sets $\tilde{\mathbf{c}}$.isovLoc to p_I.

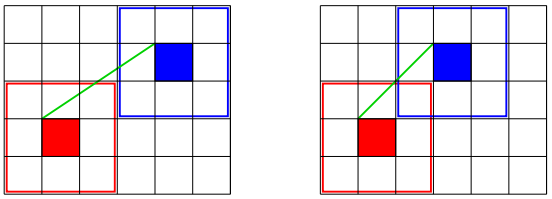
The computation in this section ensures "continuity" in the cubes which intersect a 1-dimensional feature If a 1-dimensional feature intersects a cube $\mathbf{c}$, then the 1-dimensional feature should intersect two cubes $\mathbf{c}'$ and $\mathbf{c}''$ which share a facet with $\mathbf{c}$. SHREC's computation of line-plane intersections described above, ensures that if $\mathbf{c}'$ and $\mathbf{c}''$ are active cubes, then $\mathbf{c}'$ will contain $\mathbf{c}'$.isovLoc and $\mathbf{c}''$ will contain $\mathbf{c}''$.isovLoc.

7 Selecting 0D and 1D Feature Cubes

Algorithm SHREC selects a well-spaced subset of the 0D and 1D feature cubes and merges isosurface vertices in adjacent cubes with the vertices generated by the selected cubes. To ensure that the isosurface vertices on sharp features are well-spaced, SHREC never selects two cubes which have a common vertex. Whenever SHREC selects a cube $\mathbf{c}$, any adjacent cube sharing a vertex with $\mathbf{c}$ is marked as "covered" by $\mathbf{c}$. SHREC never selects a covered cube.

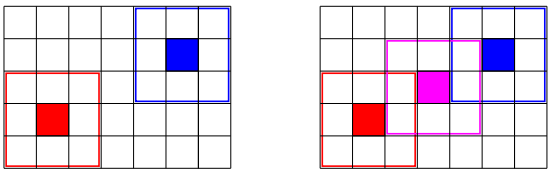
SHREC follows a MergeSharp rule in never selecting a cube $\mathbf{c}$ if $\mathbf{c}$.isov creates a large angle triangle (Algorithm 1, step 5) with vertices from previously selected cubes. SHREC does not select cube $\mathbf{c}$ if $\mathbf{c}$.isov creates such a triangle with angle 140° or greater.

As noted in Section 5, SHREC does better when selected cubes



(a) Configuration (3,2) (b) Configuration (2,2)

Figure 9: Problematic 2D configurations of selected squares. Selected squares and 3×3 subgrid around each square. (a) The green line segment is not contained within the union of the two 3×3 subgrids. (b) The green line segment intersects the boundary of the union of the two 3×3 subgrids.



(a) Configuration (4,2) (b) Additional cube

Figure 10: (a) Configuration (4,2) has space between the 3×3 subgrids around each selected square. (b) Additional cube (magenta) which packs tightly with two other squares.

are packed "tightly" together. (See also Figure 3.) In particular, SHREC tries to avoid certain configurations of nearby cubes.

Consider cubes $\mathbf{c}$ and $\mathbf{c}'$ with grid indices (x_0, x_1, x_2) and (x_0', x_1', x_2'), respectively. (A cube with grid indices (x_0, x_1, x_2) is in column x_0, row x_1 and z-plane x_2 of the grid.) The *distance vector* between the cubes is $(|x_0 - x_0'|, |x_1 - x_1'|, |x_2 - x_2'|)$. Two grid cubes are in configuration (a_0, a_1, a_2) if the distance vector between the cubes is some permutation of (a_0, a_1, a_2). Figures 8, 9, 10 and 11, contain examples of 2D and 3D configurations of grid cubes.

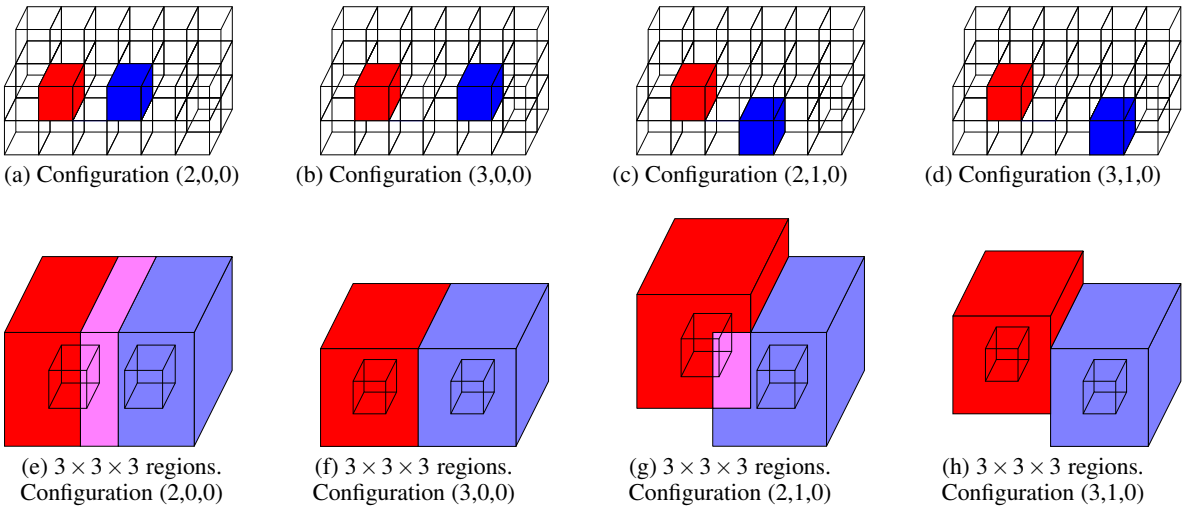
We can get some insight into 3D configurations of grid cubes by considering the 2D configurations of grid squares. Figure 8 contains configurations $(2,0)$, $(3,0)$, $(2,1)$ and $(3,1)$ of grid squares. The 3×3 subgrids around the selected squares are "tightly packed" so that any line segment with endpoints in the selected squares is contained within the union of the two 3×3 subgrids.

Figure 9 contains configurations $(3,2)$ and $(2,2)$ of grid squares. With configuration $(3,2)$, some line segments with endpoints in the selected squares are not contained within the union of the 3×3 subgrids. Configuration $(2,2)$ is somewhat better, since all line segments with endpoints in the selected squares are contained within the union of the 3×3 subgrids. Unfortunately, they are only barely contained. The green line segment in Figure 9(b) intersects the boundary of the 3×3 subgrid. Slightly curving the segment could mean that it no longer was contained in the union.

Note that if selected squares are suitably far apart, then additional squares can be selected between them. For instance, a magenta square can be selected between the two colored squares in Figure 10(a) and the 3×3 subgrids will fit together tightly.

Some examples of tightly packed 3D configurations of cubes are given in Figure 11. Any line segment with endpoints in the two selected cubes $\mathbf{c}$ and $\mathbf{c}'$ will be well contained within the union of the two $3 \times 3 \times 3$ subgrids, $R_{\mathbf{c}}^{3\times3\times3}$ and $R_{\mathbf{c}'}^{3\times3\times3}$, around $\mathbf{c}$ and $\mathbf{c}'$.

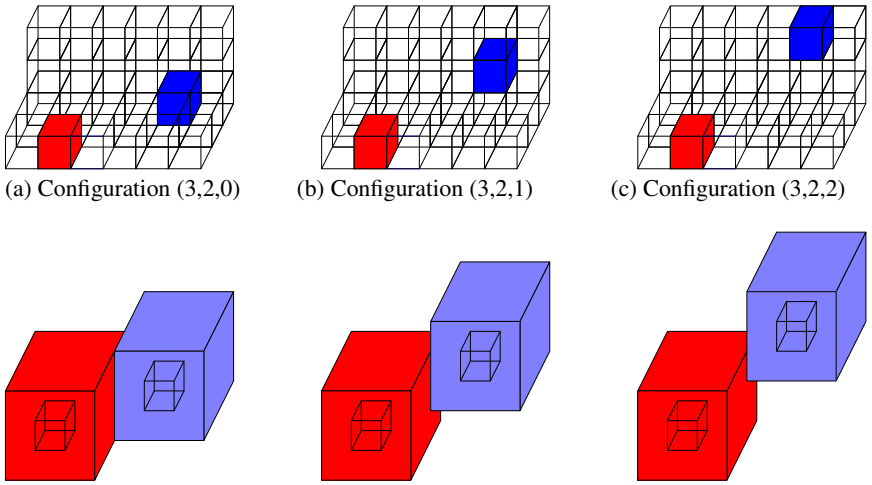
Figure 12 contains three problematic configurations, $(3,2,0)$, $(3,2,1)$ and $(3,2,2)$, of selected cubes. Line segments with endpoints in the two selected cubes $\mathbf{c}$ and $\mathbf{c}'$ may contain points outside $R_{\mathbf{c}}^{3\times3\times3} \cup R_{\mathbf{c}'}^{3\times3\times3}$. SHREC attempts to avoid selecting cubes with these configurations.



(a) Configuration (2,0,0) (b) Configuration (3,0,0) (c) Configuration (2,1,0) (d) Configuration (3,1,0)

(e) $3 \times 3 \times 3$ regions. Configuration (2,0,0) (f) $3 \times 3 \times 3$ regions. Configuration (3,0,0) (g) $3 \times 3 \times 3$ regions. Configuration (2,1,0) (h) $3 \times 3 \times 3$ regions. Configuration (3,1,0)

Figure 11: Tightly packed 3D configurations of selected cubes.



(a) Configuration (3,2,0) (b) Configuration (3,2,1) (c) Configuration (3,2,2)

(d) $3 \times 3 \times 3$ regions. Configuration (3,2,0) (e) $3 \times 3 \times 3$ regions. Configuration (3,2,1) (f) $3 \times 3 \times 3$ regions. Configuration (3,2,2)

Figure 12: Problematic 3D configurations of selected cubes.

If two selected cubes, $\mathbf{c}$ and $\mathbf{c}'$, are in configuration $(2,2,0)$, $(2,2,1)$ or $(2,2,2)$, then a line segment with endpoints in $\mathbf{c}$ and $\mathbf{c}'$ could intersect the boundary of $R_{\mathbf{c}}^{3\times3\times3} \cup R_{\mathbf{c}'}^{3\times3\times3}$. Such configurations are undesirable. Unfortunately, we found that avoiding such configurations is too restrictive.

Consider two cubes, $\mathbf{c}$ and $\mathbf{c}''$ in a $(3,2,0)$, $(3,2,1)$ or $(3,2,2)$ configuration. If a third selected cube $\mathbf{c}'$ lies between $\mathbf{c}$ and $\mathbf{c}''$, then the 1-dimensional feature will pass from $\mathbf{c}$ to $\mathbf{c}'$ to $\mathbf{c}''$. The selection of $\mathbf{c}'$ "blocks" the problematic interaction of $\mathbf{c}$ and $\mathbf{c}''$. If cubes $\mathbf{c}$ and $\mathbf{c}'$ are selected, then SHREC will permit the selection of $\mathbf{c}''$, even though $\mathbf{c}$ and $\mathbf{c}''$ have a configuration $(3,2,*)$. Figure 13 contains an example of two cubes in a $(3,2,2)$ configuration and a third cube between them.

More formally, let $\mathbf{c}$, $\mathbf{c}'$ and $\mathbf{c}''$ be grid cubes with grid indices (x_0,x_1,x_2), (x_0',x_1',x_2') and (x_0'',x_1'',x_2''), respectively. Cube $\mathbf{c}'$ *separates* $\mathbf{c}$ from $\mathbf{c}''$ if $x_i' \in [x_i,x_i'']$ for $i = 0,1,2$ and $x_i < x_i' < x_i''$ or $x_i > x_i' > x_i''$ for some i. SHREC avoids selecting cube $\mathbf{c}''$ if some already selected cube $\mathbf{c}$ forms a $(3,2,0)$ or $(3,2,1)$ or $(3,2,2)$ configuration with $\mathbf{c}''$ and no already selected cube separates $\mathbf{c}$ from $\mathbf{c}''$.

Figure 14 contains an outline of the algorithm for selecting 0D and 1D feature cubes. SHREC first selects 0D feature cubes. When the 0-dimensional feature is inside an active cube, SHREC selects the active cube containing the feature point. When the feature point is not in an active cube, SHREC selects the active cube "closest" to the feature point by choosing the cube $\mathbf{c}$ whose center has minimum L_∞ distance to the feature point.

SHREC next selects 1D feature cubes which are "near" the selected 0D feature cubes, i.e., they are contained in an $7 \times 7 \times 7$ subgrid around each selected 0D feature cube. Selecting 1D feature cubes near 0D feature cubes poses special challenges since there are multiple 1-dimensional features ending at a 0-dimensional feature. Selecting a cube which is near two such 1-dimensional features. can obscure one of the edges.

Let $\mathbf{c}$ be a selected 0D feature cube and let $\mathbf{c}'$ be a 1D feature cube which is in a $(2,0,0)$, $(2,1,0)$ or $(2,1,1)$ configuration with $\mathbf{c}$. Let $\mathbf{c}''$ be any cube sharing an edge or facet with $\mathbf{c}'$ which is contained in the $5 \times 5 \times 5$ subgrid but is not covered by $\mathbf{c}$. If $\mathbf{c}''$ is active and a 1D feature cube, then SHREC does not select cube $\mathbf{c}'$.

Once 1D feature cubes near 0D feature cubes are selected,

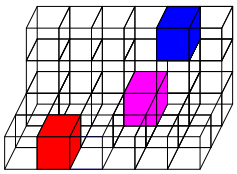


Figure 13: Problematic interaction of red and blue cubes in configuration (3,2,2) is blocked by magenta cube.

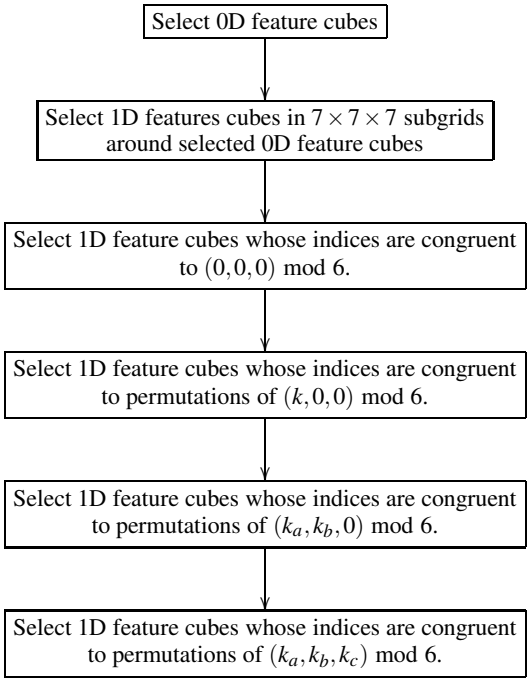


Figure 14: SHREC algorithm for selecting 0D and 1D feature cubes.

SHREC could iterate by selecting uncovered 1D feature cubes near already selected cubes. If no uncovered 1D feature cubes was near a selected cube, SHREC could select an arbitrary uncovered 1D feature cube and extend the set of selected cubes from that cube. By not selecting a cube $\mathbf{c}''$ if it forms a $(3,2,0)$, $(3,2,1)$ or $(3,2,2)$ with an already selected cube $\mathbf{c}$ (and is not separated by a selected cube from $\mathbf{c}$,) SHREC would produce a tight packing of selected cubes.

The algorithm outlined in the previous paragraph would produce a good set of selected cubes but it is highly sequential. One of the best aspects of the Marching Cubes and dual contouring algorithms is their local, distributed, parallelizable nature. Sequentially extending the set of selected cubes would totally destroy that aspect of the algorithms. Instead of selecting cubes using the sequential algorithm given above, SHREC divides the regular grid into overlapping $6\times6\times6$ subgrids, selects cubes from the boundaries of those subgrids and then from their interior. The algorithm is completely local and easily distributed and parallelizable.

Each grid cube has an index (x_0,x_1,x_2). SHREC processes the grid cubes by reducing the indices modulo six to $(x_0 \bmod 6, x_1 \bmod 6, x_2 \bmod 6)$ (Figure 15(a)). SHREC first selects 1D feature cubes with indices congruent to $(0,0,0)$ modulo six. SHREC next selects 1D feature cubes with indices congruent modulo six to $(\pm1,0,0)$ or $(0,\pm1,0)$ $(0,0,\pm1)$. (Figure 15(b)). SHREC then selects 1D feature cubes with indices congruent modulo six to $(\pm2,0,0)$ or $(0,\pm2,0)$ $(0,0,\pm2)$. Finally, SHREC selects 1D feature cubes

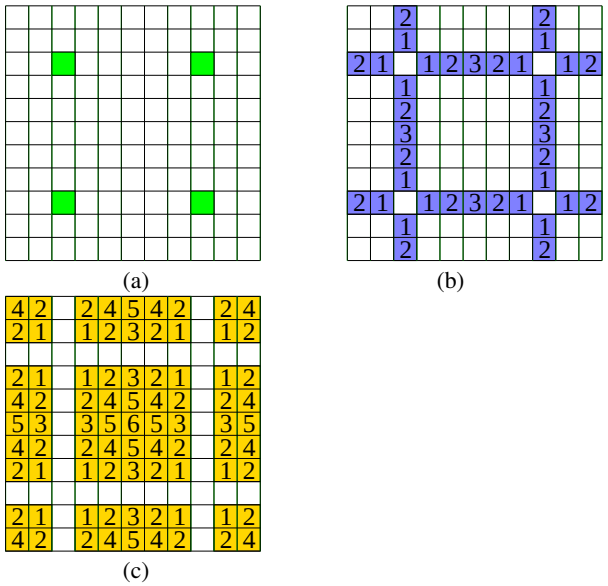


Figure 15: 2D illustration of order of cube selection based on cube indices mod 6. (a) Green squares are $(0,0)$ mod 6. (b) Blue squares are $(k,0)$ mod 6 or $(0,k)$ mod 6 for $k=1,\ldots,5$. Squares marked 1 are selected first, then squares marked 2 and then squares marked 3. (c) Yellow squares are (k_a,k_b) mod 6 for $k_a=1,\ldots,5$ and $k_b=1,\ldots,5$. Squares marked 1 are selected first, then squares marked 2, then 3, 4 and 5.

with indices congruent modulo six to $(3,0,0)$ or $(0,3,0)$ $(0,0,3)$. SHREC does not select a 1D feature cube $\mathbf{c}$ if some already selected cube $\mathbf{c}'$ forms configuration $(3,2,0)$, $(3,2,1)$ or $(3,2,2)$ with $\mathbf{c}$ and no already selected cube separates $\mathbf{c}$ from $\mathbf{c}''$.

SHREC next selects 1D feature cubes which are some permutation of $(k_a,k_b,0)$ modulo six, starting first with permutations of $(\pm1,\pm1,0)$ and ending with permutations of $(3,3,0)$. (Figure 15(c)). Finally, SHREC selects 1D feature cubes which are permutations (k_a,k_b,k_c) modulo six, starting first with permutations of $(\pm1,\pm1,\pm1)$ and ending with $(3,3,3)$.

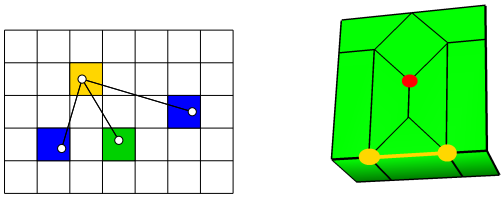
It is possible that the selection of two 1D feature cubes $\mathbf{c}$ and $\mathbf{c}''$ at distance five apart can force the selection of an edge cube $\mathbf{c}'$ which has a $(3,2,1)$ or $(3,2,2)$ configuration with either $\mathbf{c}$ or $\mathbf{c}''$. This happens if the distance vector between $\mathbf{c}$ and $\mathbf{c}''$ is $(5,3,2)$. Thus, in addition to avoiding $(3,2,*)$ configurations, SHREC also avoids selecting two cubes, $\mathbf{c}$ and $\mathbf{c}''$, which form a $(5,3,2)$ configuration. Of course, if $\mathbf{c}$ and $\mathbf{c}''$ are separated by some other selected cube, then they can both be selected.

While the above rules select a set of cubes which almost completely cover the 1-dimensional features of a surface, it is possible that the configuration restrictions leave a few areas uncovered. Therefore, SHREC repeats the selection process to select any remaining uncovered 1D feature cubes but drops the configuration restrictions.

8 Merging Points with Features Points

Algorithm MergeSharp merges neighbors of a selected cube $\mathbf{c}$ with $\mathbf{c}$ when $\mathbf{c}$ is selected. This merging step sometimes created extremely thin triangles and sometimes flips triangle orientations. SHREC is much more careful about its cube merging. SHREC also extends the merging to some cubes in a $5\times5\times5$ subgrid around each selected cube. Note that SHREC actually merges the isosurface vertices generated by the cubes, not the cubes themselves.

SHREC relies upon some angle tests to determine permissible



(a) (b)

Figure 16: (a) Isosurface in yellow square is connected to three selected cubes. Yellow square should merge with the middle green square, not with either of the blue squares. (b) Merging the red vertex with either of the yellow vertices will cause the yellow edge to be contained in four polygons.

merging. Before performing such tests, SHREC maps the grid and the computed isosurface locations $\mathbf{c}$.isovLoc to the regular grid composed of unit cubes.

8.1 Ordered Merging

SHREC first merges neighbors of selected 0D feature cubes. SHREC merges these neighbors in three steps. First, for each selected 0D feature cube $\mathbf{c}$, SHREC merges with $\mathbf{c}$ the active cubes which share a facet with $\mathbf{c}$. Next, for each selected 0D feature cube $\mathbf{c}$, SHREC merges with $\mathbf{c}$ the active cubes which share an edge with $\mathbf{c}$. Finally, for each selected 0D feature cube $\mathbf{c}$, SHREC merges with $\mathbf{c}$ the active cubes which share a vertex with $\mathbf{c}$. Of course, once an active cube is merged with some selected cube, it is never merged with any other cube.

SHREC next merges neighbors of selected 1D feature cubes. The procedure is similar to the one for 0D feature cubes. SHREC first merges active cubes which share a facet with a selected 1D feature cube, then merges active cubes which share an edge with a selected 1D feature cube, and finally merges active cubes which share a vertex with a selected 1D feature cube.

8.2 Merging Tests

For both the 0D feature cube and 1D feature cube merging, SHREC applies six different tests to avoid creating very thin triangles or flipping triangles or violating manifold conditions. Consider a cube $\tilde{\mathbf{c}}$ which SHREC would like to merge with a selected cube $\mathbf{c}$. In order for a cube $\tilde{\mathbf{c}}$ to merge with a selected cube $\mathbf{c}$, cubes $\tilde{\mathbf{c}}$ and $\mathbf{c}$ should satisfy the following conditions:

1. Some isosurface edge connects $\tilde{\mathbf{c}}$ to $\mathbf{c}$.
2. Merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ does not create an edge which is contained in four isosurface triangles or quadrilaterals (manifold condition).
3. If $\tilde{\mathbf{c}}$ is connected to three different selected 1D feature cubes, $\mathbf{c}$, $\mathbf{c}'$ and $\mathbf{c}''$, then $\mathbf{c}$ should lie "between" $\mathbf{c}'$ and $\mathbf{c}''$. (See Figure 16(a) for a 2D illustration.)
4. Merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ does not create a triangle with very small angles.
5. Merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ does not "flip" a triangle, creating a "fold" in the isosurface.
6. If cubes $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$ share an ambiguous facet, then $\tilde{\mathbf{c}}$ should merge with $\mathbf{c}$ if and only if $\tilde{\mathbf{c}}'$ merges with $\mathbf{c}$.

We describe these conditions in more detail below.

First, SHREC checks that the isosurface vertex in $\tilde{\mathbf{c}}$ is connected to the isosurface vertex in $\mathbf{c}$. A cube $\tilde{\mathbf{c}}$ is connected to a selected cube $\mathbf{c}$ if $\tilde{\mathbf{c}}$ shares an active facet or edge with $\mathbf{c}$ or some cube $\tilde{\mathbf{c}}'$ shares an active facet or edge with $\tilde{\mathbf{c}}$ and is merged with $\mathbf{c}$. If $\tilde{\mathbf{c}}$ is not connected with $\mathbf{c}$, then $\tilde{\mathbf{c}}$ is not merged with $\mathbf{c}$.

Second, SHREC checks for some manifold violations that could be caused by merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$. For each such selected cube $\mathbf{c}' \neq \mathbf{c}$ which is connected to $\tilde{\mathbf{c}}$, SHREC checks if $\mathbf{c}'$ is connected to $\mathbf{c}$. Let $\mathbf{c}'$ be a selected cube which is connected to $\mathbf{c}$ and $\tilde{\mathbf{c}}$. Each active edge $\mathbf{e}$ of $\tilde{\mathbf{c}}$ is dual to an isosurface quadrilateral with a vertex in $\tilde{\mathbf{c}}$. Let $\tilde{\mathbf{c}}_1$, $\tilde{\mathbf{c}}_2$, and $\tilde{\mathbf{c}}_3$ be the three other cubes containing $\mathbf{e}$. If some $\tilde{\mathbf{c}}_i$ merges with $\mathbf{c}$ and another $\tilde{\mathbf{c}}_j$ merges with $\mathbf{c}'$, then merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ collapses this quadrilateral to a triangle or an edge. However, if no isosurface quadrilateral dual to an active edge of $\tilde{\mathbf{c}}$ has vertices in $\tilde{\mathbf{c}}$, $\mathbf{c}$ and $\mathbf{c}'$, then merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ creates a non-manifold edge from $\mathbf{c}$ to $\mathbf{c}'$ with four incident polygons. (See Figure 16(b).) If some selected $\mathbf{c}'$ is connected to both $\mathbf{c}$ and $\tilde{\mathbf{c}}$, but no isosurface quadrilateral has vertices in $\tilde{\mathbf{c}}$, $\mathbf{c}$ and $\mathbf{c}'$, then SHREC does not merge $\tilde{\mathbf{c}}$ with $\mathbf{c}$.

Third, SHREC checks that if $\tilde{\mathbf{c}}$ is connected to three selected 1D feature cubes, $\mathbf{c}$, $\mathbf{c}'$ and $\mathbf{c}''$, then $\mathbf{c}$ is between $\mathbf{c}'$ and $\mathbf{c}''$. If not, then mapping $\tilde{\mathbf{c}}$ to $\mathbf{c}$ could create a triangle with a small angle. (See Figure 16 for a 2D illustration.) Let $\mathbf{c}$, $\mathbf{c}'$ and $\mathbf{c}''$ be grid cubes with grid indices (x_0, x_1, x_2), (x'_0, x'_1, x'_2) and (x''_0, x''_1, x''_2), respectively. If (x''_0, x''_1, x''_2) is contained in the box with corners (x_0, x_1, x_2) and (x'_0, x'_1, x'_2) and does not lie on any of the eight corners of that box, then we say that $\mathbf{c}''$ lies between $\mathbf{c}$ and $\mathbf{c}'$. If $\tilde{\mathbf{c}}$ is connected to selected 1D feature cubes $\mathbf{c}$, $\mathbf{c}'$ and $\mathbf{c}''$, and $\mathbf{c}''$ lies between $\mathbf{c}$ and $\mathbf{c}'$ or $\mathbf{c}'$ lies between $\mathbf{c}$ and $\mathbf{c}''$ then SHREC does not merge $\tilde{\mathbf{c}}$ with $\mathbf{c}$.

In the fourth and fifth tests, SHREC checks whether mapping $\tilde{\mathbf{c}}$ to $\mathbf{c}$ creates an isosurface triangle with a small angle or "flips" an isosurface triangle, creating a "fold" in the isosurface. These tests are a little more complicated than the others and are explained in the next section.

Finally, SHREC checks whether cube $\tilde{\mathbf{c}}$ has more than one isosurface vertex. If cube $\tilde{\mathbf{c}}$ has more than one isosurface vertex, then it has at least one ambiguous facet. (A facet is ambiguous if two diagonally opposite vertices have scalar value above the isovalue while the other two vertices have scalar value below the isovalue.) Let $\tilde{\mathbf{c}}'$ be a cube sharing an ambiguous facet with $\tilde{\mathbf{c}}$. If cube $\tilde{\mathbf{c}}$ has two isosurface vertices and $\tilde{\mathbf{c}}'$ is not merged with $\mathbf{c}$, then $\tilde{\mathbf{c}}$ is not merged with $\mathbf{c}$.

8.3 Distortion Tests

Let q be a quadrilateral dual to some active edge of $\tilde{\mathbf{c}}$. Let $\tilde{w}$ be the vertex of q which is generated by $\tilde{\mathbf{c}}$. Quadrilateral q can either be triangulated by adding a diagonal incident on $\tilde{w}$ or by adding a diagonal connecting the neighbors of $\tilde{w}$ in q. Triangulating $\tilde{w}$ by adding a diagonal incident on $\tilde{w}$ places more restrictions on possible locations of $\tilde{w}$. Since SHREC does not know how q will be triangulated, it assumes this more restrictive triangulation.

The diagonal of q incident on $\tilde{w}$ splits q into two triangles. SHREC checks whether mapping $\tilde{\mathbf{c}}$ to $\mathbf{c}$ will severely distort either of those triangles. Let $\tilde{\mathbf{c}}$, $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$ be the cubes containing the triangle vertices. SHREC only checks triangles where $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$ are not covered or selected.

Let $\tilde{w}$, $\tilde{w}'$ and $\tilde{w}''$ be the vertices generated by $\tilde{\mathbf{c}}$, $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$. Let w be the vertex generated by selected cube $\mathbf{c}$. In the fourth test, SHREC computes $\angle(w, \tilde{w}', \tilde{w}'')$ and $\angle(w, \tilde{w}'', \tilde{w}')$. If $\angle(w, \tilde{w}', \tilde{w}'')$ or $\angle(w, \tilde{w}'', \tilde{w}')$ is less than 5°, then SHREC does not map $\tilde{\mathbf{c}}$ to $\mathbf{c}$.

The fifth test is composed of two different parts. In the first part, SHREC checks whether mapping $\tilde{w}$ to w significantly changes the normal of triangle $(\tilde{w}, \tilde{w}', \tilde{w}'')$. If the angle between the normal of $(w, \tilde{w}', \tilde{w}'')$ is less than 30°, then the triangle passes this test.

In the second part, SHREC checks the orientation of $(w, \tilde{w}', \tilde{w}'')$. Let $\mathbf{e}$ be the grid edge shared by cubes $\tilde{\mathbf{c}}$, $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$. Let $\pi(w)$, $\pi(\tilde{w}')$ and $\pi(\tilde{w}'')$ be the orthogonal projection of w, $\tilde{w}'$ and $\tilde{w}''$, respectively, onto a plane h perpendicular to $\mathbf{e}$. If the orientation

of $\pi(w)$, $\pi(\tilde{w}')$ and $\pi(\tilde{w}'')$ matches the orientation of $\tilde{\mathbf{c}}$, $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$ around $\mathbf{e}$, then triangle $(w,\tilde{w}',\tilde{w}'')$ passes the orientation test.

To pass the fifth test, SHREC requires that a triangle pass either the normal or the orientation test, not necessarily both tests. The original triangle $(\tilde{w},\tilde{w}',\tilde{w}'')$ could have a normal which is very far from the true surface normal. In that case, the normal of triangle $(w,\tilde{w}',\tilde{w}'')$ should be far from the normal of triangle $(\tilde{w},\tilde{w}',\tilde{w}'')$. On the other hand, the projected vertices $\pi(w)$, $\pi(\tilde{w}')$ and $\pi(\tilde{w}'')$ could be nearly collinear. In that case, $\pi(w)$, $\pi(\tilde{w}')$ and $\pi(\tilde{w}'')$ could have opposite orientation from $\tilde{\mathbf{c}}$, $\tilde{\mathbf{c}}'$ and $\tilde{\mathbf{c}}''$, even though the normal of $(w,\tilde{w}',\tilde{w}'')$ is quite close to the original.

8.4 Pair Merging

A cube $\tilde{\mathbf{c}}$ may fail to merge with a selected cube $\mathbf{c}$ because of some neighboring cube $\tilde{\mathbf{c}}'$ while $\tilde{\mathbf{c}}'$ fails to merge with $\mathbf{c}$ because of $\tilde{\mathbf{c}}$. Often this "deadlocking" arises when $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$ share a common ambiguous facet.

After attempting to merge individual cubes, SHREC tries to merge pairs $(\tilde{\mathbf{c}},\tilde{\mathbf{c}}')$ of covered cubes with selected cubes. The elements of the pairs should share a common facet or edge and should both be covered by the selected cube $\mathbf{c}$. SHREC temporarily merges $\tilde{\mathbf{c}}$ with $\mathbf{c}$, and then applies all the above tests to $\tilde{\mathbf{c}}'$. SHREC then temporarily merges $\tilde{\mathbf{c}}'$ with $\mathbf{c}$, and applies all the above tests to $\tilde{\mathbf{c}}$. If both $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$ pass the tests, then SHREC merges both of them with $\mathbf{c}$.

As the merging proceeds, it is possible that a cube which was previously unable to merge with any selected cube is now able to merge with some such cube. Thus, SHREC reapplies the algorithm in Section 8.1 and in this section to any remaining covered, unmerged cubes.

8.5 Extended Merging

The merging of cubes in $3\times3\times3$ subgrid around each selected cube will clear most but not all of the vertices around sharp features. There are multiple reasons that some vertices near sharp features may remain. First, the location of some vertex outside of $R_{\mathbf{c}}^{3\times3\times3}$ may stop some cube covered by $\mathbf{c}$ from merging with $\mathbf{c}$. Second, if two selected $\mathbf{c}$ and $\mathbf{c}'$ are in a $(2,2,0)$ or $(2,2,1)$ or $(2,2,2)$ configuration, then a 1-dimensional feature with endpoints in $\mathbf{c}$ and $\mathbf{c}'$ can intersect the boundary of the $R_{\mathbf{c}}^{3\times3\times3}\cup R_{\mathbf{c}'}^{3\times3\times3}$. Vertices in uncovered cubes can be arbitrarily close to such a 1-dimensional feature. Third, while the selection step avoids most $(3,2,*)$ configurations, it does not avoid all of them. If $\mathbf{c}$ and $\mathbf{c}'$ are in a $(3,2,*)$ configurations, then a 1-dimensional feature passing through $\mathbf{c}$ and $\mathbf{c}'$ may contain points outside of $R_{\mathbf{c}}^{3\times3\times3}\cup R_{\mathbf{c}'}^{3\times3\times3}$.

To handle such problems, SHREC extends the merging to cubes in a $5\times5\times5$ subgrid around each selected cube. Merging cubes in a $5\times5\times5$ subgrid $R_{\mathbf{c}}^{5\times5\times5}$ around a selected cube $\mathbf{c}$ can create its own problems of thin or flipped triangles. Thus, SHREC only attempts to merge cubes in $R_{\mathbf{c}}^{5\times5\times5}$ which are potentially near a 1-dimensional feature through $\mathbf{c}$.

First, SHREC checks whether any covered cubes are unmerged after the steps in Sections 8.1 and 8.4. If some cube $\tilde{\mathbf{c}}$ is covered by selected cube $\mathbf{c}$ but not merged with any cube, SHREC pairs $\tilde{\mathbf{c}}$ with any adjacent active cubes $\tilde{\mathbf{c}}'$ sharing a vertex with $\tilde{\mathbf{c}}$ and attempts to merge the pair $(\tilde{\mathbf{c}},\tilde{\mathbf{c}}')$ with $\mathbf{c}$. SHREC applies the pair merging procedure in Section 8.4 to determine whether to allow the pair $(\tilde{\mathbf{c}},\tilde{\mathbf{c}}')$ to merge with $\mathbf{c}$. Note that in Section 8.4 both cubes in the pair must be contained in $R_{\mathbf{c}}^{3\times3\times3}$ while here only one cube must be contained in $R_{\mathbf{c}}^{3\times3\times3}$.

It is possible that the interaction of vertices in three cubes prevents a covered cube from merging with a selected cube. SHREC considers triples of unmerged cubes which share a common edge $\mathbf{e}$. At least one cube in the triple must be in $R_{\mathbf{c}}^{3\times3\times3}$. The triple merging procedure is the same as the pair merging procedure described in Section 8.4. Its description is omitted.

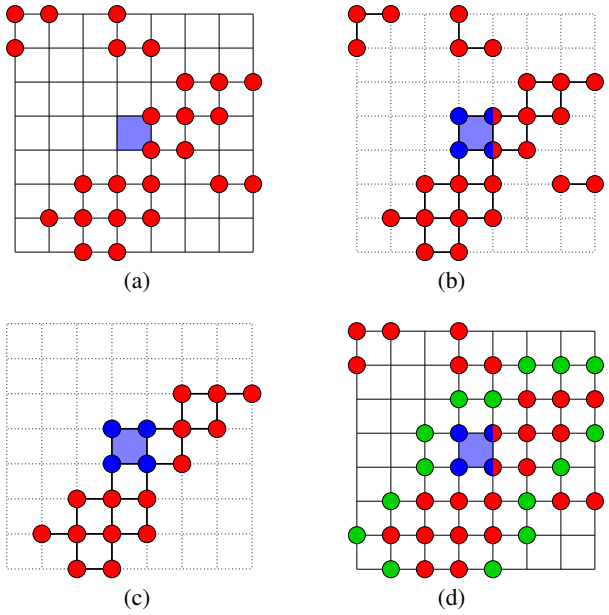


Figure 17: 2D example of selecting gradients "close" to square $\mathbf{c}$. (a) Set Q of red vertices with undefined gradients. (b) Graph $G_{\mathbf{c}}$ formed from $Q\cup Q_{\mathbf{c}}$ and the grid edges. $Q_{\mathbf{c}}$ is the set of four blue vertices of square $\mathbf{c}$. (c) Connected component G' of $G_{\mathbf{c}}$. (d) Green vertices which are connected to the vertices of G'. The green vertices are the vertices with defined gradients which are "close" to $\mathbf{c}$.

Next, SHREC tries to merge cubes which lie at the intersection $\partial R_{\mathbf{c}}^{3\times3\times3}\cap\partial R_{\mathbf{c}'}^{3\times3\times3}$ of two $3\times3\times3$ subgrids around two selected cubes $\mathbf{c}$ and $\mathbf{c}'$. More specifically, SHREC attempts to merge a cube $\tilde{\mathbf{c}}$ with a selected cube $\mathbf{c}$ if some facet of $\tilde{\mathbf{c}}$ lies on the boundary of $R_{\mathbf{c}}^{3\times3\times3}$ while some other facet of $\tilde{\mathbf{c}}$ lies on the boundary of $R_{\mathbf{c}'}^{3\times3\times3}$. Note that such a cube is contained in $R_{\mathbf{c}}^{5\times5\times5}$. SHREC applies all the tests in Section 8.2 to determine whether to merge $\tilde{\mathbf{c}}$ with $\mathbf{c}'$.

Finally, SHREC tries to merge pairs of cubes $(\tilde{\mathbf{c}},\tilde{\mathbf{c}}')$ which lie at the intersection $\partial R_{\mathbf{c}}^{3\times3\times3}\cap\partial R_{\mathbf{c}'}^{3\times3\times3}$. Both $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}'$ must have facets which lie on $R_{\mathbf{c}}^{3\times3\times3}$ and $R_{\mathbf{c}'}^{3\times3\times3}$. SHREC applies the pair merging procedure described in Section 8.4 to determine whether to allow the pair $(\tilde{\mathbf{c}},\tilde{\mathbf{c}}')$ to merge with $\mathbf{c}$.

9 Selecting Gradients

The algorithm in Section 6.1 for computing isosurface vertex locations requires a set of gradients for each cube $\mathbf{c}$. An obvious set is the gradients at the vertices of $\mathbf{c}$. However, even if correct gradients were provided at every vertex of $\mathbf{c}$, this set would not suffice. It is possible to have an active cube $\mathbf{c}$ which contains a sharp isosurface corner defined by three perpendicular planes while only two of the planes are represented by the gradients of $\mathbf{c}$.

MergeSharp computes isosurface vertex locations from the vertices of $\mathbf{c}$ and of the six cubes sharing a facet with $\mathbf{c}$. When exact gradients are provided at all grid vertices, this set of gradients suffices.

When gradients are not provided in the input, they must be computed from the scalar data. If an isosurface has sharp features, then the gradient field is discontinuous around those sharp features. As discussed in [7] computing correct gradients near gradient field discontinuities is extremely difficult. In [7] Bhattacharya and Wenger give an algorithm for computing correct gradients in the presence of gradient discontinuities, but the algorithm does not compute correct

gradients at all the grid vertices. Instead, the algorithm returns a set of correct gradients at some of the grid vertices while returning no gradient information at other vertices. Under suitable conditions, the algorithm returns the correct gradient at any vertex which is at least three edge lengths from any gradient discontinuity.

Because the algorithm in [7] does not return gradients near gradient discontinuities, no gradient information may be available for the vertices of cubes which contain those discontinuities or for neighbors of such cubes. Thus, we must use gradients from a $7 \times 7 \times 7$ subgrid around each cube $\mathbf{c}$.

When scalar data is provided by computer tomography (CT), then the scalar values near gradient discontinuities may also be incorrect. As discussed in [7], we must go out even further to a $9 \times 9 \times 9$ subgrid around each cube $\mathbf{c}$ to get gradient information.

Only gradients which determine isosurface tangent planes near $\mathbf{c}$ should be used in $\mathbf{c}$.isovLoc. We use three tests on the vertices in a $7 \times 7 \times 7$ or $9 \times 9 \times 9$ subgrid around $\mathbf{c}$ to determine such gradients. First, we are only interested in vertices which are near the isosurface. Thus, we only choose vertices from edges where one endpoint has scalar value below the isovalue and one endpoint has scalar value at or above the isovalue. Second, we are only interested in vertices whose gradients generate planes which are close to $\mathbf{c}$. We construct a cube $\mathbf{c}'$ of size $2 \times 2 \times 2$ centered at $\mathbf{c}$ and only choose a vertex v if the plane h_v given by Equation 1 intersects $\mathbf{c}'$.

Third, if vertices "close" to $\mathbf{c}$ have been chosen, then there is no reason to select vertices "far" from $\mathbf{c}$. In fact, choosing gradients at vertices "far" from $\mathbf{c}$ in smooth, curved surfaces can cause the creation of non-existent sharp features in the isosurface.

To choose only vertices close to $\mathbf{c}$, we consider the set Q of vertices of the $9 \times 9 \times 9$ (or $7 \times 7 \times 7$) subgrid, whose gradients are undefined. (See Figure 17.) We choose only vertices with defined gradients which are adjacent to vertices in Q.

More precisely, let Q be the set of vertices of the subgrid whose gradients are undefined (Figure 17(a)). Let $Q_{\mathbf{c}}$ be the vertices of $\mathbf{c}$. Let $G_{\mathbf{c}}$ be the graph whose vertices are $Q \cup Q_{\mathbf{c}}$ and whose edges are (u,v) where (u,v) is a grid edge (Figure 17(b)). We find the connected component G' of $G_{\mathbf{c}}$ containing $Q_{\mathbf{c}}$ (Figure 17(c)). A grid vertex $u \notin V(G')$ is on the boundary of G' if (u,v) is a grid edge and v is in $V(G')$. As out third test, we only a choose a vertex if it is in $Q_{\mathbf{c}}$ or if it is on the boundary of G'. (Figure 17(d)).

Applying the three tests gives a set of vertices and their gradients which define planes. We use those vertices and their gradients to compute $p_{\mathbf{c}}$ as described in Section 6.1.

10 Constructing isosurface triangles and quadrilaterals

The last step in SHREC is the construction of isosurface triangles and quadrilaterals. SHREC first applies the Manifold Dual Marching Cubes algorithm described in [36] to the full resolution grid (no merged grid cubes) to construct a set of isosurface quadrilaterals. The algorithm is essentially the same as the Nielson's Dual Marching Cubes algorithm [26] but with some small changes to the isosurface in certain ambiguous cases. We briefly describe Manifold Dual Marching Cubes.

Each grid vertex with scalar value below the isovalue receives a negative ("$-$") label. Each grid vertex with scalar value equal to or above the isovalue receives a positive ("$+$") label. A grid edge is active if it has one positive endpoint and one negative endpoint.

The eight labels on the eight vertices of a cube $\mathbf{c}$ determine the configuration of the cube. There are $2^8 = 256$ possible configurations. A lookup table stores the number of isosurface vertices for each configuration κ. Cubes with configuration κ contain the corresponding number of isosurface vertices. For each configuration κ, the lookup table also stores an assignment of active edges to isosurface vertices. If a cube has configuration κ, then the isosurface quadrilateral dual to active edge $\mathbf{e}$ is incident on the isosurface vertex assigned to edge $\mathbf{e}$.

As with all dual contouring algorithms, Manifold Dual Marching Cubes constructs one isosurface quadrilateral dual to each active grid edge. The four vertices of the isosurface quadrilateral q lie in the four grid cubes containing the active edge $\mathbf{e}$. Using the lookup table, we determine the four isosurface vertices in the four grid cubes which form the vertices of q and store q by its four vertices.

When two cubes, $\mathbf{c}$ and $\mathbf{c}'$, share an ambiguous facet and each has only one isosurface vertex, the procedure in the previous paragraph will generate a non-manifold edge contained in four quadrilaterals. In this case, we replace the configurations of $\mathbf{c}$ and $\mathbf{c}'$ with complementary configurations where all the positive and negative vertex labels are flipped. The complementary configurations generate two isosurface vertices in each cube, avoiding the non-manifold edge. Because the initial configuration for each cube generates only one isosurface vertex, each cube has only one ambiguous facet, the facet shared by $\mathbf{c}$ and $\mathbf{c}'$. Thus, using the complementary configurations for $\mathbf{c}$ and $\mathbf{c}'$ does not create any "cracks" in the connection of $\mathbf{c}$ and $\mathbf{c}'$ to other cubes.

After constructing a full resolution isosurface mesh, SHREC collapses mesh quadrilaterals using the previously constructed grid cube merging. For each selected cube $\mathbf{c}$, let $w_{\mathbf{c}}$ be the isosurface vertex generated by $\mathbf{c}$. If $\mathbf{c}$ generates more than one isosurface vertex, arbitrarily choose one to be $w_{\mathbf{c}}$.

For each isosurface vertex of a non-selected cube $\mathbf{c}'$, if $\mathbf{c}'$ merges a selected cube $\mathbf{c}$ let $M(w)$ equal $w_{\mathbf{c}}$. For each isosurface vertex w of a selected cube $\mathbf{c}$, let $M(w)$ equal $w_{\mathbf{c}}$. For every other isosurface vertex w, let $M(w)$ equal w.

Replace each quadrilateral (w,w',w'',w''') by $(M(w),M(w'),M(w''),M(w'''))$. Replacing the vertices of these quadrilaterals will map some of the quadrilaterals to triangles and collapse others to edges, vertices or pairs of edges. Remove the quadrilaterals which are mapped to vertices or edges.

For each selected cube $\mathbf{c}$, assign $w_{\mathbf{c}}$ the isosurface vertex location computed for $\mathbf{c}$ as described in Section 6. For each unselected, unmerged cube containing a single isosurface vertex w, also assign w the location computed for $\mathbf{c}$ as described in Section 6. The remaining isosurface vertices are in unselected, unmerged cubes which contain multiple isosurface vertices. For each such isosurface vertex w, let E_w be the cube edges dual to the quadrilaterals incident on w. Using linear interpolation, compute the intersection point of the isosurface and each edge $\mathbf{e} \in E_w$. Locate w at the centroid of the intersection points.

Manifold Dual Marching Cubes generates an isosurface which is always a manifold. However, the merging of isosurface vertices can create non-manifold edges. In the experiments described in Section 13, SHREC almost always produced a manifold isosurface. If a manifold isosurface is required, then tests such as in [11] or [30] can be added to the merging step to ensure that the resulting surface is a manifold.

10.1 Grid Spacing

The distortion tests for cube merging (Section 8.3) depend upon two angle parameters, one for testing triangle angles and one for testing changes in normals. If the grid is not spaced by the same length in the x, y and z directions, these tests will be biased in certain directions. To avoid this bias, we map the isosurface vertex locations to locations in a grid with uniform $1 \times 1 \times 1$ spacing before applying these tests.

In gradient selection (Section 9), we use a $2 \times 2 \times 2$ region around cube $\mathbf{c}$ to determine the selected gradients. Of course, this region size is only for a grid with uniform $1 \times 1 \times 1$ spacing. If the grid spacing is $\delta_x \times \delta_y \times \delta_z$, we use a region of size $(2/\delta_x, 2/\delta_y, 2/\delta_z)$ around cube $\mathbf{c}$.

In sharp cube selection (Section 7), we use an angle test to avoid selecting a cube whose isosurface vertex may create a thin triangle with vertices in other selected cubes. For that test, we use the vertex locations in the original grid.

11 Parameters

Algorithm SHREC has two major parameters, one determining the number of large singular values in matrix A and the second determining the size of the $k \times k \times k$ subgrid from which gradients are selected around each cube (Section 9). The first parameter is a value ε between 0 and 1. As described in Section 6.1, a singular value σ_i of matrix A is called "large" if σ_i/σ_1 is greater than or equal to ε. The number of large singular values determines whether a vertex is on a sharp feature or a smooth subgrid of the isosurface. To the best of our knowledge, all algorithms which reconstruct surfaces with sharp features require some parameter to distinguish the sharp features from the smooth regions of the surface.

The second parameter is an odd integer $k \geq 3$. Each cube $\mathbf{c}$ uses gradients from a $k \times k \times k$ subgrid around the cube in selecting gradients for computing $\mathbf{c}$.isovLoc. The size of k depends upon the input data. If correct gradients are provided at each grid vertex, then k should equal 3 for a $3 \times 3 \times 3$ subgrid around each cube. If gradients are computed from correct scalar values using the algorithm in [7], then k should equal 7 for a $7 \times 7 \times 7$ subgrid around each cube. If gradients are computed from CT data using the algorithm in [7], then k should equal 9 for a $9 \times 9 \times 9$ subgrid around each cube.

SHREC uses three other constants: one for the triangle test in selecting vertices, one for the angle test in merging cubes and a second for the normal test in merging cubes. In the triangle test, if selecting cube $\mathbf{c}$ would possibly create a triangle between three selected cubes with angle greater than 140°, cube $\mathbf{c}$ is not selected. In the angle test, if merging cube $\tilde{\mathbf{c}}$ with $\mathbf{c}$ would create a triangle with angle less than 5°, then cube $\tilde{\mathbf{c}}$ is not selected. In the normal test, if merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ would change the normal of some triangle $(\tilde{w}, \tilde{w}', \tilde{w}'')$ by less than 30°, then merging $\tilde{\mathbf{c}}$ with $\mathbf{c}$ does not distort triangle $(\tilde{w}, \tilde{w}', \tilde{w}'')$.

Because SHREC is based on the regular grid, the constants used in all three of these tests do not depend upon the input data and should work well for any scalar fields. Note that SHREC maps the input grid and the computed isosurface locations $\mathbf{c}$.isovLoc to the regular grid composed of unit cubes before applying the angle or normal tests.

12 Measuring Angle Distance between Surfaces

As described in Section 3, we evaluated MergeSharp in [1] by extracting sharp mesh edges (dihedral angle less than 140°) and comparing the 1-skeleton formed by those sharp edges with the 1-skeleton of the sharp edges in an ideal surface. We present here a different way of evaluating a reconstructed mesh containing sharp features.

Let Σ_P and Σ_Q be two surfaces. Given a point $p \in \Sigma_P$, the distance, $d(p, \Sigma_Q)$, from p to Σ_Q is the distance from p to the closest point on Σ_Q, i.e., $d(p, \Sigma_Q) = min_{q \in \Sigma_Q} d(p, q)$. We would like a similar measurement of the difference between the normal at p and the normals of Σ_Q.

The simplest approach would be to locate the point $q \in \Sigma_Q$ closest to p and measure the difference between their normals. As shown in Figure 18, this measurement is not be very useful. Rectangle A' is a slightly translation of rectangle A. The boundaries of rectangles A and A' are close under the Hausdorff metric and their normals are the same. However, point p is in the intersection of the two boundaries, ∂A and $\partial A'$, but the normal to point p in ∂A is 90° from the normal to point p in $\partial A'$. Furthermore, for any point $\tilde{p} \in A$ in a suitably small neighborhood of p, the closest point in $\tilde{p}' \in A'$ has normal which is 90° from the normal of A at $\tilde{p}$. Note that the translation could be arbitrarily small and there would still be a point in A whose normal was 90° from the closest point in A'.

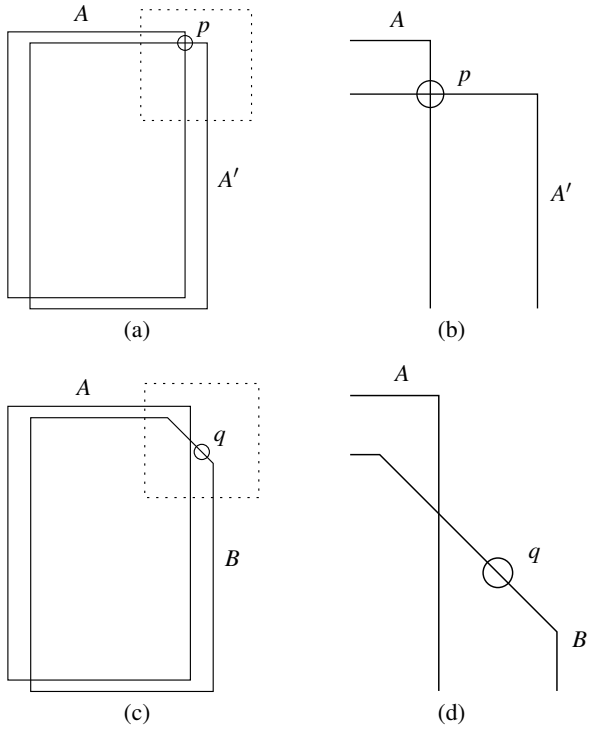


Figure 18: (a) Rectangle A' is a slight perturbation of rectangle A. (b) Close up view of dotted region around point p. (c) The corner of polygon B is clipped near q. (d) Close up view of dotted region around point q.

In contrast to the match between the normals of A and A', the corner of polygon B in Figure 18(c) is clipped and its normal is 45° from any normal of A. We would like some measurement which gives a high value between $q \in B$ and A while giving a low value between any point of A' and A. Our idea is to compare the normal at $p \in \Sigma_P$ with the normals of Σ_Q in some suitable neighborhood around p.

Let Σ_P^S and Σ_Q^S be the set of smooth points in Σ_P and Σ_Q. Let n_p and n_q be the normals of $p \in \Sigma_P^S$ and $q \in \Sigma_Q^S$, respectively. Let $B_p(\varepsilon)$ be the ball around p of radius ε. For each point $p \in \Sigma_a^S$ where $B_p(\varepsilon) \cap \Sigma_Q^S \neq \emptyset$, define the angle distance between p and Σ_Q in the ε-neighborhood as:

$$\tilde{d}_A^\varepsilon(p, \Sigma_Q) = \liminf\{\angle(n_p, n_q) : q \in \Sigma_Q^S \cap B_p(\varepsilon)\}.$$

Define the directed angle distance between Σ_P and Σ_Q in ε-neighborhood as:

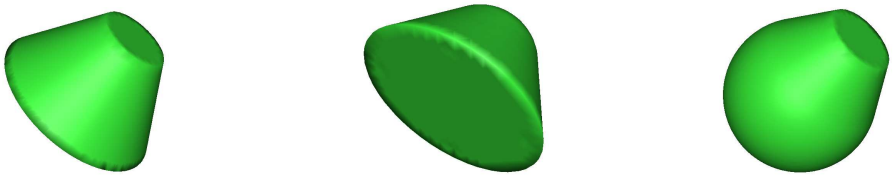
$$\tilde{d}_A^\varepsilon(\Sigma_P, \Sigma_Q) = \limsup_{p \in \Sigma_P^S \text{ and } B_p(\varepsilon) \cap \Sigma_Q^S \neq \emptyset} \tilde{d}_A^\varepsilon(p, \Sigma_Q).$$

Finally, define the angle distance between Σ_P and Σ_Q in ε-neighborhood as:

$$d_A^\varepsilon(\Sigma_P, \Sigma_Q) = \max(\tilde{d}_A^\varepsilon(\Sigma_P, \Sigma_Q), \tilde{d}_A^\varepsilon(\Sigma_Q, \Sigma_P)).$$

The angle distance depends upon the parameter ε. If some point $p \in \Sigma_P^S$ is not within ε of Σ_Q, then $\tilde{d}_A^\varepsilon(p, \Sigma_Q)$ is undefined. To address this problem, we replace the neighborhood $B_p(\varepsilon)$ by an

(a) TwoCubes (b) Annulus (c) Flange



(d) Frustrum (e) Smooth Tip Cone (f) Cannon

Figure 19: Isosurfaces constructed from synthetic test datasets.

extended neighborhood $B_p(\varepsilon + d(p,\Sigma_q))$ where $d(p,\Sigma_q)$ is the distance from p to Σ_Q. The ball around p is now guaranteed to intersect Σ_Q.

Define the angle distance between $p \in \Sigma_p^S$ and Σ_Q in the extended ε-neighborhood as:

$$\tilde{d}_A^{+\varepsilon}(p,\Sigma_Q) = \liminf\{\angle(n_p,n_q) : q \in \Sigma_Q^S \cap B_p(\varepsilon + d(p,\Sigma_Q))\}.$$

As before, $d(p,\Sigma_Q)$ is the distance from p to the closest point on Q. Note that the addition of $d(p,\Sigma_Q)$ ensures that the ball around p intersects Σ_Q.

Define the directed angle distance between Σ_P and Σ_Q in extended ε-neighborhood as:

$$\tilde{d}_A^{+\varepsilon}(\Sigma_P,\Sigma_Q) = \limsup_{p \in P_S} \tilde{d}_A^{+\varepsilon}(p,\Sigma_Q).$$

Define the angle distance between Σ_P and Σ_Q in extended ε-neighborhood as:

$$d_A^{+\varepsilon}(\Sigma_P,\Sigma_Q) = \max(\tilde{d}_A^{+\varepsilon}(\Sigma_P,\Sigma_Q), \tilde{d}_A^{+\varepsilon}(\Sigma_Q,\Sigma_P)).$$

Unfortunately, neither $d_A^{\varepsilon}(\Sigma_p,\Sigma_q)$ nor $d_A^{+\varepsilon}(\Sigma_p,\Sigma_q)$ obey the triangle inequality. Nevertheless, we think that the angle distance is a useful way of measuring the difference in surface normals between two surfaces.

The angle distance depends upon a consistent orientation of the two surfaces. If the surface is not oriented or if it is represented by a set of polygons without a consistent orientation, then we can still measure an "unoriented" angle distance by using $\min(\angle(n_p,n_q),\angle(n_p,-n_q))$ in place of $\angle(n_p,n_q)$. We call the two versions of the angle distance, the oriented angle distance and the unoriented angle distance.

The angle distance, both in the absolute and extended neighborhoods, can be modified to give a variety of measurements of surface normal differences. Instead of measuring the maximum angle difference between normals, one could count the number of polygons with normal difference above a threshold or the total area of such polygons. One could also produce a histogram of the number or total area of such polygons with normal difference in given ranges. Some such histograms are provided in Section 13.

13 Experimental Results on Synthetic Data

13.1 Synthetic Scalar and Gradient Data

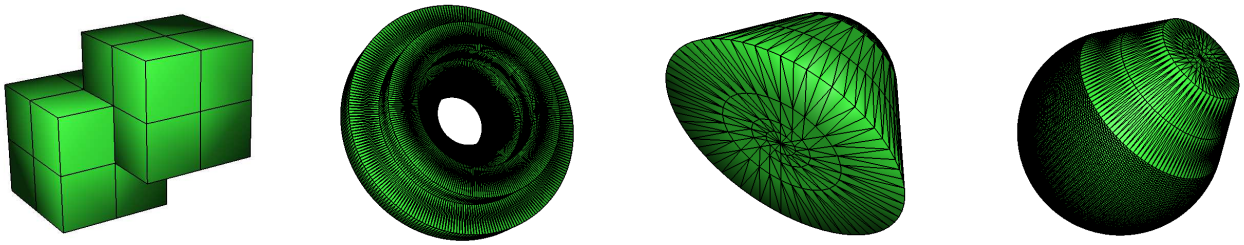
To measure the quality of our reconstruction, we used a number of synthetic scalar and gradient datasets. (See Figure 19.) Given a point p, let $f_p^{L_1}(q)$ be the L_1 distance from p to q. A *Cube* dataset is generated by sampling $f_p^{L_1}$ and its gradients on vertices of the regular grid.

Level sets of $f_p^{L_1}$ are cubes whose edges are parallel to the coordinate axes and whose facets are orthogonal to those axes. By rotating $f_p^{L_1}$ around p, we can generate a scalar field whose level sets are cubes that are not axis-aligned. By taking the minimum of two (rotated) scalar fields, $f_p^{L_1}$ and $f_{p'}^{L_1}$, centered around two different points, p and p', respectively, we get a scalar field whose level sets are the boundaries of the unions of the two cubes. (See Figure 19(a).) We call a regular grid sampling of such a scalar field and its gradients a *TwoCubes* dataset. The isosurface of a TwoCubes dataset has twenty 0-dimensional features, fourteen of which are cube corners and six of which are "saddle points" where the two boundaries of two cubes meet. We use the TwoCubes datasets with various rotations as test sets for the reconstruction of 0-dimensional features.

Let ℓ be a line. Let $f_\ell^{Cyl}(q)$ be the Euclidean distance from point q to ℓ. The level sets of $f_\ell^{Cyl}(q)$ are infinite cylinders around ℓ. Let $f_{\ell,r}^{Cyl\times 2}(q)$ equal $|f_\ell^{Cyl}(q) - r|$. The level sets of $f_{\ell,r}^{Cyl\times 2}(q)$ are pairs of infinite cylinders at equal distances from the cylinder of radius r around ℓ. Let $f_{p,\ell}^{Pl\times 2}(q)$ be the (unsigned) distance from q to the plane that contains point p and is orthogonal to line ℓ. Let $f_{p,\ell,r}^{Ann}(q)$ be the maximum of $f_{\ell,r}^{Cyl\times 2}(q)$ and of $f_{p,\ell}^{Pl\times 2}(q)$. The level sets are the boundaries of thickened annuli. (See Figure 19(b).)

The width of the thickened annuli defined by $f_{p,\ell,r}^{Ann}$ equals their height. We can adjust change the difference between the width and height by adding constants to $f_{\ell,r}^{Cyl\times 2}$ or $f_{p,\ell}^{Pl\times 2}(q)$. Let $f_{p,\ell,r,c_1,c_2}^{Ann}(q)$ be the maximum of $f_{\ell,r}^{Cyl\times 2}(q) + c_1$ and of $f_{p,\ell}^{Pl\times 2}(q) + c_2$. If c_1 is greater than c_2, then the height is $c_1 - c_2$ units greater than the width. If c_2 is greater than c_1, then the width is $c_2 - c_1$ units greater than the height.

Define $f_{p,\ell,r,c}^{Fr}(q)$ as the minimum of $f_{p,\ell,r,c,0}^{Ann}(q)$ and $f_{p,\ell,r,0,c}^{Ann}(q)$.



(a) TwoCubes (b) Flange (c) Smooth Tip Cone (d) Cannon

Figure 20: Polygonal meshes representing level sets.

The level sets of $f^{Fr}_{p,\ell,r,c}(q)$ are the boundaries of the unions of two thickened annuli. (See Figure 19(c).) One annuli has height c units greater than its width while the other has width c units greater than its height. A *Flange* dataset is a regular grid sampling of $f^{Fr}_{p,\ell,r,c}(q)$ and its gradients. The Flange dataset has no 0-dimensional features. We use the Flange datasets as test sets for the reconstruction of 1-dimensional features.

We use two other types of datasets to test our algorithm on dihedral angles other than 90°. A cone is defined by an apex p, an axis direction ζ, and an angle $\alpha < 90^\circ$. The cone is the union of all the rays from p forming angle α with ζ. Let ℓ be the directed line through p with direction ζ, i.e. the cone axis. Let $\pi_\ell(q)$ be the projection of point q on ℓ. Let $D_1(q)$ be the distance from p to $\pi_\ell(q)$ and let $D_2(q)$ be the signed distance from $\pi_\ell(q)$ to p. (The distance is negative if $\pi_\ell(q) - p$ points in the opposite direction from ζ.) Define $f^{Cone}_{p,\zeta,\alpha}(q)$ as $\cos(\alpha)D_1(q) - \sin(\alpha)D_2(q)$. Function $f^{Cone}_{p,\zeta,\alpha}(q)$ is the signed distance from point q to its orthogonal projection on the cone, when that orthogonal projection exists. The level sets of $f^{Cone}_{p,\zeta,\alpha}(q)$ are cones with axis ℓ.

We truncate the cone by defining a plane orthogonal to ℓ. Let p' be a point on ray $p + t\zeta$ where ζ is a unit vector. Let $f^{Pl}_{p',\zeta}(q)$ equal $(q-p')\cdot\zeta$, the (unsigned) distance from q to the plane containing p' orthogonal to ζ. Define

$$f^{Fr}_{p,\zeta,\alpha,p'}(q) = \max\left(f^{Cone}_{p,\zeta,\alpha}(q), f^{Pl}_{p',\zeta}(q), f^{Pl}_{p',-\zeta}(q)\right).$$

The level sets of $f^{Fr}_{p,\zeta,\alpha,p'}(q)$ are frustra. (See Figure 19(d).)

The level sets of $f^{Fr}_{p,\zeta,\alpha,p'}(q)$ have two closed curves forming 1-dimensional features. The dihedral angle on one of these curves is $90^\circ + \alpha$ while the dihedral angle on the other is $90^\circ - \alpha$. To generate a field with dihedral angles of $90^\circ + \alpha$ or $90^\circ - \alpha$, but not both, we modify $f^{Fr}_{p,\zeta,\alpha,p'}(q)$ so that the level sets are capped by spheres at one end.

Define

$$f^{SCone}_{p,\zeta,\alpha,p'}(q) = \begin{cases} |q-p'| & \text{if } \angle(q,p',p) \le 90^\circ - \alpha, \\ f^{Cone}_{p,\zeta,\alpha}(q) & \text{if } \angle(q,p',p) > 90^\circ - \alpha, \end{cases}$$

$$f^{TCone}_{p,\zeta,\alpha,p'}(q) = \max\left(f^{SCone}_{p,\zeta,\alpha,p'}(q), f^{Pl}_{p',\zeta}\right).$$

A level set of $f^{SCone}_{p,\zeta,\alpha,p'}(q)$ is a cone with its tip smoothed. A level set of $f^{TCone}_{p,\zeta,\alpha,p'}(q)$ is a truncated cone with its tip smoothed. The level set has a single closed curve forming a 1-dimensional feature with dihedral angle $90^\circ - \alpha$. The *Smooth Tip Cone* dataset is a regular sampling of $f^{TCone}_{p,\zeta,\alpha,p'}(q)$ and its gradients. We use Smooth Tip Cone datasets to evaluate reconstruction of dihedral angles less than 90°.

Let p' and p'' be points on the ray $p + t\zeta$ where $|p-p'|$ is less than $(1-\sin(\alpha))|p-p''|$. Define

$$f^{X}_{p,\zeta,\alpha,p''}(q) = \begin{cases} |q-p''| & \text{if } \angle(q,p',p) > 90^\circ - \alpha, \\ f^{Cone}_{p,\zeta,\alpha}(q) & \text{if } \angle(q,p',p) \le 90^\circ - \alpha. \end{cases}$$

$$f^{Can}_{p,\zeta,\alpha,p',p''}(q) = \max\left(f^{X}_{p,\zeta,\alpha,p''}(q), f^{Pl}_{p',-\zeta}\right).$$

A level set of $f^{Can}_{p,\zeta,\alpha,p'}(q)$ has the shape of a cannon. The level set has a single closed curve forming a 1-dimensional feature with dihedral angle $90^\circ + \alpha$. The *Cannon* dataset is a regular sampling of $f^{Can}_{p,\zeta,\alpha,p'}(q)$ and its gradients. We use Cannon datasets to evaluate reconstruction of dihedral angles greater than 90°.

13.2 Gradients

Algorithm SHREC requires gradients at the grid vertices. Exact formulas can be given for gradients for each of the scalar fields described in the previous section. However, if scalar data is acquired from scanning devices such as CT scanners, only scalar information is available. In [7], we describe an algorithm, Religrad, for constructing a set of reliable gradients from scalar data. We evaluated SHREC both on gradients computed by exact formulas and on the gradients produced by Religrad from the scalar data.

13.3 Measurements

For each test scalar field f and test isovalue σ, we constructed a polygonal mesh which represented the level set, $f^{-1}(\sigma)$. (See Figure 20.) The polygonal mesh was designed specifically for each surface to accurately represent the 0-dimensional and 1-dimensional features on the surface.

To compare isosurfaces constructed by different software on different scales, we rescaled every isosurface and polygonal mesh to lie in the unit cube. Our isosurfaces were computed from grids of $150 \times 150 \times 150$ or $200 \times 200 \times 200$ so this rescaled the cube size to about 0.005 units. We computed the angular distance in the extended ε-neighborhood as defined in Section 12 between each isosurface and the corresponding polygonal mesh. We also computed the number of triangles whose normals differed by more than 30, 40 or 50 degrees from polyhedral triangles in the extended ε-neighborhood around each triangle. We set ε equal to 0.01 or about twice the cube size for the extended neighborhood.

We also used the degree test as described in [1] to measure errors in reconstructing sharp features. We extracted the 1-skeleton of isosurface edges with dihedral angle less than 140° and counted the number of vertices in the 1-skeleton with degree other than two. For an isosurface Σ, let $N_{\neq 2}(\Sigma)$ be the number of vertices in the 1-skeleton with degree other than two. Isosurfaces from the Flange, Smooth Tip Cone and Cannon datasets should have zero vertices with degrees other than two. The number of degree errors for these isosurfaces is $N_{\neq 2}(\Sigma)$. Isosurfaces from the TwoCubes datasets

Dataset Type	Num Datasets	Grid Size	Isovalue	Avg Num Active Cubes	Avg Num Iso Vert	Avg Num Iso Tri	Avg Time
TwoCubes	34	$150 \times 150 \times 150$	20.1	25K	22K	45K	0.8 sec
Flange	39	$200 \times 200 \times 200$	10.1	90K	75K	150K	1.8 sec
Smooth Tip Cone	15	$100 \times 100 \times 100$	10.2	4K	3.5K	7K	0.2 sec
Cannon	15	$100 \times 100 \times 100$	0.0	7K	7K	14K	0.2 sec

Table 1: Synthetic test dataset sizes, isovalues, and average statistics on isosurfaces produced by SHREC. Average number of active cubes, average number of isosurface vertices, average number of isosurface triangles, and average SHREC running time. All isosurface quadrilaterals are triangulated before counting the number of isosurface triangles. Because SHREC merges isosurface vertices, the average number of isosurface vertices is less than the average number of active cubes.

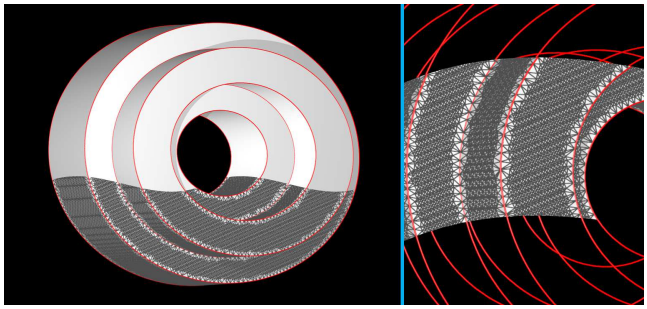


Figure 21: SHREC Flange isosurface. "Sharp" edges (with dihedral angle less than 140°) are marked in red. "Smooth" edges are in dark gray. The magnified region shows a blend of the "sharp" edges with a subset of the "smooth" edges.

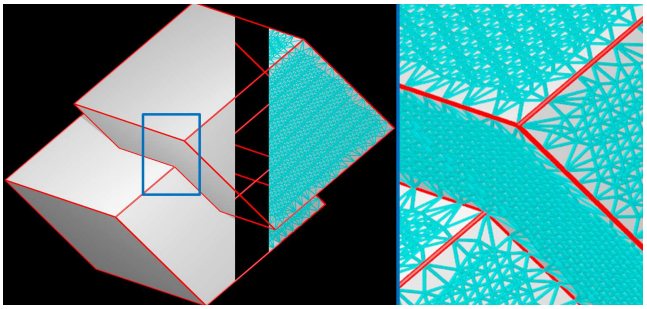


Figure 22: SHREC TwoCubes isosurface. "Sharp" edges (dihedral angle less than 140°) are marked in red. Smooth edges are shown in cyan. The magnified region shows the output mesh edges around a corner.

should have exactly twenty vertices with degrees other than two. The number of degree errors for these isosurfaces is $|N_{\neq 2}(\Sigma) - 20|$.

13.4 Isosurface Reconstruction on Synthetic Data

We tested SHREC on 39 Flange datasets with 39 different orientations and 34 TwoCubes datasets with 34 different orientations. Table 1 presents dataset sizes, isovalues, and average isosurface sizes. We tested SHREC both on gradients produced by exact formulas and on gradients produced by Religrad. We set the singular value threshold ε to 0.1, and used $9 \times 9 \times 9$ subgrids for gradient selection. We measured the oriented angle distance between the SHREC isosurfaces and the polygonal meshes representing the level sets. We also counted the number of isosurface triangles whose normal differences from the polyhedral mesh were above 30°, 40° and 50°. Finally, we measured the number of degree errors in the 1-skeleton of sharp isosurface edges.

Figure 21 shows a Flange isosurface produced by SHREC using exact gradients. The magnified region shows edges with dihedral angle less than 140° in red blended with a subset of the "non-sharp" edges.

Figure 22 shows a TwoCubes isosurface and isosurface edges constructed by SHREC. SHREC reproduces the 0-dimensional features with single mesh vertices and the 1-dimensional features with a sequence of mesh edges with dihedral angle 90°.

With exact gradients, all the Flange isosurfaces had oriented angle distance under 15 degrees from the polygonal meshes. They also had no degree errors in the 1-skeletons of sharp isosurface edges, i.e. all the vertices in the 1-skeletons had degree two.

Figure 23 displays oriented angle distance and degree error information for the 39 Flange isosurfaces when gradients were produced using Religrad. As shown in Figures 23(a) and 23(e), only one isosurface had oriented angle distance greater than 50° and more than half had oriented angle distance less than 40°. No isosurfaces had more than 5 triangles with angle distance greater than 40° and no isosurfaces had more than 15 triangles with angle distance greater than 30°.

Figure 23(b) shows the 1-skeleton degree errors for the 39 Flange isosurfaces when gradients were produced using Religrad. Thirteen isosurfaces had no degree errors. No isosurface had more than fourteen degree errors.

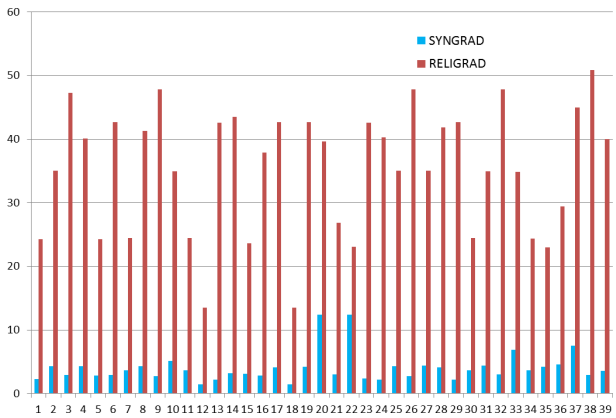
Figure 23 displays angle distance and degree error information for the 34 TwoCubes isosurfaces produced by SHREC. Figure 23(c) shows the oriented angle distance using exact gradients and Religrad gradients. As with the exact gradients in the Flange datasets, the angle distance is very low (less than 5°) for all the TwoCubes isosurfaces produced using exact gradients. For most isosurfaces produced using Religrad gradients, the angle distance is still low (under 15°), although six isosurfaces have angle distances above 30°. The high angle distances indicate errors in the SHREC reconstruction of sharp features.

Figure 23(f) gives the number of TwoCubes Religrad isosurface triangles whose normal differences to the polyhedral mesh are above 30°, 40° and 50°. No isosurface has more than 10 triangles with normal difference above 30°.

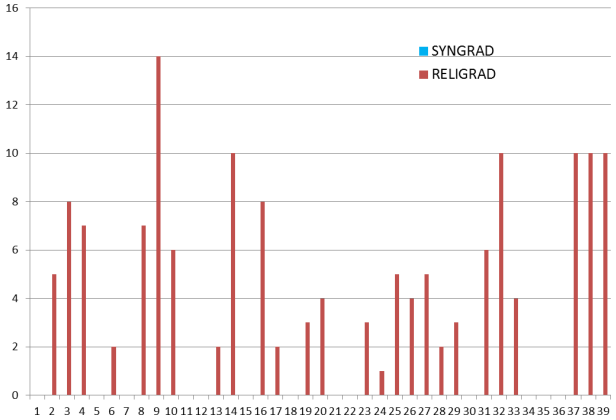
With exact gradients, the 34 TwoCubes isosurfaces had no 1-skeleton degree errors, i.e. they all had exactly 20 vertices with degree three in the 1-skeleton of their sharp edges. Figure 23(d) shows the degree errors in the 1-skeleton of sharp edges when gradients were produced using Religrad. 29 out of the 34 TwoCubes isosurfaces had no degree errors. Of the 5 isosurfaces with degree errors, the maximum number of degree errors was four.

To measure the dihedral angles other than 90°, we ran SHREC on 15 Cannon datasets with 15 different orientations and 14 Smooth Tip Cone datasets with 14 different orientations. The 1-dimensional features in the Cannon level sets had dihedral angles of 120°. The 1-dimensional features in the Smooth Tip Cone level sets had dihedral angles of 60°. As expected, SHREC produced more errors than on the Flange or TwoCubes datasets, although it still did quite well.

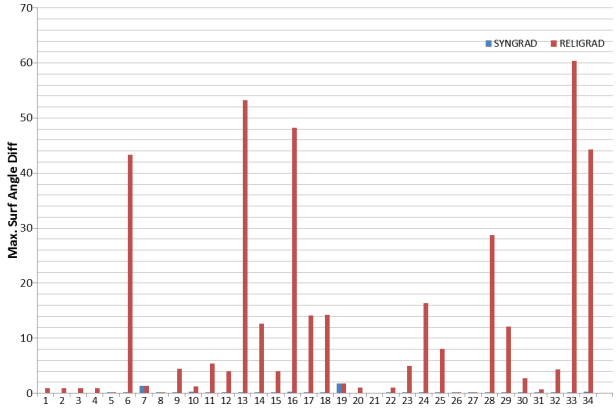
Figure 24 shows Cannon and Smooth Tip Cone isosurfaces



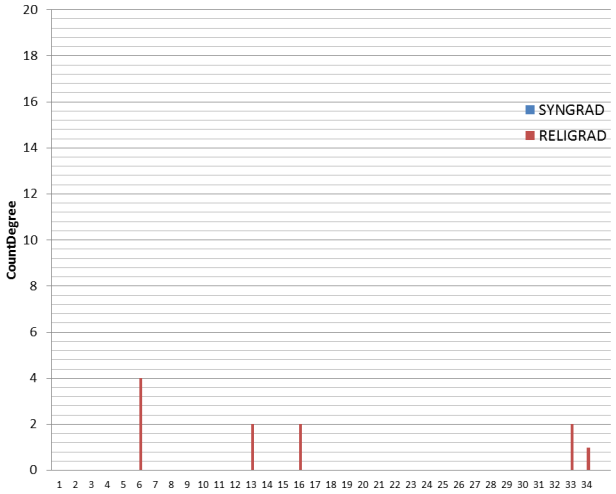
(a) Flange: Oriented angle distance



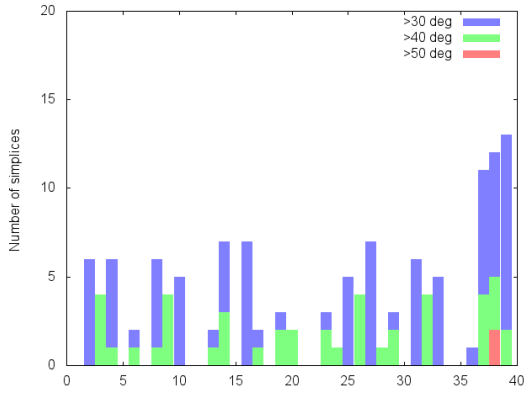
(b) Flange: Degree errors



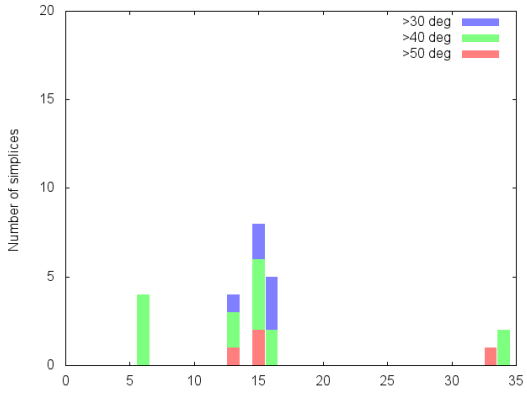
(c) TwoCubes: Oriented angle distance



(d) TwoCubes: Degree errors



(e) Flange: Triangle normal differences

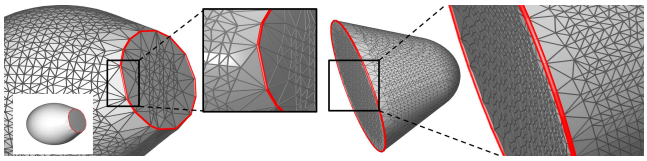


(f) TwoCubes: Triangle normal differences

Figure 23: Results of SHREC using exact gradients and Religrad gradients on 39 Flange datasets and 34 TwoCubes datasets. (a) Oriented angle distances between Flange isosurfaces (exact and Religrad gradients) and polygonal mesh. (b) Number of degree errors in the Flange 1-skeletons of sharp edges (Religrad gradients). (SHREC using exact gradients produces no degree errors.) (c) Oriented angle distances between TwoCubes isosurfaces (exact and Religrad gradients) and polygonal mesh. (d) Number of degree errors in the TwoCubes 1-skeletons of sharp edges (Religrad gradients). (SHREC using exact gradients produces no degree errors.) (e) Number of Flange isosurface triangles with normal difference to polygonal mesh above 30, 40 and 50 degrees. (f) Number of TwoCubes isosurface triangles with normal difference to polygonal mesh above 30, 40 and 50 degrees.

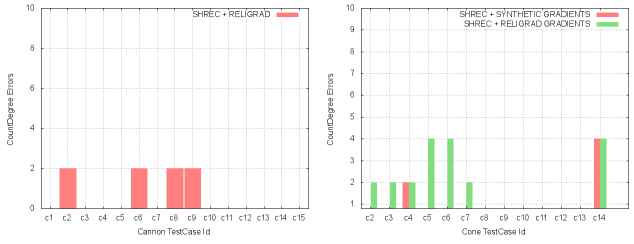
Software	Algorithm	URL	References
EMC (IsoEx)	Extended Marching Cubes	www.graphics.rwth-aachen.de/software	[22]
EMCpoly	Extended Marching Cubes	web.cse.ohio-state.edu/research/graphics/isotable	
PolyMender	Dual Contouring	www.cse.wustl.edu/ taoju/code/polymender.htm	[19, 20, 31]
SingularCocone	Singular Cocone	web.cse.ohio-state.edu/ tamaldey/cocone.html	[8, 12, 13]
MergeSharp	MergeSharp	web.cse.ohio-state.edu/research/graphics/isotable	[1, 6]
SHREC	SHREC	web.cse.ohio-state.edu/research/graphics/isotable	

Table 2: Surface reconstruction software. EMC is a sample implementation of Extended Marching Cubes. EMCpoly is a modification of EMC which creates TwoCubes and Flange isosurfaces. PolyMender is an implementation of dual contouring for fixing polygonal meshes. SingularCoconeis an implementation of a Voronoi based algorithm for surface reconstruction from point clouds. The algorithm handles 0 and 1 dimensional features including non-manifold features. MergeSharp is an implementation of a dual contouring reconstruction algorithm which uses cube merging to obtain better reconstruction around sharp features. SHREC is an implementation of the algorithm in this paper.



(a) Cannon (b) Smooth Tip Cone

Figure 24: SHREC Cannon and Smooth Tip Cone isosurfaces (Religrad gradients.) (a) Cannon isosurface. 1-dimensional feature has dihedral angle 120°. (b) Smooth Tip Cone isosurface. 1-dimensional feature has dihedral angle 60°.



(a) (b)

Figure 25: Results of SHREC on 15 Cannon and 14 Smooth Tip Cone datasets. (a) Number of degree errors on 1-skeleton of Cannon isosurfaces constructed using Religrad gradients. (Cannon isosurfaces constructed using exact gradients had no degree errors.) (b) Number of degree errors on 1-skeleton of Smooth Tip Cone isosurfaces constructed using Religrad gradients (green) and exact gradients (red).

produced by SHREC using Religrad gradients. In each, the 1-dimensional is represented by a sequence of isosurface mesh edges with the appropriate dihedral angles.

Figure 25 gives the number of degree errors in the 1-skeleton of sharp edges for Cannon and Smooth Tip Cone isosurfaces. SHREC using exact gradients produced no degree errors on the 15 Cannon datasets. SHREC using Religrad produced degree errors on only four Cannon datasets. No SHREC Cannon Religrad isosurface had more than two degree errors. SHREC using exact gradients produced degree errors in only two of the Smooth Tip Cone datasets, one with two errors and one with four. SHREC using Religrad gradients produced degree errors on 7 out of the 14 Smooth Tip Cone datasets. The maximum number of degree errors in any Smooth Tip Cone isosurface was 4.

14 Comparison with Other Algorithms

We compared SHREC with software implementations of four other algorithms: MergeSharp [1, 6], PolyMender [19], EMC (Extended Marching Cubes) [22] and SingularCocone [12]. Table 2 contains descriptions of the software.

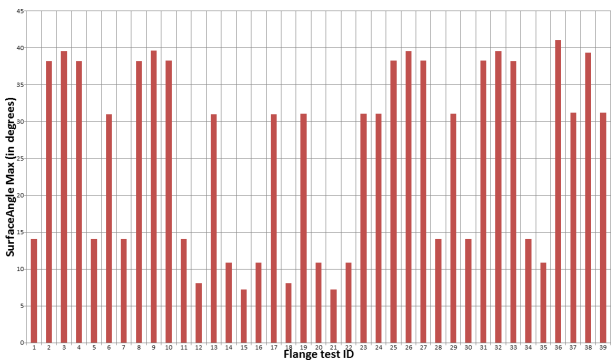
Input to both SHREC and MergeSharp is a regular grid sampling of a scalar field and the gradients at the grid vertices. Input to the software implementations of the other algorithms is very different from input to SHREC or inputs to each other, so one should be extremely careful in make comparisons between these algorithms based on the results presented here. For instance, on average, EMC has fewer degree errors than PolyMender or SingularCocone, but EMC computes scalar, distance and normal values from formulas hard coded into the software. PolyMender and SingularCocone (and SHREC and MergeSharp) would certainly do much better if their input came from formulas hard coded into their software.

SingularCocone has the most degree errors, but SingularCocone is designed for reconstruction from point samples of a (possibly non-manifold) surface, not for reconstruction from sampling of a scalar field. Input to SingularCocone is point samples of a surface, not scalar grid or gradient information.

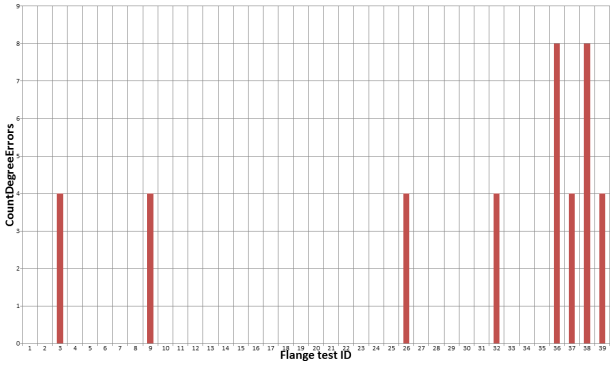
Comparison with MergeSharp. Input to MergeSharp is a regular grid sampling of a scalar field and the gradients at the grid vertices. We ran MergeSharp on the same 39 Flange and 34 TwoCubes synthetic datasets that we applied to SHREC in Section 13.4. We used the exact gradients on MergeSharp, not the Religrad gradients. Figures 26(a) and 26(c) show the oriented angle distances between the MergeSharp isosurfaces and the polygonal mesh isosurfaces. Five of the MergeSharp TwoCubes isosurfaces have angle distance near 180° indicating "flipped" triangles produced by folds in the mesh. Figures 26e and 26f present the number of triangles with angle difference greater than 30, 40 and 50 degrees in the MergeSharp Flange and TwoCubes isosurfaces. 24 out of 39 of the Flange isosurfaces and 18 out of 34 of the TwoCubes isosurface have angle distance above 30° indicating significant problems in the reconstruction of sharp features.

Figures 26(b) and 26(d) gives the number of degree errors in the 1-skeleton's of the sharp edges. 8 out of 39 of the Flange isosurfaces and 15 out of 34 of the TwoCubes isosurfaces have degree errors. The maximum number of degree errors is eight.

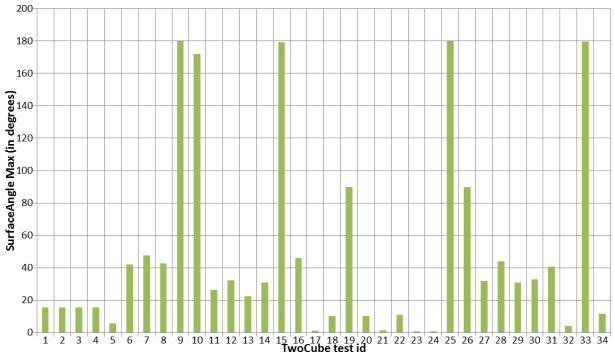
Note that all the results in Figure 26 are for MergeSharp isosurfaces produced from EXACT gradients. Thus, they should not be compared with the SHREC results in Figure 23 for Flange isosurfaces produced using Religrad gradients. The 39 SHREC Flange isosurfaces produced using exact gradients all have angle distance under 15° compared with angle distances above 30° for 24 of the corresponding MergeSharp isosurfaces. None of the SHREC Flange isosurfaces produced using exact gradients have degree errors compared with 8 MergeSharp isosurfaces with 4 to 8 degree errors.



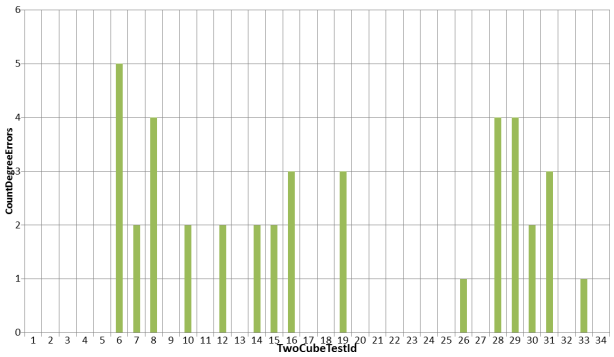
(a) Flange: Oriented angle distance



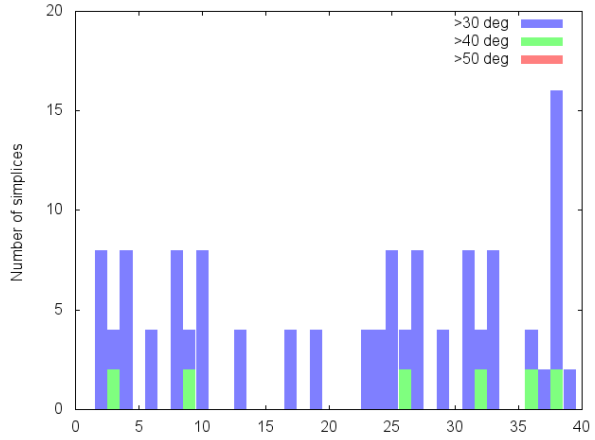
(b) Flange: Degree errors



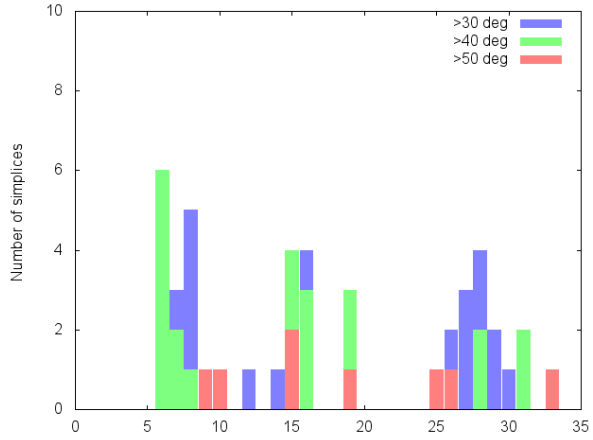
(c) TwoCubes: Oriented angle distance



(d) TwoCubes: Degree errors

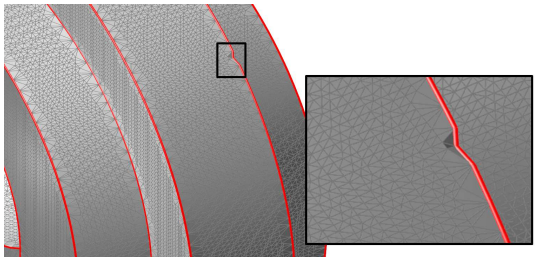


(e) Flange: Triangle normal differences

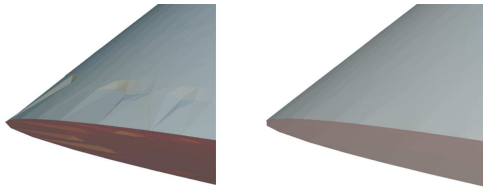


(f) TwoCubes: Triangle normal differences

Figure 26: Results of MergeSharp on 39 Flange and 34 TwoCubes datasets (exact gradients). (a) Oriented angle distance to polygonal Flange meshes. (b) Degree errors in 1-skeleton of sharp edges of Flange datasets. (c) Oriented angle distance to polygonal TwoCubes meshes. (d) Degree errors in 1-skeleton of sharp edges of TwoCubes datasets. (f) Number of triangles with normal difference to polygonal Flange mesh above 30, 40 and 50 degrees. (No Flange isosurface triangles have normal difference above 50°.) (e) Number of triangles with normal difference to polygonal TwoCubes mesh above 30, 40 and 50 degrees.

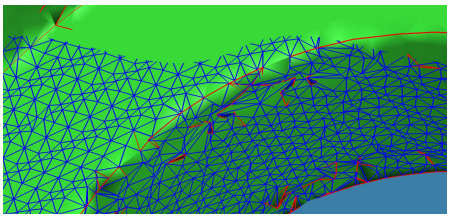


(a) Notch in MergeSharp Flange isosurface

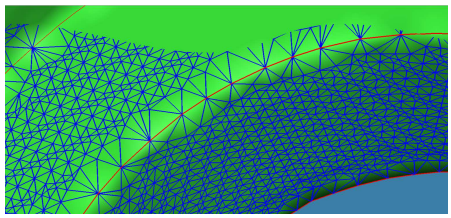


(b) Dimpled MergeSharp isosurface (left) and smooth SHREC isosurface (right)

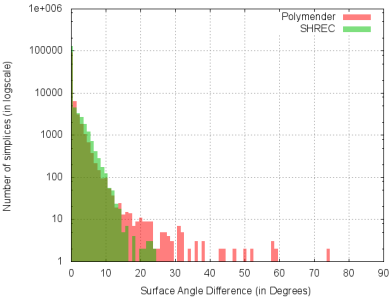
Figure 27: (a) Notch in a MergeSharp Flange isosurface. (b) Dimples in a MergeSharp Cone isosurface (left). Compare with smooth SHREC isosurface (right).



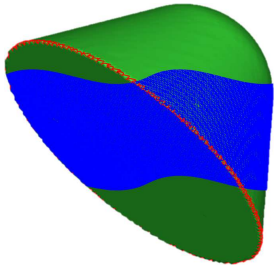
(a) PolyMender Flange isosurface



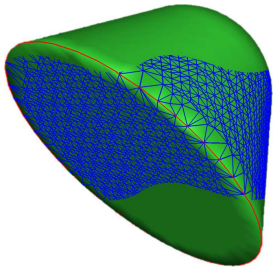
(b) SHREC Flange isosurface



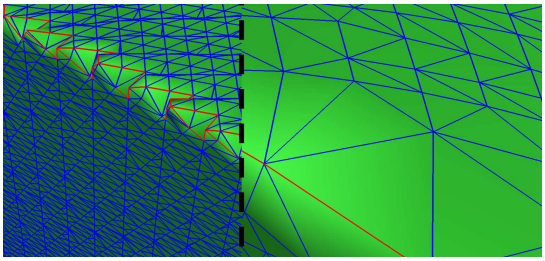
(c) Triangle normal differences



(d) PolyMender



(e) SHREC



(f) Magnified Smooth Tip Cone isosurfaces

Figure 28: PolyMender and SHREC comparison. Mesh edges with dihedral angle below 140° are colored red. (a) PolyMender Flange isosurface. (b) Corresponding SHREC Flange isosurface. (c) Distribution of differences of triangle normals between PolyMender isosurface and polygonal mesh (red) and between SHREC isosurface and polygonal mesh (green). (d) PolyMender Cone isosurface. (e) Corresponding SHREC Cone isosurface. (f) Magnified view of PolyMender isosurface (left) and corresponding SHREC isosurface (right).

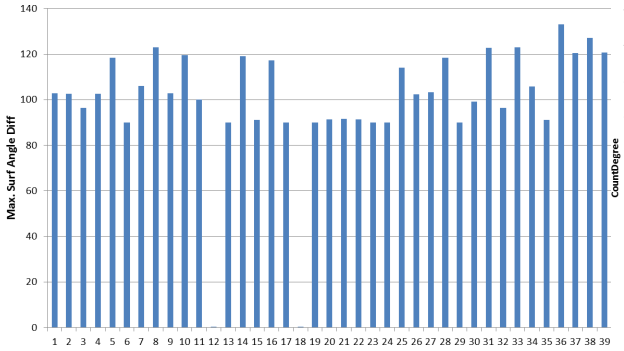
As shown in Figure 23(c), the 39 SHREC TwoCubes isosurfaces produced using exact gradients all have angle distance under 2°. 18 of the corresponding MergeSharp TwoCubes isosurfaces have angle distance above 30°. None of the SHREC TwoCubes isosurfaces produced using exact gradients have degree errors compared with 15 of the TwoCubes isosurfaces with 2 to 5 degree errors.

Figure 27(a) shows "notches" in a MergeSharp Flange isosurface. Figure 27(b) shows gentle dimples in a MergeSharp Cone isosurface (left image). The corresponding SHREC isosurface is smooth and does not have these dimples.

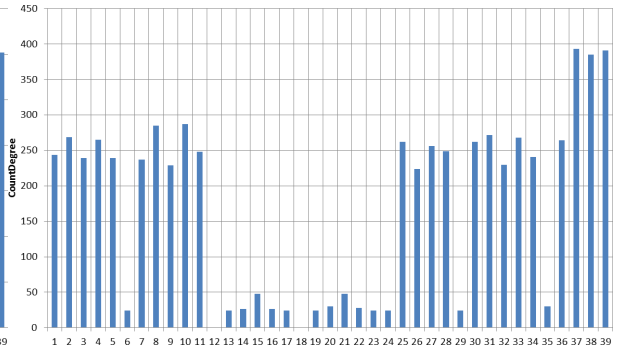
Comparison with PolyMender. PolyMender is an implementation of mesh repairing algorithm from Ju [19]. Input to PolyMender is a set of triangles representing a surface, but not necessarily properly connected in a polygonal mesh. PolyMender uses the input triangles to build a regular scalar grid representing the signed distance to the surface. It also uses the input triangles to determine surface normals on the surface. It extracts an isosurface mesh using the dual contouring algorithm described in [20, 31]. The extracted isosurface mesh is the "mended" polygonal mesh.

Because PolyMender contains an implementation of the dual contouring algorithm from [20, 31], we used it to compare SHREC and the algorithm from [20, 31]. Our inputs to PolyMender were the polygonal meshes (Figure 20) designed specifically for each surface to accurately represent the 0-dimensional and 1-dimensional features on the surface.

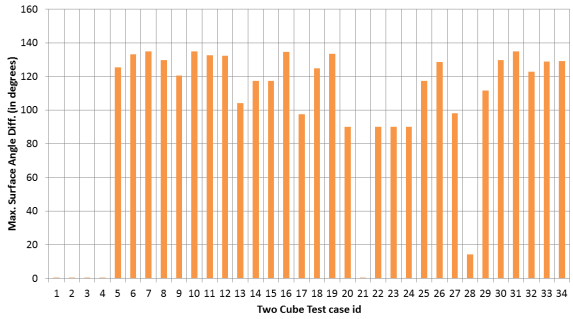
PolyMender builds a multiresolution isosurface using an octree instead of a fixed regular grid. The highest resolution is determined by the depth of the octree. For all tests, we ran PolyMender with octree depth set of 7. At the highest resolution, this octree sampled data from a regular grid with dimensions $2^7 \times 2^7 \times 2^7$ or $128 \times 128 \times 128$. We set the scale to 0.9 and used defaults for all other parameters.



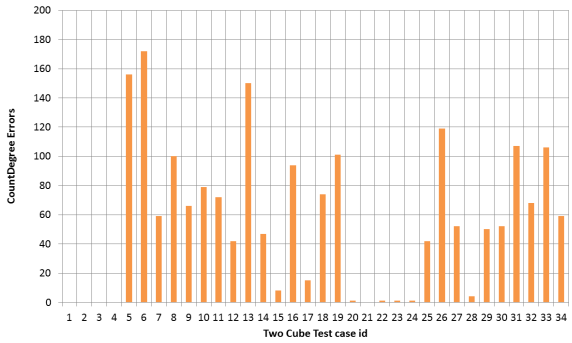
(a) Flange: Oriented angle distance



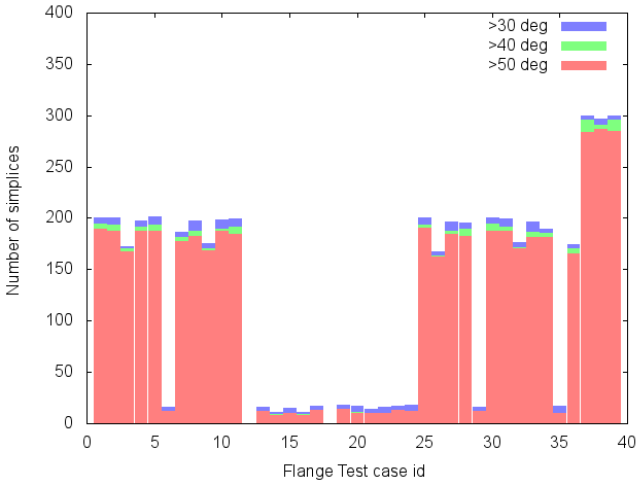
(b) Flange: Degree errors



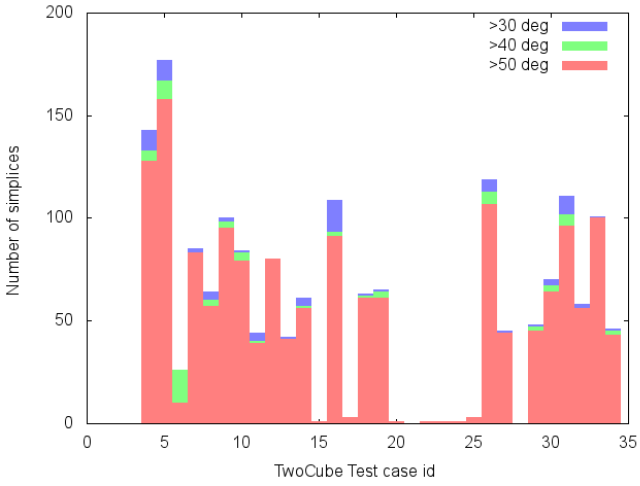
(c) TwoCubes: Oriented angle distance



(d) TwoCubes: Degree errors

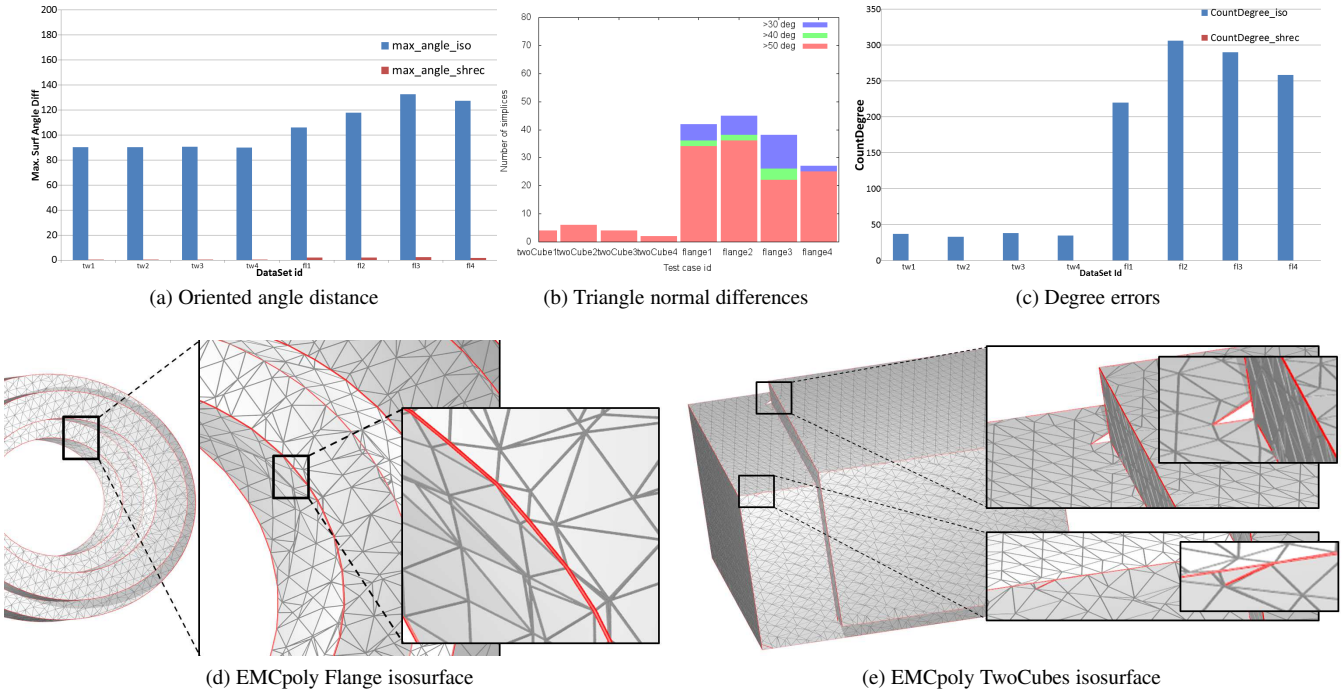


(e) Flange: Triangle normal differences



(f) TwoCubes: Triangle normal differences

Figure 29: Results of PolyMender on 39 Flange and 34 TwoCubes datasets. (a) Oriented angle distance to polygonal Flange meshes. (b) Degree errors in 1-skeleton of sharp edges of Flange datasets. (c) Oriented angle distance to polygonal TwoCubes meshes. (d) Degree errors in 1-skeleton of sharp edges of TwoCubes datasets. (Note: Y-scale is 0 to 200 compared with y-scale 0 to 450 in (b).) (e) Number of triangles with normal difference to Flange mesh above 30, 40 and 50 degrees. (f) Number of triangles with normal difference to TwoCubes mesh above 30, 40 and 50 degrees. (Note: Y-scale is 0 to 200 compared with y-scale 0 to 400 in (e).)



(a) Oriented angle distance (b) Triangle normal differences (c) Degree errors

(d) EMCpoly Flange isosurface (e) EMCpoly TwoCubes isosurface

Figure 30: EMCpoly (Extended Marching Cubes) and SHREC comparison. Tests tw1-tw4 are four TwoCubes scalar fields. Tests fl1-fl4 are four Flange scalar fields. (a) Oriented angle distances for EMCpoly (blue) and SHREC (red). (b) Number of triangles in EMCpoly isosurfaces with oriented angle difference to polygonal mesh above 30, 40 and 50 degrees. (c) Degree errors in 1-skeleton of EMCpoly isosurface (blue). (SHREC has no degree errors for these cases.) (d) EMCpoly Flange isosurface. The magnified region shows very thin mesh triangles aligned with a 1-dimensional feature. (e) EMCpoly TwoCubes isosurface. The magnified region shows errors folds and degenerate triangles in the EMCpoly isosurface mesh.

Because PolyMender computes its scalar field and surface normals from an input polygonal mesh, we did not think it fair to compare PolyMender to SHREC with exact gradients. Instead, we compared PolyMender to SHREC using Religrad gradients. Since PolyMender surface normals come from triangles on the polygonal meshes used for the angle distance measurements while Religrad gradients are computed from the scalar data, we think this comparison is actually biased in favor of PolyMender.

Figures 28(a) and 28(b) show part of a Flange isosurface reconstructed by PolyMender and a corresponding isosurface produced by SHREC (Religrad gradients). Note the flipped triangles and distorted "sharp" curve (red) in the PolyMender isosurface. Figure 28(c) shows the distribution of differences of triangle normals between the PolyMender isosurface and the polygonal mesh and between the SHREC isosurface and the polygonal mesh. All SHREC triangle normals are within 25° of the polygonal mesh normals, while PolyMender has numerous triangles with normals greater than 30° of the polygonal mesh normals.

Figures 28(d), 28(e) and 28(f) show a PolyMender Smooth Tip Cone isosurface and the corresponding SHREC isosurface (Religrad gradients). The 1-dimensional feature around the base of the cone has a 60° dihedral angle. In the PolyMender isosurface, there are numerous "notches" along the 1-dimensional feature.

We ran PolyMender on the same 39 Flange and 34 TwoCubes meshes that we applied to SHREC in Section 13.4. Figure 29 shows the oriented angle distances from the polygonal meshes and the degree errors in the 1-skeletons of sharp edges. Most of the PolyMender Flange and TwoCubes isosurfaces had an oriented angle distance greater than 90° from the corresponding polygonal mesh normals. 24 out of 39 of the PolyMender Flange isosurfaces had over 150 triangles whose normals were more than 50° from the normals of the corresponding polygonal meshes. In comparison, only one SHREC Flange isosurface (Religrad gradients) had triangles whose normals were more than 50° from the normals of the corresponding polygonal meshes (Figure 23). No SHREC Flange isosurface (Religrad gradients) had more than 15 triangles whose normals were more than 30° from the normals of the corresponding polygonal meshes Almost all of the PolyMender Flange isosurfaces had degree errors and 25 out of 39 had over 200 degree errors. Only 8 out of 39 SHREC Flange isosurfaces (Religrad gradients) had degree errors and the maximum number of degree errors was eight.

PolyMender did a bit better on the TwoCubes isosurfaces, probably because the 1-dimensional features in the TwoCubes level sets are line segments. Only one of the 34 PolyMender TwoCubes isosurfaces had over 150 triangles whose normals were more than 50° from the normals of the corresponding polygonal meshes. 18 out of 34 PolyMender TwoCubes isosurfaces had over 50 triangles whose normals were more than 50° from the normals of the corresponding polygonal meshes. Most of the PolyMender TwoCubes isosurfaces had degree errors, but none had more than 180 degree errors. 8 out of 34 had more than 100 degree errors and 22 out of 34 had more than 40 degree errors. 5 out of 34 SHREC TwoCubes isosurfaces (Religrad gradients) had angle distance greater than 40° to the corresponding polygonal meshes. The same five SHREC isosurfaces had degree errors, but no SHREC isosurface had more than four such errors.

Comparison with EMC (Extended Marching Cubes): EMC is a sample implementation by Mario Botsch of the Extended Marching Cubes algorithm [22]. The program has a function which provides the scalar value of a point cloud dataset, the directed distances along the x, y and z axes to some level set of the point cloud

(a sphere), and the gradients at query points near the level set. Isosurfaces with sharp features are constructed by combining the functions to represent the union, intersection or difference of balls defined by the point cloud datasets.

The union, intersection and difference of balls produces visually interesting surfaces but such surfaces do not seem a realistic approximation to the surfaces found in industrial products. To compare EMC with SHREC, we implemented functions which produce scalar values, directed distances, and gradients, for Cube and Annulus scalar fields. By combining these functions, we the corresponding values for the TwoCubes and Flange scalar fields. We added only functions to provide the relevant scalar field measurements, but did not change the EMC code which generates the isosurface. Our modification of EMC is called EMCpoly.

We ran EMCpoly on four Flange scalar fields with four randomly generated axes directions and four TwoCubes scalar fields with four randomly generated orientations. We ran SHREC on eight corresponding datasets representing the same scalar fields. For both EMCpoly and SHREC, the Flange scalar fields were sampled on a $200 \times 200 \times 200$ regular grid and the TwoCubes scalar fields were sampled on a $150 \times 150 \times 150$ regular grid.

Figure (a) shows the angle distance between the EMCpoly isosurfaces and the polygonal meshes The oriented angle distance of the SHREC isosurfaces is less than 20°, but is over 90° for all of the EMCpoly isosurfaces. Figure 30(b) shows the number of triangles in EMCpoly isosurfaces whose normals differ more than 30, 40 or 50 degrees from the polygonal mesh normals. All four of the EMCpoly Flange isosurfaces had over 20 triangles with normals differing more than 50° from the polygonal mesh. The four EMC TwoCubes isosurfaces were better, with only a few triangles with normals differing more than 50°.

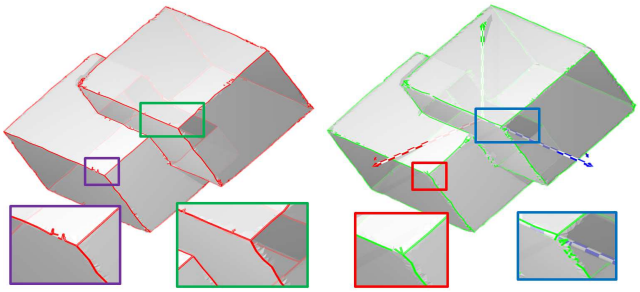
Figure 30(c) shows the degree errors in the 1-skeletons of the EMCpoly isosurfaces. All eight EMCpoly isosurfaces had some degree errors, but each of the TwoCubes isosurfaces had under 40 degree errors while each of the Flange isosurfaces had over 200 errors. None of the eight SHREC isosurfaces had any degree errors. Examples of errors in EMCpoly Flange and TwoCubes isosurfaces are shown in Figures 30(d) and 30(e).

Visually, the sharp features in the EMCpoly isosurface look quite good. The large number of triangles with normals very different from the polygonal mesh normals and the high number of degree errors are probably caused by the large number of near degenerate triangles. Each of the EMCpoly TwoCubes isosurfaces has over 250 triangles (out of 46K-48K) with angle less than 1°. Each of the EMCpoly Flange isosurfaces has over 1500 triangles (out of 200K) with angle less than 1°. In contrast, none of the eight corresponding SHREC isosurfaces had any triangles with angles less than 4°. The SHREC TwoCubes and Flange isosurfaces had about 40K triangles, and $150K$ triangles, respectively.

Small perturbations in the locations of vertices of thin triangles create almost arbitrary normal orientations contributing to the normal differences and degree errors in the EMCpoly isosurfaces. These thin triangles are aligned with the 1-dimensional features so they do not create large visual anomalies in the EMCpoly isosurfaces.

As previously noted, EMCpoly uses hard coded functions to directly compute scalar values, directed distances and gradient information. Thus, the vertex locations both on and near the sharp features are extremely precise. If EMCpoly computed such information from an input mesh as does PolyMender or from scalar and gradient data, as does SHREC, those vertex positions would be much less precise. We conjecture that under those circumstances, the numerous thin triangles would be much more visible, creating numerous visual anomalies.

Comparison with SingularCocone. Extensive research has been done on reconstruction of surfaces with sharp features from point cloud data. Regular grid scalar data can easily be converted to point cloud data by approximating the intersection points between a given level set and the grid edges, and applying a point cloud reconstruction algorithm to the intersection points. Perhaps, this is an effective way to reconstruct isosurfaces with sharp features?



(a) (b)

Figure 32: SingularCocone TwoCubes isosurface. (a) Input cloud is a supersampled polygonal mesh. (b) Input cloud is a supersampled set of isosurface vertices produced by Marching Cubes. The magnified regions show some of the errors. The reconstruction from Marching Cube vertices is worse than the reconstruction from the supersampled polygonal mesh..

To test the efficacy of point cloud reconstruction for isosurface reconstruction, we tested one algorithm, SingularCocone, on reconstructing the TwoCubes isosurfaces. Cube merging in MergeSharp and SHREC plays a similar role to the "protective balls" in SingularCocone, so we thought that SingularCocone would perform better than other point cloud reconstruction algorithms.

We note that we are only evaluating whether SingularCocone is more effective than SHREC in reconstructing isosurfaces. SingularCocone is really built for point cloud data which is much noisier and more difficult to handle than scalar data. SingularCocone also can reconstruct non-manifold surfaces and their non-manifold features. SHREC has no application to reconstruction from point cloud data or to reconstruction of non-manifold surfaces.

As described in Section 3, SingularCocone extracts a surface mesh from a weighted Delaunay triangulation of a set of sample points of the features and the smooth portions of the mesh. SingularCocone requires two inputs: a point cloud and a weighted sampling of the 0 and 1 dimensional features of the surface. Weight of the sampling determines the size of "protecting balls" around surface features. SingularCocone outputs a feature sensitive mesh.

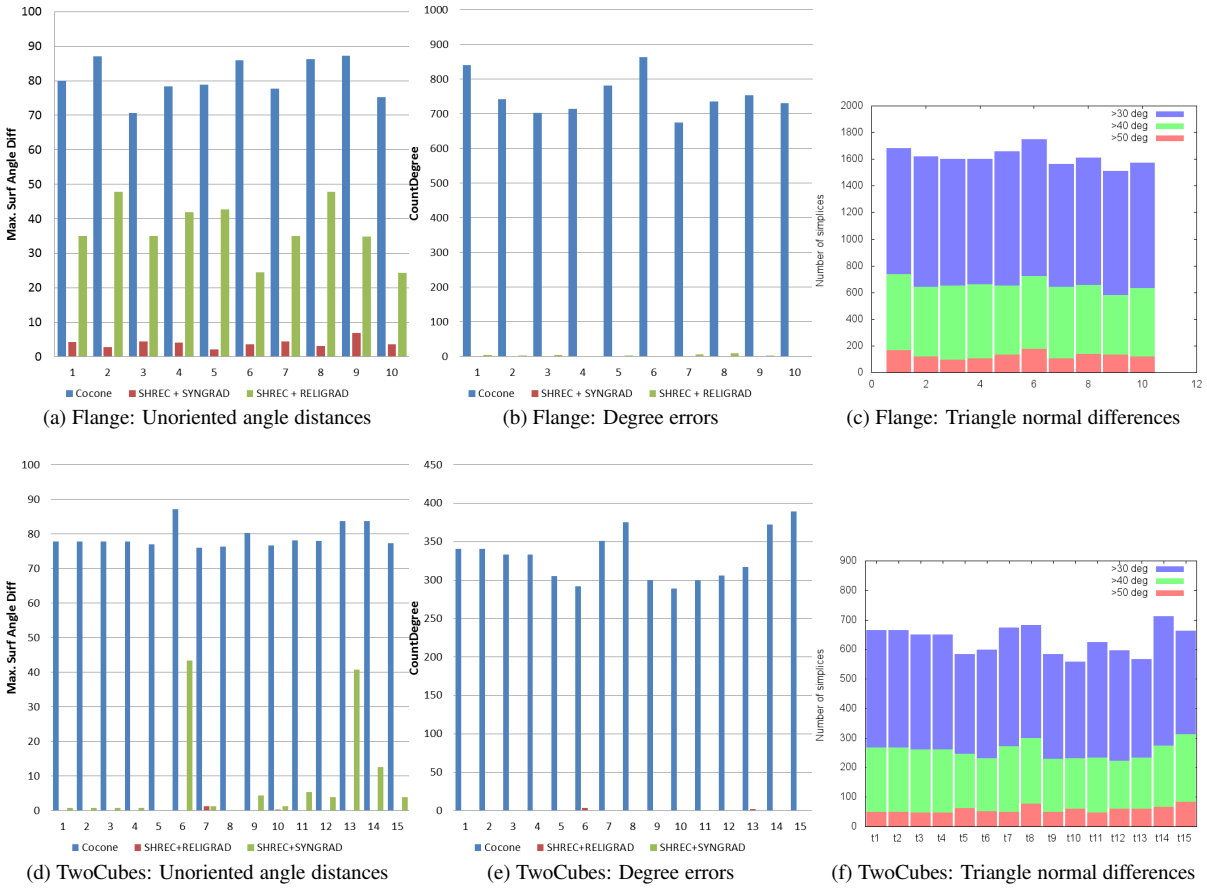
In order to run SingularCocone, we must construct a weighted sampling of surface features from a set of points sampling the surface. We use the algorithm FeatureRecon by Dey et al. [13] to compute sample points on the surface features.

Experimental details: The input to FeatureRecon is a point cloud. We tested two different input point cloud;

1. We applied Marching Cubes [24] to fifteen TwoCubes datasets. We supersampled the vertices of the Marching Cubes mesh using the Monte Carlo point sampling, to generate a point cloud with approximately sixty-five thousand points.

2. We supersampled the polygonal meshes described in Section 13.3 to generate a point cloud with approximately sixty-five thousand points.

The first experiment measured how well FeatureRecon and SingularCocone could reconstruct an isosurface from scalar data. The



(a) Flange: Unoriented angle distances (b) Flange: Degree errors (c) Flange: Triangle normal differences

(d) TwoCubes: Unoriented angle distances (e) TwoCubes: Degree errors (f) TwoCubes: Triangle normal differences

Figure 31: Comparison of SHREC and SingularCocone on 10 Flange and 15 TwoCubes datasets. Point clouds for SingularCocone (and FeatureRecon) were computed by supersampling the polygonal meshes. (a) Unoriented angle distances from Singular Cocone (blue) Flange isosurfaces and from SHREC Flange isosurfaces (exact gradients in red, Religrad gradients in green) to polygonal Flange meshes. (b) Degree errors in the 1-skeleton of the sharp edges for SingularCocone Flange isosurfaces (blue) and SHREC isosurfaces (Religrad gradients, red). (SHREC isosurfaces based on exact gradients have no degree errors.) (c) Number of SingularCocone triangles with normal difference to Flange mesh above 30°, 40° and 50°. (d) Unoriented angle distances from Singular Cocone (red) TwoCubes isosurfaces and from SHREC TwoCubes isosurfaces (Religrad gradients, green) to polygonal TwoCubes meshes. (e) Degree errors in the 1-skeleton of the sharp edges for SingularCocone TwoCubes isosurfaces (red) and SHREC isosurfaces based (Religrad gradients, blue). (f) Number of SingularCocone triangles with normal difference to TwoCubes mesh above 30°, 40° and 50°.

second experiment measured how well FeatureRecon and SingularCocone reconstructed surface meshes from point cloud samplings of those meshes.

The two different point clouds are used as input to FeatureRecon to extract sample points on the surface features. The feature sample points generated by FeatureRecon along with the supersampled point cloud are used as input to SingularCocone.

FeatureRecon has nine separate parameters. Experimentally and also noted by the authors Dey et al. [13], we found FeatureRecon to be heavily reliant on parameter fine tuning. We used the following parameter values for the TwoCubes datasets, (as suggest by the authors Dey et al. [13]). $-t = 25, -fl = 0.04, -cl = 0.06, -dc = 0, -\rho 3 = 0.32, -\rho 1 = 0.0, -rc = 3$.

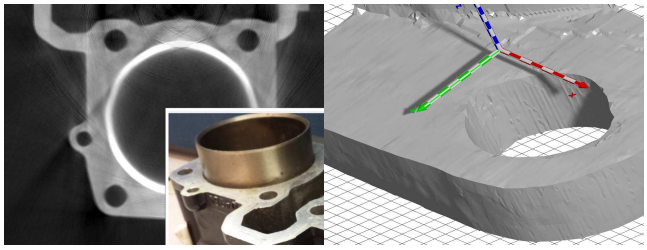
Figure 32 shows a TwoCubes mesh reconstructed by SingularCocone. The associated "sharp" edges are also shown. Figure 32(a) shows the reconstruction from the point cloud generated from the polygonal mesh. The red "sharp" edges show that the reconstruction has many errors. The magnified regions show some of the errors along the sharp edges and corners. Figure 32(b) shows the reconstruction from the point cloud generated from running Marching Cubes. The magnified regions show the same regions as Figure 32(a). The reconstruction is worse than SingularCocone reconstruction from the supersampled polygonal mesh.

Figure 31 shows a comparison of SingularCocone and SHREC on 15 different datasets. The inputs to SingularCocone for these tests were generated from the polygonal meshes. The results using sample points created from Marching Cubes are much worse and not shown. Because SingularCocone does not assume the output is a manifold, the triangles in the SingularCocone mesh have arbitrary orientations. Thus, we use the unoriented angle distance to compare the SingularCocone and SHREC meshes.

We ran SingularCocone and SHREC on the same 15 datasets, and used both exact and Religrad gradients for SHREC. Figure 31(d) shows the unoriented angle distance between the SingularCocone isosurfaces and the corresponding polygonal meshes, and the SHREC isosurfaces based on Religrad gradients and the corresponding polygonal meshes. The maximum angle distance was 43° for SHREC with the mean error of 8° for the 15 tests. In comparison, the maximum angle distance was 87° for SingularCocone, with a mean of 79°. Figure 31(f) shows the number of

Dataset	Grid Size	Spacing	Avg Num Isovalue	Avg Num Active Cubes	Avg Num Iso Vert	Iso Tri	Time
Motorcycle Engine	$200 \times 129 \times 62$	$0.27 \times 0.27 \times 0.68$ (mm)	22000	25k	26k	51k	1.9 sec
Volt	$411 \times 431 \times 61$	$1 \times 1 \times 1$	4000	519k	527k	1039k	39 sec
CMM	$500 \times 500 \times 196$	$0.2 \times 0.2 \times 0.31$ (mm)	20000	861k	863k	1726k	60 sec

Table 3: CT dataset sizes, isovalues, and average statistics on isosurfaces produced by SHREC. Average number of active cubes, average number of isosurface vertices, average number of isosurface triangles, and average SHREC running time. All isosurface quadrilaterals are triangulated before counting the number of isosurface triangles.



(a) Single image slice, original machine part (b) Sharp Isosurface

Figure 33: Motorcycle Engine dataset. (a) a slice of the original CT image, note the streaking artifacts introduced during the scanning process. The inset shows the original machine part. Figure (b) shows the sharp mesh.

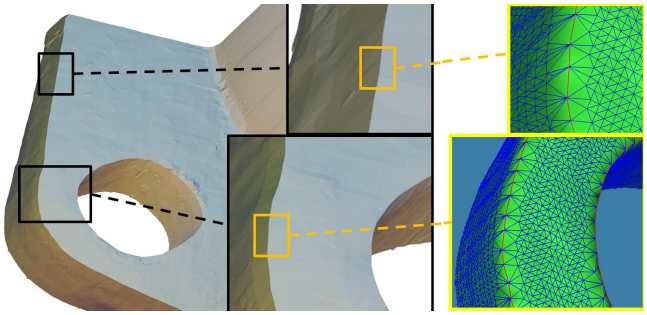


Figure 34: SHREC with Religrad gradients computed from part of industrial CT data (Motorcycle Engine). Magnified regions show "sharp" edges reconstructed. In red, picture from the original item.

SingularCocone triangles with normal difference greater than 30, 40 or 50 degrees to the corresponding polygonal mesh normals. On all the test cases there are large numbers of triangles with normal differences more than 40° and 50°.

Figure 31(e) shows the degree errors for SingularCocone isosurfaces and SHREC isosurfaces based on Religrad gradients. SHREC with Religrad gradients generates degree errors in only two isosurfaces (t6 and t13) and does not generate more than 4 degree errors per isosurface. SHREC with exact gradients generates no degree errors. In contrast, SingularCocone generates an average of 330 degree errors in each isosurface.

15 Experimental Results on CT Data

We applied the SHREC algorithm on industrial X-ray computed tomography (CT) scans of three objects. We set the singular value threshold ε to 0.1, and used $9 \times 9 \times 9$ subgrids for gradient selection. The Motorcycle Engine dataset is an industrial CT scan of a motorcycle engine cylinder. (See photo in Figure 33(a).) The Volt dataset is an industrial CT scan of a 440 voltage electrical connector. is an industrial CT scan of a solid aluminum shape used to calibrate measurements from CT scans. CMM stands for "coordinate measuring machine". Since the CT scanner provides only scalar values for each object, we used Religrad to construct gradients at the grid vertices.

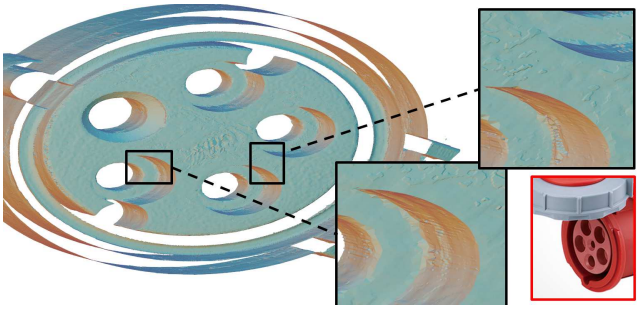


Figure 35: SHREC with Religrad gradients computed from part of the VOLT data. Magnified regions show "sharp" reconstructed edges. In red picture of the original item.

Dataset sizes, spacing, isovalues and average isosurface sizes are presented in Table 3. The full Motorcycle Engine dataset has very large dimensions so we only report on a small $200 \times 129 \times 62$ corner of that dataset depicted in Figures 33 and 34. Note the non-uniform spacing in the Motorcycle Engine and CMM datasets.

Figures 33 and 34 show a small corner of the motorcycle engine cylinders and the results of reconstruction of that corner. The magnified regions (with black border) in Figure 34 show that the sharp edges are well reconstructed. In yellow border boxes, we see magnified parts of the reconstructed mesh along with the sharp and non-sharp edges generated by SHREC.

Figure 35 shows part of the 440 voltage connector (in red) and the SHREC reconstruction of the Volt dataset. To aid in visualization of the sharp features, the figure displays a cropped image of the the reconstructed isosurface. Once again we see that SHREC is able to reconstruct the sharp (flange-like) curves accurately.

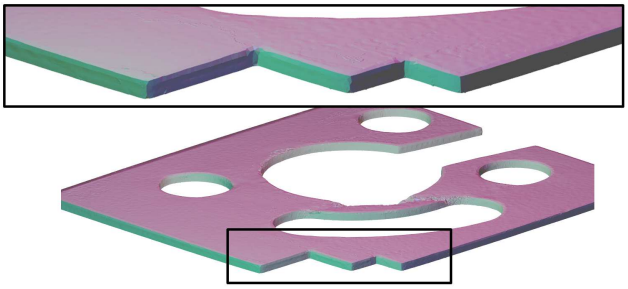
Figure 36 shows the SHREC reconstruction of the CMM dataset. Again, the reconstructed isosurface is cropped to better display the sharp features. Figure 36(b) shows a single slice of the CT scan, with scalar values mapped to the "heat" color map.

SHREC does a good job of reproducing the 0 and 1 dimensional features from each of these data sets and in producing meshes which accurately reflect those features.

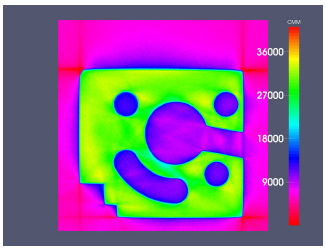
16 Timings

Table 4 gives the running time of the various algorithms on a TwoCubes dataset. The running time was measured on a Windows PC with an Intel Core i7-4770 processor (3.4 GHz) and 16 GB RAM.

Inputs to Marching Cubes, EMCpoly, MergeSharp and SHREC were grids of size $150 \times 150 \times 150$. PolyMender constructed an oct tree with depth 7, with a full resolution grid of size $128 \times 128 \times$



(a)



(b)

Figure 36: (a) SHREC with Religrad gradients computed from part of the CMM dataset. Magnified regions show "sharp" reconstructed edges. (b) Single slice of the CT scan, with scalar values mapped to the "heat" color map.

Software	Num Iso Tri	Time
Marching Cubes	41K	0.4 sec
EMCpoly	46K	0.8 sec
PolyMender	85K	0.7 sec
MergeSharp	35K	0.6 sec
SHREC grad3	34K	1.0 sec
SHREC grad9	34K	1.4 sec
Religrad	34K	30 sec
SingularCocone	125K	1.8 sec
FeatureRecon	125K	344 sec

Table 4: Timings. SHREC grad3 and SHREC grad9 is SHREC using gradients from $3 \times 3 \times 3$ and $9 \times 9 \times 9$ subgrids, respectively, around selected vertices.

128. Inputs to SingularCocone and FeatureRecon had 60K sample points.

The point cloud reconstruction software SingularCocone and FeatureRecon are very different from the isosurface reconstruction software, so one should be extremely careful in comparing its timings to the others. SingularCocone receives a set of points on 0 and 1-dimensional features as part of its input, while EMCpoly, PolyMender, MergeSharp and SHREC must spend time computing such points from gradients or surface normals. On the other hand, FeatureRecon receives a set of sample points without any gradient or surface normal information, while EMCpoly, PolyMender, MergeSharp and SHREC receive gradient or surface normals as part of their input. Computing points on 0 and 1-dimensional features is much easier if gradient or surface normals are provided.

EMCpoly, PolyMender, MergeSharp and SHREC take comparable times. SHREC takes the longest and the time increases if gradients are selected from larger subgrids around each cube. The extra time taken by SHREC is spent selecting gradients around each cube. The time for merging cubes in MergeSharp and SHREC is a small part of the total, since merging is only performed around selected grid cubes covering sharp features.

If a gradient grid file is not available, then it must be constructed from the scalar grid. The Religrad time (30 second) to construct the gradient grid should be added to the SHREC time. Note that EMCpoly, PolyMender and MergeSharp would also require Religrad or some similar program to construct a gradient grid or set of surface normals, so the programs still take comparable times. FeatureRecon takes considerably longer than Religrad but input to FeatureRecon is set of surface sample points, not a scalar grid.

17 Software and Datasets

Software and datasets used in this paper can be downloaded from: web.cse.ohio-state.edu/research/graphics/isotable.

In particular, the web site contains source code for SHREC, Religrad, MergeSharp and EMCpoly. The web site also contains programs to generate regular grid samplings of scalar and gradient fields (ijkgenscalar), generate polygonal meshes of annuli, flanges, two cubes, cones, frustra, smooth tip cones and cannons (ijkgenmesh), compute angle distances (angle_dist), find sharp edges (findsharp) and count the vertex degree in the graph formed by the sharp edges (countdegree). Finally, the web site contains the Cannon and Cone scalar and gradient datasets and some of the Flange and TwoCubes scalar and gradient datasets. (Because of restricted space, we were not able to include all 39 Flange datasets and 34 TwoCube datasets in the web site.)

18 Conclusion and Future Work

SHREC produces far fewer polygons with normal errors than any other software we tested, but it still occasionally produces such errors. Because of the regular structure of the regular grid, we were hoping, but unable, to eliminate all such errors and even to prove "correctness" of the reconstruction under appropriate conditions. We still believe that some algorithm along the lines of SHREC should be able to construct isosurfaces with provable guarantees on the reconstruction of sharp features and surface normal directions.

One of the problems with the work on sharp feature reconstruction is the lack of quantitative measures of the accuracy of the feature reconstruction. We hope that the extensive quantitative comparisons in this paper will set a precedent for such measurements in future work on sharp feature reconstruction.

We presented the angle distance as a measurement of the difference between the normals in two surfaces. The angle distance does not obey the triangle inequality and is not a metric. It would be nice to have some measurement similar to angle distance which measured the difference between surface normals and is a metric.

19 Acknowledgments

We would like to thank Craig Leffel, Brent Obermiller and Honda of America Manufacturing for providing the CMM and MotorCycle Engine datasets. We would like also like to thank Christoph Heinzl from the University of Applied Sciences Upper Austria for providing the industrial CT volt dataset.

References

[1] A.Bhattacharya, R.Wenger. Constructing isosurfaces with sharp edges and corners using cube merging. *Computer Graphics Forum*, 32:11–20, 2013.

[2] K. Ashida and N. I. Badler. Feature preserving manifold mesh from an octree. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, pages 292–297. ACM Press, 2003.

[3] N. Aspert, D. Santa Cruz, and T. Ebrahimi. Mesh: measuring errors between surfaces using the hausdorff distance. In *ICME (1)*, pages 705–708, 2002.

[4] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or. $\updownarrow_1$-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG)*, 29(5):135, 2010.

[5] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):349–359, 1999.

[6] A. Bhattacharya and R. Wenger. Experimental results on mergesharp. Technical Report OSU-CISRC-3-15-TR05, Dept. of Computer Science and Engineering, The Ohio State University, 2013.

[7] A. Bhattacharya and R. Wenger. Computing reliable gradients from scalar data. Technical Report OSU-CISRC-6-15-TR11, Dept. of Computer Science and Engineering, The Ohio State University, 2015.

[8] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1096–1105, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[9] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[10] J. Daniels, L. K. Ha, T. Ochotta, and C. T. Silva. Robust smooth feature extraction from point clouds. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pages 123–136. IEEE, 2007.

[11] T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. geometric combinatorics. *Publ. Inst. Math. (Beograd) (N.S.)*, 66:23–45, 1999.

[12] T. K. Dey, X. Ge, Q. Que, I. Safa, L. Wang, and Y. Wang. Feature-preserving reconstruction of singular surfaces. *Comp. Graph. Forum*, 31(5):1787–1796, Aug. 2012.

[13] T. K. Dey and L. Wang. Voronoi-based feature curves extraction for sampled singular surfaces. *Computers & Graphics*, 37(6):659 – 668, 2013. Shape Modeling International (SMI) Conference 2013.

[14] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24:544–552, July 2005.

[15] S. F. F. Gibson. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI 1998*, pages 888–898. Springer-Verlag, 1998.

[16] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 23–30, 1998.

[17] A. Greß and R. Klein. Efficient representation and extraction of 2-manifold isosurfaces using kd-trees. *Graphical Models*, 66(6):370–397, 2004.

[18] C. Ho, F. Wu, B. Chen, and M. Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum*, 24:2005, 2005.

[19] T. Ju. Robust repair of polygonal models. *ACM Transactions on Graphics*, 23(3):888–895, Aug. 2004.

[20] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[21] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[22] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001*, pages 57–66. ACM Press, 2001.

[23] P. Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000*, pages 259–262. ACM Press/Addison-Wesley Publishing Co., 2000.

[24] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–170, 1987.

[25] J. Manson and S. Schaefer. Isosurfaces over simplicial partitions of multiresolution grids. *Computer Graphics Forum*, 29(2):377–385, 2010.

[26] G. M. Nielson. Dual Marching Cubes. In *Proceedings of IEEE Visualization 2004*, pages 489–496. IEEE Computer Society, 2004.

[27] T. Ochotta and D. Saupe. Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields. Technical report, University of Konstanz, 2004.

[28] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[29] N. Salman, M. Yvinec, and Q. Merigot. Feature preserving mesh generation from 3d point clouds. *Computer Graphics Forum*, 29(5):1623–1632, 2010.

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

[31] S. Schaefer and J. Warren. Dual contouring: The secret sauce. Technical Report TR 02-408, Dept. of Computer Science, Rice University, 2002.

[32] S. Schaefer and J. Warren. Dual marching cubes: Primal contouring of dual grids. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 70–76. IEEE Computer Society, 2004.

[33] J. Schreiner, C. E. Scheidegger, S. Fleishman, and C. T. Silva. Direct (re) meshing for efficient surface processing. *Computer graphics forum*, 25(3):527–536, 2006.

[34] G. Varadhan, S. Krishnan, Y. J. Kim, and D. Manocha. Feature-sensitive subdivision and isosurface reconstruction. In *Proceedings of IEEE Visualization 2003*, pages 99–106. IEEE Computer Society, 2003.

[35] J. Wang, Z. Yu, W. Zhu, and J. Cao. Feature-preserving surface reconstruction from unoriented, noisy point data. *Computer Graphics Forum*, 32(1):164–176, 2013.

[36] R. Wenger. *Isosurfaces: geometry, topology, and algorithms*. CRC Press, 2013.

[37] N. Zhang, W. Hong, and A. Kaufman. Dual contouring with topology-preserving simplification using enhanced cell representation. In *Proceedings of IEEE Visualization 2004*, pages 505–512. IEEE Computer Society, 2004.

A Closest Point under the L_∞ Distance

Let $p = (p_x, p_y, p_z)$ be a point and let L be a line in $\mathbb{R}^3$. We wish to find the point on L closest to p under the L_∞ distance.

Parameterize line L by $tu + q$ where $u = (u_x, u_y, u_z)$ and $q = (q_x, q_y, q_z)$. Let q^* be the point of L closest to p under the L_∞ distance. Let δ be the L_∞ distance from q^* to p and let $\mathbf{c}$ be a $2\delta \times 2\delta \times 2\delta$ cube centered at p. Line L is tangent to $\mathbf{c}$ at point q^*.

Let $\pi_i(p)$, $\pi_i(L)$, $\pi_i(q^*)$ and $\pi_i(\mathbf{c})$ be the projection of p, L, q^* and $\mathbf{c}$ onto a plane orthogonal to axis i. For some axis i, projected line $\pi_i(L)$ is tangent to square $\pi(\mathbf{c})$ at point $\pi_i(q^*)$. For this axis, $\pi_i(q^*)$ is the point of $\pi_i(L)$ closest to $\pi_i(p)$ under the L_∞ distance. Therefore, if $tu + q$ is the point on L closest to p under L_∞, then $t\pi_i(u) + \pi_i(q)$ is the point on $\pi_i(L)$ closest to $\pi_i(p)$ under L_∞ for some axis i. Thus, we project p and L onto the three planes orthogonal to the three axes and find t_i such that $t_i\pi_i(u) + \pi_i(q)$ is closest to $\pi_i(p)$.

Consider a projection, π_{xy}, of point p and line L onto the xy plane. The projected point $\pi_{xy}(p)$ has coordinates (p_x, p_y) and the projected line $\pi_{xy}(L)$ is parameterized by $t(u_x, u_y) + (q_x, q_y)$. The point on $\pi_{xy}(L)$ which is closest to $\pi_{xy}(p)$ under the L_∞ distance satisfies the equation:

$$|tu_x + q_x - p_x| = |tu_y + q_y - p_y|.$$

Equivalently,

$$t = \frac{(q_y - p_y) - (q_x - p_x)}{u_x - u_y}, \text{ or}$$
$$t = \frac{(q_y - p_y) + (q_x - p_x)}{u_x + u_y}.$$

Solving for t in each direction, computing the L_∞ distance from $tu+q$ to p, and taking the minimum, identifies the point $tu+q$ which is closest to p under the L_∞ distance.