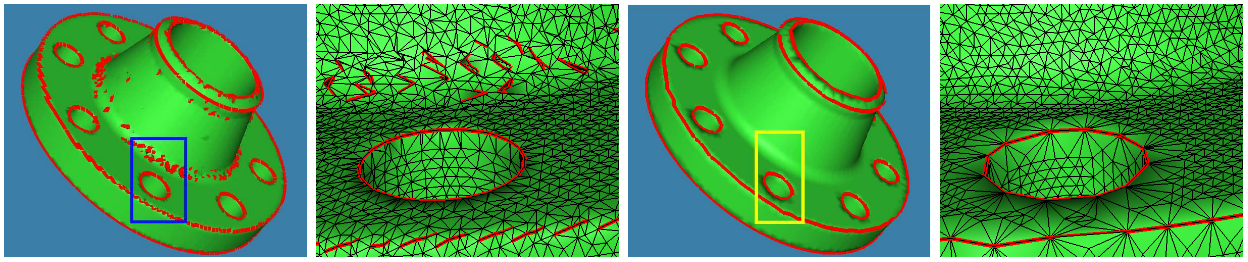
EUROGRAPHICS 2013 / B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

Constructing Isosurfaces with Sharp Edges and Corners using Cube Merging

A. Bhattacharya and R. Wenger[†]



(a) Polymender isosurface. (b) Close-up of blue square. (c) MergeSharp isosurface. (d) Close-up of yellow square.

Figure 1: Isosurfaces from the weld dataset with 'sharp ' mesh edges (large dihedral angles) marked in red. 1a), 1b) Polymender generates disconnected sharp mesh edges representing a sharp edge of the object. It also generates mesh edges with large dihedral angle in smooth regions of the surface. 1c), 1d) MergeSharp correctly generates a connected curve of sharp mesh edges to represent a sharp edge of the object. It does not generate mesh edges with large dihedral angles in the smooth region of the surface.

Abstract

A number of papers present algorithms to construct isosurfaces with sharp edges and corners from hermite data, i.e. the exact surface normals at the exact intersection of the surface and grid edges. We discuss some fundamental problems with the previous algorithms and describe a new approach, based on merging grid cubes near sharp edges, that produces significantly better results. Our algorithm requires only gradients at the grid vertices, not at each surface-edge intersection point. We also give a method for measuring the correctness of the resulting sharp edges and corners in the isosurface.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—

1. Introduction

Let $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ be a scalar field. Various algorithms have been proposed for constructing isosurface approximations to a surface $f^{-1}(\sigma)$ with sharp edges and corners from distance fields [GK04, HWCO05, KBSS01, ZHK04] or from an explicit definition of f [AB03, JLSW02, HWCO05, VKKM03, SW04, MS10]. All these algorithms rely on the ability to compute the exact intersection point of the isosurface and each grid edge and an exact surface normal at that intersection point. Ju et al. [JLSW02] coined the term "hermite" data to describe such inputs.

The algorithms listed above have a number of drawbacks. First, some of these algorithms have fundamental problems when surfaces or portions of surfaces are not oriented along grid axes. They produce isosurface meshes which poorly model the sharp edges or corners (Figure 2.a) or contain degenerate or overlapping triangles (Figure 2.b).

Second, the algorithms listed above rely upon precise cal-

[†] Computer Science and Engineering Department, The Ohio State University, E-mails: bhattaca@cse.ohio-state.edu, wenger@cse.ohio-state.edu.

culation of both the intersection point and the normal. They have little tolerance for approximation errors in those values. Those algorithms which compute multiple intersections or increase the grid resolution create small thin polygons near sharp features. Such polygons are very sensitive to approximation error. Third, all these algorithms are restricted to hermite data or variations such as a signed distance field which permit the computation of surface normals along grid edges.

In [CDR07], Cheng et al. describe a method for meshing piecewise smooth complexes using protecting balls around the sharp or boundary edges of those complexes. We apply this idea to isosurface reconstruction by merging grid cubes around sharp edges and corners, creating regions which act like protecting balls. A single isosurface vertex is placed in each such region.

By placing a single isosurface vertex near the center of each region, we guarantee that the isosurface vertices on sharp edges and corners are well separated from any other isosurface vertices. Separating these isosurface vertices avoids the creation of degenerate quadrilaterals or the incorrect ordering of isosurface vertices along sharp edges. It also avoids the creation of notches along sharp edges. A notch is a discontinuity in the tangent of the sharp edge. (See Figure 2a.)

The separation of isosurface vertices on sharp edges makes the sharp edge much less sensitive to changes in vertex location. Our algorithm is tolerant of small approximation errors in the intersection locations or the surface normal coordinates.

Because our algorithm is tolerant of approximation errors in isosurface vertex location, we no longer require hermite data as input. Instead we can use gradient grid data, regular grids with scalar values and gradients at each of the grid vertices. We compute isosurface vertex locations directly from this data, using gradients from neighboring cubes to increase reliability. As shown in Section 9, our algorithm is tolerant of noise in this gradient data.

Previous papers on reconstructing isosurfaces with sharp edges and corners lacked any quantitative measure of the quality of the reconstruction. The lack of such measure makes it difficult to evaluate the claims of these papers or compare the results of their algorithms in any systematic way. In Section 8, we propose a simple method for measuring the quality of the reconstructed sharp edges and corners. Our evaluation method helps us quickly find errors in the sharp edge and corner reconstructions and allows us to evaluate our algorithm on numerous test datasets without requiring visual inspection of the results of each test. It also permits us to test and compare parameter changes to our algorithm and to compare our algorithm with Polymender [Ju04].

Our paper contains three major contributions:

1. We present a new algorithm based on cube merging for constructing isosurfaces with sharp edges and corners. Our algorithm solves some fundamental problems with previous techniques and is significantly more robust.
2. We show how gradient grid data, not just hermite data, can be used to calculate the locations of isosurface vertices on sharp edges and corners.
3. We present a simple method for evaluating the quality of our reconstruction of sharp edges and corners, and evaluate our algorithm using that method.

2. Definitions

A scalar grid vertex is *negative* if its scalar values is less than the isovalue. A grid vertex is *positive* if its scalar values is greater than or equal to the isovalue.

A grid edge is *bipolar* if one endpoint is negative and one endpoint is positive.

A grid cube $\mathbf{c}$ is a *vertex neighbor* of grid cube $\mathbf{c}'$ if $\mathbf{c}$ shares a vertex with $\mathbf{c}'$.

3. Related Work

The Marching Cubes Algorithm [LC87] by Lorensen and Cline places all the isosurface vertices on grid edges. Because it does not place any vertices in cube interiors, it cannot represent sharp edges or corners.

Instead of adding isosurface vertices on grid edges, Gibson's dual contouring algorithm [Gib98a, Gib98b] places isosurface vertices inside grid cubes intersected by the isosurface. Later, Nielson [Nie04] described a dual contouring algorithm, Dual Marching Cubes, which sometimes adds multiple isosurface vertices inside a grid cube.

Kobbelt et al. [KBSS01] published the first algorithm to construct isosurfaces with sharp features. Their algorithm constructs a parametric representation of an implicit surface with sharp features from a grid of directed distances to that surface. Using linear interpolation, the algorithm computes a set of isosurface vertices on grid edges. It also computes surface normals at each of these isosurface vertices. Grid cubes with widely varying surface normals are identified as containing sharp features. If a grid cube does not have sharp features, an isosurface patch is retrieved from a lookup table as in Marching Cubes. If a grid cube has sharp features, then an additional isosurface vertex is added to the interior of the grid cube and connected to the isosurface vertices on the cube edges. The new isosurface vertex is positioned to minimize its least squares distance to tangent planes of the neighboring isosurface vertices. The final step applies edge flipping to connect vertices on sharp features in adjacent cubes.

Ju et al. [JLSW02,SW02] gave an alternative approach using dual contouring to construct parametric representations of implicit surfaces with sharp features. Input to their algorithm is hermite data instead of directed distances but the difference is minimal. Hermite data contains the exact intersection points of a surface with a regular grid and the exact normals. These values are easily computed from implicit surface representations.

The algorithm by Ju et al. retrieves the intersection points of grid edges and the implicit surface from the hermite data along with the normals at each intersection point. The normals define tangent planes at each intersection point. As in [KBSS01], the algorithm positions the isosurface vertex within a grid cube to minimize the least squares distance to tangent planes.

Algorithms by Varadhan et al. [VKKM03] and Zhang et al. [ZHK04] extended the dual contouring algorithm of Ju et al. by modeling multiple intersections of an isosurface and a grid edge and adding more than one isosurface vertex per grid cube. These algorithms can model thinner features than the original dual contouring algorithm.

Algorithms by Ho et al. [HWCO05] and Ashida and Badler [AB03] approximate the intersection of a surface and each grid cube boundary by a polygonal curve. They connect the curve to a single isosurface vertex in the interior of the cube. Sharp features are represented by appropriate positioning of the isosurface vertex and the curve vertices.

Schaefer and Warren [SW04] construct a dual grid whose vertices and edges are on sharp isosurface features. They applied Marching Cubes to the dual grid to extract the isosurface. Manson and Schaefer [MS10] tetrahedralize the regular grid and place the tetrahedra vertices on sharp isosurface features. They apply Marching Cubes to the tetrahedralization to extract the isosurface.

Except for the original dual contouring algorithm by Gibson and Nielson's Dual Marching Cubes, all the dual contouring algorithms listed above support multiresolution isosurface extraction. Other multi-resolution dual contouring algorithms are given in [SJW07] and [GK04]. Techniques for creating intersection free dual contouring isosurfaces are in [JU06] and [Wan11].

Garland and Heckbert [GH97] represented the least squares distance to a set of tangent planes by a 4x4 matrix which they called the quadric error measure (QEM). The matrix size is independent of the number of tangent planes. Lindstrom [Lin00] used the quadric error measure to compute the point which minimizes the least squares distance to the represented set of tangent planes. When the tangent planes define an edge or a smooth portion of the isosurface, Lindstrom's algorithm selects the point closest to the cube center. The feature sensitive dual contouring algorithms all use some variation of the quadric error measure to compute isosurface vertex locations.

Cheng et al. [CDR07] described an algorithm to mesh piecewise smooth complexes using weighted Delaunay triangulations. They choose points along the boundaries of each smooth piece and construct protecting balls around each such point by assigning a weight to each point. They then choose sample points from the smooth portion of the surface outside of the protecting balls and returned the weighted Delaunay triangulation of the points. Further descriptions can be found in [CDS13].

Salman et al. [SYM10] and Dey et al. [DGQ*12] use the protecting balls from [CDR07] to reconstruct surfaces with sharp features from point cloud data. Both papers identify sharp features and protect them with balls. They then reconstruct the surface using the protecting balls.

4. Computing Vertex Locations

As noted in the previous section, hermite data determines tangent planes to the isosurface, one at each intersection of the grid edge and the isosurface. For a grid cube **c**, the k tangent planes on its edges give a set of k equations

$$Mx = b$$

where M is a $k \times 3$ matrix and x and b are column vectors of length k. In general, this system is over-determined so we wish to find the least squares solution. The least squares solution is the solution to

$$M^T Mx = M^T b.$$

The 3×3 matrix $A = M^T M$ and the column vector $b' = M^T b$ gives the quadric error measure.

The singular valued decomposition (SVD) of A is $A = U\Sigma V$ where

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

σ_1, σ_2, and σ_3 are the singular values of A. If all three singular values of A are large, then **c** contains a sharp corner. If two singular values are large, then **c** contains a sharp edge. Otherwise, cube **c** does not contain a sharp feature.

$$\sigma'_i = \begin{cases} \sigma_i & \text{if } \sigma_i/\sigma^* > \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

where σ^* is the largest singular value and ε is a threshold parameter. Let $A' = U\Sigma' V^T$ where Σ' is the diagonal matrix with diagonal entries $(\sigma'_1, \sigma'_2, \sigma'_3)$.

When A has three large singular values, $A' = A$ and there is a single point x such that $A'x = b$. When A has two large singular values, $\{x : A'x = b'\}$ is a line. When A has one large singular value, $\{x : A'x = b'\}$ is a plane.

Let

$$\sigma^+_i = \begin{cases} 1/\sigma'_i & \text{if } \sigma'_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Let Σ^+ be the diagonal matrix with diagonal entries

$(\sigma_1^+, \sigma_2^+, \sigma_3^+)$. As in [Lin00], compute:

$$x' = q_\mathbf{c} + V\Sigma^+ U^T (b' - Aq_\mathbf{c}). \qquad (1)$$

When A has three large singular values, x' is the point solving $Ax = b$. When A has two large singular values, x' is the point closest to $q_\mathbf{c}$ on the line $A'x = b$. When A has only one large singular value, x' is a closest to $q_\mathbf{c}$ on the plane $A'x = b$. Lindstrom uses the center of grid cube $\mathbf{c}$ as the point $q_\mathbf{c}$.

As described above, matrices A and b' are determined by tangent planes. However, they can also be determined using gradients. Let g_i and s_i be the gradient and scalar value, respectively, at point p_i. Let σ be the isovalue. The set $\{x : g_i \cdot (x - p_i) + s_i = \sigma\}$ is a plane in 3D. Equivalently, this plane is $\{x : g_i \cdot x = \sigma - (g_i \cdot p_i + s_i)$. Let g_i be the rows of M be $g_i/|g_i|$ and the elements of b be $(\sigma - (g_i \cdot p_i + s_i))/|g_i|$. (We divide by $|g_i|$ so that all normal directions have equal weight.) Compute $A = M^T M$ and $b' = M^T b$ and solve as above. This formulation allows us to compute sharp isosurface vertex locations directly from gradients, without first transforming the gradients to hermite data.

Instead setting $q_\mathbf{c}$ in Equation 1 to be the center of cube $\mathbf{c}$, Schaefer and Warren [SW02] propose setting $q_\mathbf{c}$ to the centroid of the intersections of the edges of $\mathbf{c}$ and the isosurface. For reasons discussed in the next section, this choice of $q_\mathbf{c}$ improves the reconstruction results.

Hermite data gives the locations of the intersections of the edges $\mathbf{c}$ and the isosurface. If the input is a gradient grid, then these intersections must be computed from the input. A simple, but inaccurate approach, is to use linear interpolation to compute these intersections points. A more accurate approach is to use the gradients at the edge endpoints to determine the intersection point.

5. Problems with Vertex Locations

The Extended Marching Cubes algorithm by Kobbelt et al. [KBSS01] and the dual contouring algorithm by Ju et al. [JLSW02, SW02] compute an isosurface vertex for each grid cube using the quadric error measure (QEM). When the surface is relatively smooth, the isosurface vertex lies within the grid cube. However, if the surface has a sharp feature, the vertex location computed using QEM may lie outside the grid cube. Should the isosurface vertex be placed at the location outside the grid cube?

The problem of isosurface vertex locations lying outside the grid cube was noted by Schaefer and Warren in [SW02]. One reason an algorithm may compute a location outside a grid cube is that it chooses the wrong location on a sharp edge. Schaefer and Warren used the centroid of the intersections of the cube edge and the isosurface for the point p in Equation 1 to make it more likely that a point on the intersection of the sharp edge and the grid cube is chosen. However, what happens if the sharp edge or sharp corner does not intersect the grid cube? In that case, Schaefer and Warren's algorithm will still return a location outside the grid cube.

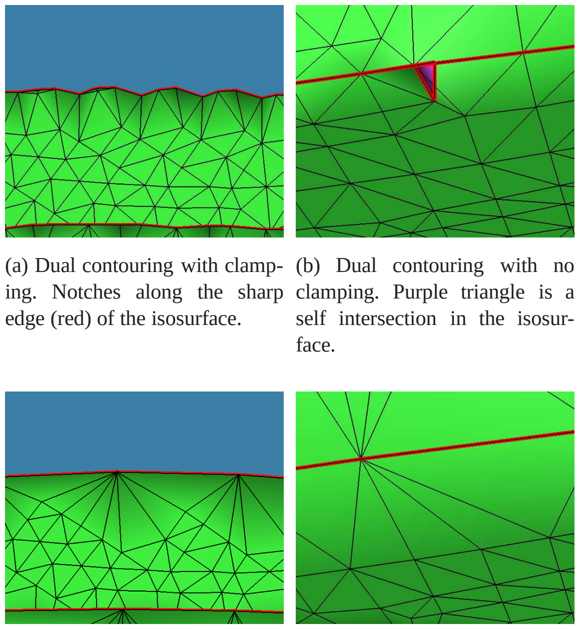

(a) Dual contouring with clamping. Notches along the sharp edge (red) of the isosurface.
(b) Dual contouring with no clamping. Purple triangle is a self intersection in the isosurface.
(c) MergeSharp of the region in 2a shows no notches.
(d) MergeSharp of the region in 2b shows no intersections.

Figure 2: Isosurface errors produced by dual contouring and corresponding MergeSharp results. Mesh edges with large dihedral angle are colored red.

Consider a grid cube $\mathbf{c}$ which generates a vertex location p on a sharp edge or corner which does not intersect $\mathbf{c}$. Let $\mathbf{c}' \neq \mathbf{c}$ be the grid cube containing p. If cube $\mathbf{c}'$ has a bipolar edge, then it also generates an isosurface vertex v'. Placing v in $\mathbf{c}'$ may create degenerate mesh triangles or it may create overlapping triangles as the mesh folds back on itself. (See Figure 2b.) On the other hand, if $\mathbf{c}'$ does not have any bipolar edge, then it does not generate an isosurface vertex. Clamping v to lie inside $\mathbf{c}$ will create notches in the isosurface or cut off the corner. (See Figure 2a.)

One plausible approach might be to clamp p to cube $\mathbf{c}$ only if $\mathbf{c}'$ contains a bipolar edge. As shown in Section 9, this approach also produces numerous errors.

For surfaces whose sharp edges are parallel to the xy, yz, or xz planes, grid cubes which intersect the surface corners or edges will almost always have bipolar edges. Thus clamping isosurface vertex v will not create notches along the sharp edges and will not cut off corners. On the other hand, rotating such surfaces by any significant angle with respect to the grid creates many sharp corners or edges which intersect cubes with no bipolar edges.

```
    MERGE(Grid,$C_{\text{selected}}$)
1   foreach cube c do
2       Compute the centroid $q_{\mathbf{c}}$ of the intersections of the
        edges of c and the isosurface;
3       Compute vertex location $p_{\mathbf{c}}$ using Equation 1;
4   end
5   $C_{\text{corner}} \leftarrow$ cubes c where $p_{\mathbf{c}}$ is on a sharp corner;
6   $C_{\text{edge}} \leftarrow$ cubes c where $p_{\mathbf{c}}$ is on a sharp edge;
7   Sort $C_{\text{corner}}$ and $C_{\text{edge}}$ in increasing order by
    $|p_{\mathbf{c}} - q_{\mathbf{c}}|_\infty$ (the Linf distance);
8   Mark all cubes as Uncovered;
9   MERGECUBES($C_{\text{corner}}$,$C_{\text{selected}}$);
10  MERGECUBES($C_{\text{edge}}$,$C_{\text{selected}}$);
11  foreach cube $\mathbf{c} \in C_{\text{corner}} \cup C_{\text{edge}}$ do
12      if (c is not selected) and (c is not Covered) then
        $p_{\mathbf{c}} \leftarrow q_{\mathbf{c}}$;
13  end
```

Algorithm 1: Algorithm MERGE.

```
    MERGECUBES($C, C_{\text{selected}}$)
1   foreach cube $\mathbf{c} \in C$ do
2       if (cube c is Uncovered) and ($|p_{\mathbf{c}} - q_{\mathbf{c}}|_\infty \leq \gamma$)
        then
3           if (NOLARGEANGLETRI(c,$C_{\text{selected}}$)) then
4               Add c to $C_{\text{selected}}$;
5               foreach vertex neighbor $\mathbf{c}'$ of c do
6                   if cube $\mathbf{c}'$ is Uncovered then
7                       $\mathbf{c}'$.MergeWith $\leftarrow$ c;
8                       Mark c as Covered;
9                   end
10              end
11          end
12      end
13  end
```

Algorithm 2: Algorithm MERGECUBES.

6. Merging Grid Cubes

To address the problems discussed in the previous section, we use a technique similar to the protecting balls from [CDR07]. We identify the grid cubes whose isosurface vertices lie on sharp edges or corners, and select a subset such that no two selected cubes are vertex neighbors. (A grid cube **c** is a *vertex neighbor* of grid cube $\mathbf{c}'$ if **c** shares a vertex with $\mathbf{c}'$.) We merge each selected cube **c** with its vertex neighbors. We generate a single vertex for the merged region.

Let C_{corner} be the grid cubes whose isosurface vertices lie on sharp corners. Let C_{edge} be the grid cubes whose isosurface vertices lie on sharp edges. We start by selecting cubes from C_{corner}.

To select the cubes from C_{corner}, we compute the vertex location $p_{\mathbf{c}}$ using Equation 1. We use the centroid of the intersections of the edges of **c** and the isosurface as point $q_{\mathbf{c}}$. We process the cubes of C_{corner} in increasing order of the L_∞ distance between $p_{\mathbf{c}}$ and $q_{\mathbf{c}}$. This causes cubes which are "closer" to the corners to be processed first.

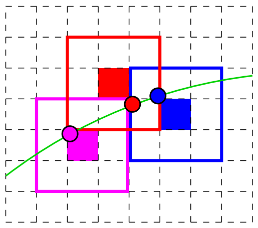


Figure 3: 2D illustration of cube stack (blue, magenta, red) which are close enough together to form a triangle yet far enough apart so that no $3\times3\times3$ cube region covers the other two cubes. Sharp edge is represented by the green curve. Each cube generates a vertex location (with the same color as the cube) on the sharp edge. The triangle formed by the three vertices is almost degenerate.

As we select cubes, we merge their vertex neighbors with them. A cube which has been merged with a selected cube is labeled "covered". All cubes are initially uncovered. We select only uncovered cubes. When we select cube **c**, we merge the uncovered vertex neighbors of **c** with **c**. Those vertex neighbors become covered. Thus no two selected cubes can be vertex neighbors. If the L_∞ distance between $p_{\mathbf{c}}$ and $q_{\mathbf{c}}$ is greater than some threshold, we do not select **c**. We repeat this procedure to select cubes from C_{edge}, processing them in increasing order by the L_∞ distance between $p_{\mathbf{c}}$ and $q_{\mathbf{c}}$.

The procedure, as just described, may still create many degenerate or near-degenerate triangles. Three cubes near a sharp edge could be close enough together to form a triangle yet far enough apart so that no cube region covers the other two cubes. (See Figure 3.) When the vertices generated by the cube stack are mapped to the sharp edge, the angle at the middle vertex becomes 180° or near 180°. To avoid such problem triangles, we do not select a cube which forms a large angle triangle with already selected cubes. (See Algorithm NOLARGEANGLETRI.)

Checking each pair of cubes in C_{selected} can be a time consuming operation. We use a simple nearest neighbor data structure based on a low resolution subgrid to quickly identify points in C_{selected} which are near **c**.

7. Isosurface Construction

The algorithm MERGE (Algorithm 1) constructs regions with irregular shapes around selected vertices. Instead of extracting the dual isosurface from those regions, we extract the dual isosurface from the full grid and then merge isosurface vertices which are from cubes in the same region. This

```
   NOLARGEANGLETRI($\mathbf{c}, C_{\text{selected}}$)
1  for each pair $\mathbf{c}', \mathbf{c}'' \in C_{\text{selected}}$ near $\mathbf{c}$ do
2      $R' \leftarrow$ Reg3x3x3($\mathbf{c}$) $\cap$ Reg3x3x3($\mathbf{c}$');
3      $R'' \leftarrow$ Reg3x3x3($\mathbf{c}$) $\cap$ Reg3x3x3($\mathbf{c}$");
4      $R''' \leftarrow$ Reg3x3x3($\mathbf{c}$') $\cap$ Reg3x3x3($\mathbf{c}$");
5      if ($R'$ has a bipolar edge and $R''$ has a bipolar edge
       and $R'''$ has a bipolar edge) then
6          $\alpha \leftarrow$ max angle of triangle $\Delta(p_{\mathbf{c}}, p_{\mathbf{c}'}, p_{\mathbf{c}''})$;
7          if ($\alpha \geq 140°$) then return (false);
8      end
9  end
10 return (true);
```

Algorithm 3: Algorithm NOLARGEANGLETRI.

is simpler than trying to extract the dual isosurface from the regions.

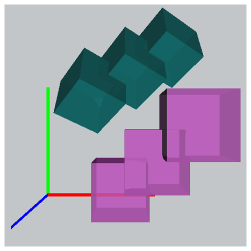
Our algorithm for isosurface construction has four steps. First, compute an isosurface vertex location for each grid cube with a bipolar edge. Second, select a set of isosurface vertices on sharp corners and edges and their surrounding regions. Third, apply Nielson's Dual Marching Cubes algorithm [Nie04] to construct the dual contouring isosurface from the full grid. Finally, for each merged region, merge isosurface vertices generated by the cubes in that region (Algorithm 2). Merging isosurface vertices transforms isosurface quadrilaterals to triangles and creates some degenerate quadrilaterals which are removed.

8. Measuring Correctness of Sharp Features

A procedure called FindSharp is used to compute and visualize sharp edges. FindSharp computes the dihedral angle between adjacent surface polygons. It reports any isosurface edge whose dihedral angle is greater than a threshold ($40°$) or which is incident on three or more isosurface polygons. We visualize such edges by drawing them in red, either with or without the underlying isosurface.

The set of isosurface edges reported by FindSharp is a graph embedded in $\mathbb{R}^3$. Procedure CountDegree measures the correctness of the sharp feature reconstruction by counting the number of vertices of each degree in this graph. The absolute difference between this and the expected number gives the number of errors for that degree. The total error is the sum of those errors.

For example, the flange isosurfaces (Figure 4b) have no corners, so the graph of its sharp edges should have no vertices of degree other than two. Any vertices of any other degree are errors. The isosurface of cube stack (Figure 4a) have 32 corners and saddles. The graph of its sharp edges should have exactly 32 degree 3 vertices. Any vertices of degree 1 or degree more than three are errors.


(a) Cube stack.

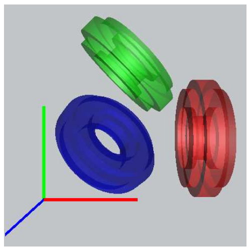

(b) Flange.

Figure 4: Elements from our evaluation datasets. The cube stacks and flanges are rotated at various angles to test the robustness of the algorithms. Note that each data set contains only one cube stack or flange, not two or three as shown here.

9. Experimental Results

9.1. Benchmark Datasets and Evaluation Interpretation

We developed two sets of data to test our algorithm. The cube stack datasets sample a scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ where $f(p)$ is the minimum of the L_∞ distance to three points, q_1, q_2 and q_3. Isosurfaces in the datasets are three (overlapping) cubes, whose centers are q_1, q_2 and q_3 (Figure 4a). To tilt the cubes, we used orthogonal frames other than the standard one given by the x, y, z axes, and computed the L_∞ metric in those other frames. The cube stack datasets test the performance of our algorithm on sharp corners or saddles.

The flange datasets sample a scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ where $f(p)$ is the combination of the distance $d_C(p)$ to a cylinder and the distance $d_P(p)$ to a plane orthogonal to the cylinder axis. The function f is defined as:

$$f(p) = \max(\min(d_C(p), d_P(p)), \max(d_C(p), d_P(p))/2).$$

Isosurfaces in these datasets are flanges with sharp concave and convex edges. (Figure 4b). The flange datasets test the performance of our algorithm on sharp edges.

We embedded each scalar field in a 100x100x100 grid at various angles to the grid axes. The grids contain both scalar values AND the gradients of the underlying scalar field at each grid vertex. With the different embeddings, we have a total of 100 datasets, which we test with six different isovalues for a total of 600 test cases. We applied FindSharp and CountDegree to the isosurfaces produced from each dataset and measured the total degree errors as described in the previous section. Isosurfaces that have poor representation of sharp features produce numerous graph vertices with degree one, three or higher. We counted the number of isosurface which had no errors, the number which have between 1 and 10 errors and the number with more than 10 errors. For instance, the second column of Table 1 shows that our algorithm produced 343 isosurfaces with no detected errors, 243 isosurfaces with 1 to 10 errors, and 14 isosurfaces with more than 10 errors. In contrast, the various versions of dual con-

		Dual Contouring		
Num Errors	Merge-Sharp	Clamp	No Clamp	Clamp Conflict
0	343	10	56	32
1-10	243	23	370	221
> 10	14	567	174	347

Table 1: Error counts for MergeSharp and for dual contouring. Error counts for dual contouring where isosurface vertex positions are clamped to their generating cube (Clamp), not clamped (No Clamp), or clamped if they initially lie in some cube which has a bipolar edge (Clamp Conflict).

touring with and without clamping produced fewer than 60 isosurface with no detected errors.

9.2. Evaluation

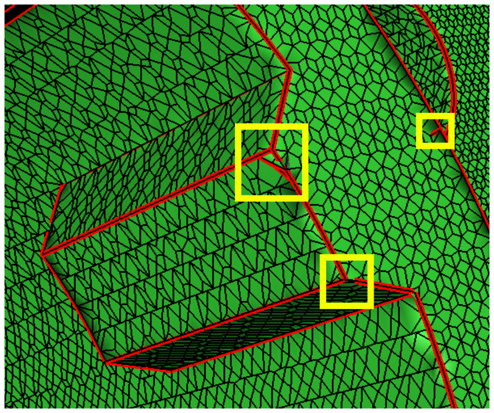
We first compared our algorithm based on merging cubes with an implementation of dual contouring with no cube merging. We compared it against versions of dual contouring where isosurface vertices are clamped to their generating grid cube, where isosurface vertices are not clamped to their generating grid cube, and where isosurface vertices are clamped only if they lie in some other grid cube $\mathbf{c}'$ which has a bipolar edge.

The flange and cube stack datasets contain scalar values and gradients at grid vertices. To make the dual contouring algorithms as similar as possible to [JLSW02], we converted the grid data to hermite data by computing the intersection of the isosurface and each grid edge using the gradients at the edge endpoints. We ran both our algorithm and the dual contouring on this hermite data.

The results are described in the Table 1. Our algorithm (MergeSharp) does significantly better than the various versions of dual contouring. Figure 2 contains sample problems region produced by dual contouring with conflict clamping and with no clamping and the MergeSharp isosurface for the corresponding regions.

The program Polymender [Ju04] produces and outputs hermite data representing the distance to an input polygonal model. We downloaded four triangle mesh datasets from grabcad.com/library/software/stl-for-3d. We rotated three of the datasets (brake, rotor, rotor arm) so that their sharp edges were not parallel to the *xy*, *yz* or *xz* planes. We used Polymender to produce hermite data for the datasets and compared the isosurfaces produced by Polymender and by our algorithm. We used an octree depth of 8 and a scale parameter of 0.8 for Polymender.

Figures 1 and 5 contain visual comparisons of outputs of Polymender and MergeSharp on the weld and rotor datasets. Results in Table 2 show that MergeSharp is far superior at constructing sharp isosurface edges which represent the


(a) Polymender isosurface.

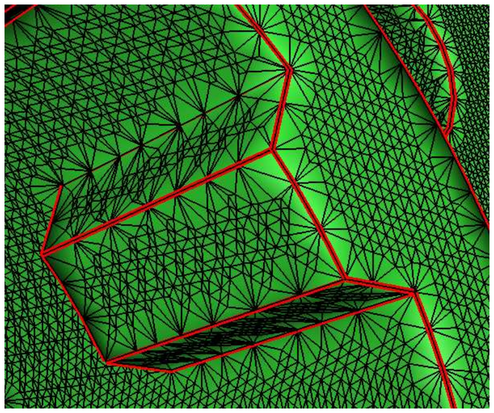

(b) MergeSharp isosurface.

Figure 5: Close-up isosurfaces on the rotor dataset. Edges with large dihedral angle are shown in red. (a) Yellow rectangles indicate some problematic regions in Polymender isosurface.

sharp edges in the original polygonal model. Polymender did well on the three original models of brake, rotor and rotor arm when no rotation was applied to those models.

An implementation by J. Manson of isosurfacing from simplicial decompositions [MS10] is available at josiahmanson.com/research. Figure 6 contains an isosurface of a non-axis aligned cylinder produced by Manson's program. The isosurface contains many small, thin triangles. The smooth part of the isosurface contains many (red) edges with large dihedral angles. There are also many errors around the sharp cylinder edge.

We applied FindSharp and CountDegree to the isosurface displayed in Figure 6. The graph of sharp edges of this isosurface has 381 vertices with degree other than two. It should have zero vertices with degree other than two. Because Manson's program does not produce hermite data or a scalar grid, we were unable to compare it directly with MergeSharp.

Dataset	Polymender				MergeSharp			
	Degree 1	Degree 3	Degree ≥ 4	Total $\neq$ 2	Degree 1	Degree 3	Degree ≥ 4	Total $\neq$ 2
Weld	437	429	176	1042	48	34	3	85
Brake	674	490	282	1446	7	35	3	45
Rotor arm	58	72	19	149	2	6	2	10
Rotor	647	485	188	1320	2	6	2	10

Table 2: Comparison of Polymender and MergeSharp on triangle mesh data sets.

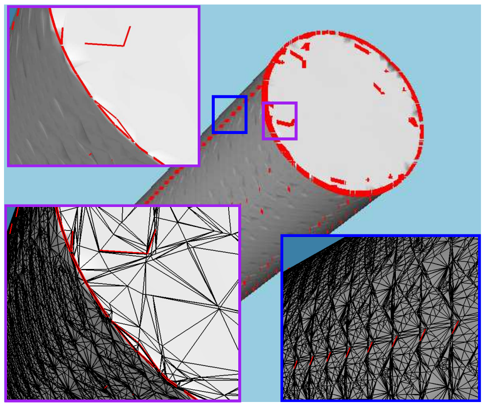


Figure 6: Results of Manson's implementation of [MS10] on a non-axis aligned cylinder.

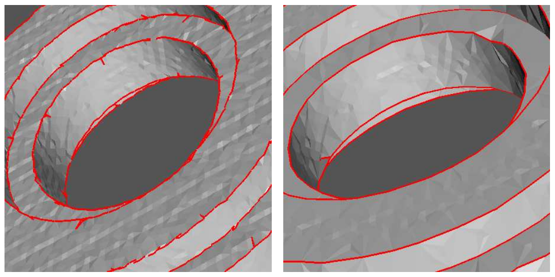
Num Errors	min large singular value		
	0.05	0.1	0.2
0	418	411	344
1-10	175	185	250
$\geq$ 10	7	4	6

Table 3: Comparing different thresholds for the singular value. Singular values below the threshold are set to 0 in computing vertex locations.

As in [SW02] we truncate the singular values based on the relative magnitude to the largest singular value. For all previous tables, we use 0.1 as the threshold. In Table 3 we show the results of running MergeSharp with other threshold values.

Finally, we claim that our algorithm was robust to noise. Figure 7 contains a visual comparison of the results of our algorithm and MergeSharp. Table 4 contains results comparing MergeSharp on perfect data and on data with noisy gradient directions. The results from dual contouring (Fig 7a) contain numerous errors, with almost all cases resulting in more than 10 errors.

More evaluation results are contained in the technical report [BW13].


(a) Dual contouring, no clamping.

(b) MergeSharp.

Figure 7: Noisy data where gradients were perturbed uniformly by an angle of 20 degrees. a) Isosurface from dual contouring with no clamping. b) MergeSharp isosurface.

Num Errors	perfect grad	10 degrees	20 degree
0	201	155	43
1-10	96	135	100
$\geq$ 10	3	10	157

Table 4: Effect of adding uniform noise to the gradients (300 test cases). Gradients were perturbed uniformly within the given angular bound.

9.3. Timings

We ran our experiments on a standard desktop with four giga-byte ram and with two cores Intel CPU. Table 5 gives execution times both for hermite data and gradient data. Figure 8 graphs execution time as a function of grid size.

Both Table 5 and Figure 8 show that the most time consuming step is computing vertex locations. This step is common to all algorithms which construct isosurfaces with sharp features. Vertex locations are computed for every cube intersected by the isosurface. Thus, the time for computing locations is proportional to the number of cubes intersected by the isosurface. The time to merge cubes, a step specific to MergeSharp, is insignificant compared to the time to compute vertex locations.

MergeSharp takes slightly more time on gradient data than on hermite data. The gradient based computation uses more vectors per cube than the hermite based computation.

Dataset	# cubes	Time (seconds)			
		pos	merge	trimesh	total
Weld	60389	6.14	0.13	1.36	7.63
Rotor	174323	11.6	0.3	1.4	13.3
Flange(h)	154380	11.1	0.22	1.25	12.57
Flange(g)	154380	9.98	0.22	1.25	10.45

Table 5: Execution times for MergeSharp on weld, rotor and flange datasets. All data set dimensions are 256^3. Flange(h) is computation time on hermite data. Flange(g) is computation time on gradient data. Second column (# cubes) is the number of cubes intersected by the isosurface. Reported times are time to compute isosurface vertex positions (pos), time to merge cubes using Algorithm 2 (merge), time to construct the triangle mesh (trimesh) and total time to construct the isosurface (total).

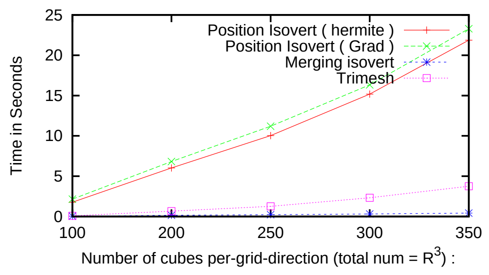


Figure 8: MergeSharp on flange datasets of various sizes. Graphs of time to position the isosurface using hermite data, time to position the isosurface using gradient data, time to merge grid cubes and time to extract isosurface triangles.

The gradient based computation also takes time to select the relevant gradients.

To compute the isosurface vertex location we use the Eigen Library, which we suspect is slow. A dedicated singular value decomposition solver for a 3x3 matrix would decrease the time to compute vertex locations.

10. Discussion and Future Work

MergeSharp produces significantly better results than the dual contouring algorithm [JLSW02] implemented in Polymender [Ju04]. It also does better than the tetrahedral based algorithm in [MS10]. MergeSharp also shows measurable robustness under noise and it can run directly from gradient data, not just hermite data.

We do not have implementations for the other algorithms for constructing isosurfaces with sharp features. We discuss those algorithms below.

The Extended Marching Cubes [KBSS01] behaves quite like the dual contouring algorithm of [JLSW02]. We believe that our algorithm would also do measurably better against Extended Marching Cubes on the same test cases given in 9.

The algorithms in [AB03, GK04, HWCO05, VKKM03, ZHK04] implicitly address the problems described in Section 5 by adding more vertices and polygons around sharp features. Our approach is the exact opposite, using fewer, not more, vertices to represent the isosurface near sharp features. We don't know how well those algorithms would perform on the FindSharp and CountDegree tests of Section 8, but they may do quite well. However, because of their subvoxel constructions, we believe that they are very sensitive to errors in their input data. We are interested in computing sharp features from gradient data and ultimately scalar data where the gradients are computed from a scalar grid. The precise position and gradient information needed for those algorithms is probably not available in scalar input data.

The algorithm in [SW04] is similar to the one in [MS10]. We believe it would have similar problems, producing small, thin triangles that poorly model the sharp features.

Many of the dual contouring algorithms support multi-resolution contouring. Our algorithm can easily be modified to support multi-resolution contouring in the smooth regions of the grid. Supporting multi-resolution contouring around the sharp features would be more difficult, but possible.

The use of fewer, not more, isosurface vertices around sharp features may seem counter-intuitive but we believe it is in fact correct. Continuity in smooth regions implicitly gives information about surface location. This information is missing near sharp features requiring the use of more sample points to determine feature location. Thus isosurface resolution should be lower, not greater, around sharp features. However, if our algorithm processes smooth regions with high levels of detail as regions containing a sharp feature, then some of that detail will be lost. This is an unfortunate, unintended consequence of our processing of sharp regions. We plan to run experiments to determine the extent of this problem. We also think that some extra processing to determine smooth regions with high levels of detail will help ameliorate this problem.

Our algorithm does not guarantee that the output isosurface is a manifold. We are currently working on modifying our algorithm to make this guarantee.

Our ultimate goal is to reconstruct sharp isosurfaces from just scalar data. Gradient information must be computed from the scalar data, and will inherently have some errors. Our algorithm which is robust under gradient errors is an important step toward our goal.

11. Acknowledgments

Work on this research was partially supported by NSF grant CCF-0635008.

References

[AB03] ASHIDA K., BADLER N. I.: Feature preserving manifold mesh from an octree. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications* (2003), ACM Press, pp. 292–297.

[BW13] BHATTACHARYA A., WENGER R.: *Experimental Results on MergeSharp*. Tech. Rep. OSU-CISRC-3-15-TR05, Dept. of Computer Science and Engineering, The Ohio State University, 2013.

[CDR07] CHENG S.-W., DEY T. K., RAMOS E. A.: Delaunay refinement for piecewise smooth complexes. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1096–1105.

[CDS13] CHENG S.-W., DEY T. K., SHEWCHUK J. R.: *Delaunay Mesh Generation*. Chapman and Hall / CRC computer and information science series. CRC Press, 2013.

[CK07] CHENEY E. W., KINCAID D. R.: *Numerical Mathematics and Computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2007.

[DGQ*12] DEY T. K., GE X., QUE Q., SAFA I., WANG L., WANG Y.: Feature-preserving reconstruction of singular surfaces. *Comp. Graph. Forum 31*, 5 (Aug. 2012), 1787–1796.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997* (1997), pp. 209–216.

[Gib98a] GIBSON S. F. F.: 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* (1998), Springer-Verlag, pp. 888–898.

[Gib98b] GIBSON S. F. F.: Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization* (1998), pp. 23–30.

[GK04] GRESS A., KLEIN R.: Efficient representation and extraction of 2-manifold isosurfaces using kd-trees. *Graphical Models 66*, 6 (2004), 370–397.

[HWCO05] HO C., WU F., CHEN B., OUHYOUNG M.: Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum 24* (2005), 2005.

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Transactions on Graphics 21*, 3 (2002), 339–346.

[Ju04] JU T.: Robust repair of polygonal models. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 888–895.

[JU06] JU T., UDESHI T.: Intersection-free contouring on an octree grid. In *Proceedings of the 14th Pacific Conference on Computer Graphics and Applications* (2006).

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001* (2001), ACM Press, pp. 57–66.

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (1987), 163–170.

[Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 259–262.

[MS10] MANSON J., SCHAEFER S.: Isosurfaces over simplicial partitions of multiresolution grids. *Computer Graphics Forum 29*, 2 (2010), 377–385.

[Nie04] NIELSON G. M.: Dual Marching Cubes. In *Proceedings of IEEE Visualization 2004* (2004), IEEE Computer Society, pp. 489–496.

[SJW07] SCHAEFER S., JU T., WARREN J.: Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (May 2007), 610–619.

[SW02] SCHAEFER S., WARREN J.: *Dual Contouring: The Secret Sauce*. Tech. Rep. TR 02-408, Dept. of Computer Science, Rice University, 2002.

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: Primal contouring of dual grids. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (2004), IEEE Computer Society, pp. 70–76.

[SYM10] SALMAN N., YVINEC M., MERIGOT Q.: Feature preserving mesh generation from 3d point clouds. *Computer Graphics Forum 29*, 5 (2010), 1623–1632.

[VKKM03] VARADHAN G., KRISHNAN S., KIM Y. J., MANOCHA D.: Feature-sensitive subdivision and isosurface reconstruction. In *Proceedings of IEEE Visualization 2003* (2003), IEEE Computer Society, pp. 99–106.

[Wan11] WANG C.: *Intersection-free Dual Contouring on Uniform Grids: An Approach Based on Convex/Concave Analysis*. Tech. rep., Dept. of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, 2011.

[ZHK04] ZHANG N., HONG W., KAUFMAN A.: Dual contouring with topology-preserving simplification using enhanced cell representation. In *Proceedings of IEEE Visualization 2004* (2004), IEEE Computer Society, pp. 505–512.