Constructing Isosurfaces with Sharp Features from Scalar Data - Tech Report

Arindam Bhattacharya and Rephael Wenger

January 13, 2014

Abstract

Previous algorithms for constructing isosurfaces with sharp features required gradients or surface normals to be provided as input to the algorithm. We present an algorithm for constructing isosurfaces with sharp features from scalar data on a regular grid. Our algorithm has two parts: (1) an algorithm to construct reliable gradients from scalar data; (2) an algorithm to construct isosurfaces with sharp features from scalar data combined with gradient data. We give experimental results on industrial CT scans. Our algorithm produces isosurface meshes with sharp edges and corners reliably represented by mesh edges and vertices. Our algorithm also produces reliable representations and good visualizations of sharp features in the isosurface.

1 Introduction

X-ray computed tomography (CT) scanners produce regular grids of scalar values representing material densities of scanned objects. These scalar values can be viewed as samples of some scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. An *isosurface* with *isovalue* σ is a surface which approximates the level set $f^{-1}(\sigma)$. The isosurface separates grid vertices with scalar values less than σ from grid vertices with scalar values greater than σ. If the scanned object has material density s different than the density s' of its surroundings, then an isosurface with isovalue σ between s and s' will represent the boundary surface Σ of the object. We are interested in reconstructing isosurfaces which represent boundary surfaces Σ with sharp edges and corners. In particular, we are interested in reconstructing isosurfaces from industrial CT scans of machine parts whose boundaries Σ have such sharp edges and corners.

Assume that a surface Σ is piecewise smooth, i.e. Σ is composed of a finite set of smooth surface patches whose boundaries are finite sets of smooth curves. Every point q in the interior of a smooth surface patch has a tangent plane and two opposing normal directions. *Non-smooth* points of Σ are points on the boundary of the surface patches where there is a discontinuity in the surface normals. Tangent planes and normal directions are not defined at non-smooth points.

An *edge* of Σ is a curve of non-smooth points on the intersection of the boundaries of two surface patches. A *sharp edge* of Σ is an edge with dihedral angle (defined in

Section 2) bounded away from 180 degrees. A *corner* of Σ is a non-smooth point which is the intersection of three or more sharp edges or does not lie on any sharp edge. A *sharp corner* of Σ is a corner with solid angle bounded away from 2π. Sharp edges and corners are more generally called *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 surface Σ. A sharp edge of Σ should be represented by a single, connected sequence of mesh edges with similar dihedral angle. A sharp corner of Σ should be represented by a single isosurface vertex with similar solid angle. 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.

Instead of scanned data, one could start with an explicit function $f : \mathbb{R}^3 \to \mathbb{R}$, sample the values of $f(v)$ on regular grid vertices v, and then reconstruct an isosurface representing $f^{-1}(\sigma)$ for isovalue σ. If $f^{-1}(\sigma)$ is piecewise smooth, then the isosurface mesh should model the sharp edges and corners of $\Sigma = f^{-1}(\sigma)$ as described above. Given an explicit function f, one can compute surface normals at intersection points of $f^{-1}(\sigma)$ and regular grid edges or compute gradients of f at grid vertices.

Previous algorithms to construct isosurfaces with sharp features relied upon exact surface normals being provided to the algorithm [3, 10, 12, 15, 16, 18, 27, 23, 28]. In recent work Bhattacharya et al. [1], gave a robust algorithm to construct isosurfaces with sharp features from gradient grid data where gradients are provided at each grid vertex. In this paper, we extend that work to scalar data where no surface normal or gradient information is provided. (See Figure 1.)

Our algorithm has two parts:

1. Construct reliable gradients from a regular grid of scalar values.

2. Construct an isosurface with sharp features from gradient data.

If the level set $f^{-1}(\sigma)$ of a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$ has sharp features, then there are discontinuities in the gradients of f at the sharp features. Constructing the gradients near those discontinuities is difficult. Formulas for approximating gradients such as the central difference formula [5] or higher order approximations [2, 13, 19] assume the gradient is continuous at the given location. Anisotropic diffusion [4, 6, 24, 25, 26] removes noise in low curvature regions of the scalar field without affecting high curvature regions, but it does not produce correct gradients near gradient discontinuities.

Instead of attempting to produce correct gradients at all grid vertices, our algorithm identifies correct gradients and uses only those gradients to predict locations of isosurface vertices. Gradients are marked correct if they agree with nearby gradients and if they correctly predict the scalar values at nearby grid points.

To construct the isosurface from gradient data, we use the MERGESHARP algorithm from [1]. However, the MERGESHARP algorithm as described in [1] only looks at gradients from grid cube $\mathbf{c}$ or adjacent cubes to predict the location of the isosurface vertex generated by $\mathbf{c}$. When constructing gradients from industrial CT data, the vertices of $\mathbf{c}$ and its neighbors may have few or no gradients which are marked correct. In Section 5, we modify MERGESHARP to selectively look at gradients in a larger neighborhood of $\mathbf{c}$.

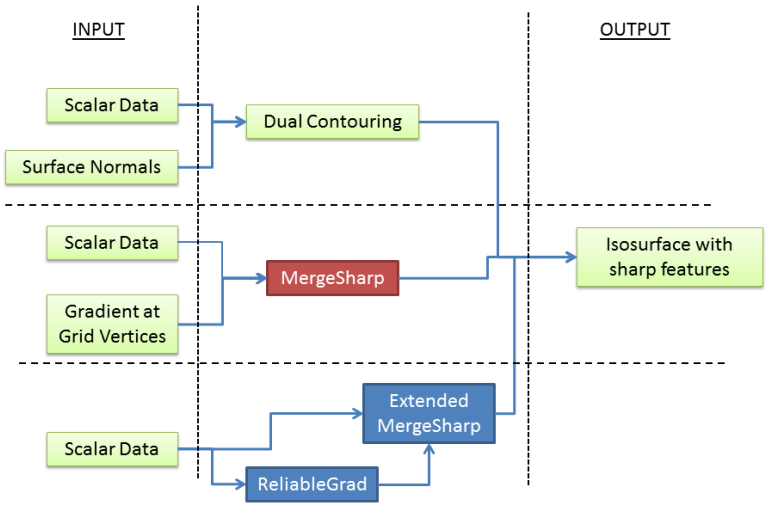


Figure 1: Approaches to feature reconstruction. (a) Dual contouring [15] and its variations require surface normals at the intersection of grid edges and isosurface vertices. (b) Algorithm MERGESHARP requires gradients at grid vertices. (c) Algorithm RELIABLEGRAD computes gradients for use by MERGESHARP allowing the reconstruction of isosurfaces with sharp features from only scalar data. RELIABLEGRAD does not compute gradients at vertices adjacent to sharp features. Thus, MERGESHARP must be modified to compute isosurface vertices from gradients in an extended neighborhood around a cube.

We present the results of our algorithm both on synthetic data and industrial CT data. To the best of our knowledge, this is the first presentation of isosurface with sharp features produced from industrial CT data.

Our algorithm represents sharp edges by mesh edges with dihedral angles below a threshold. The sharp edges can be quickly identified and visualized by displaying mesh edges with dihedral angles below that threshold, either together with the isosurface or separately.

Our paper contains four major contributions:

1. We give an algorithm for identifying correct gradients computed from scalar grid data;
2. We modify our MERGESHARP algorithm to selectively look for gradients in large neighborhoods around grid cubes;
3. We show that applying correct gradient identification and the modified MERGESHARP to industrial CT data produces isosurfaces with sharp features;
4. We apply our algorithm to identify and visualize sharp edges on an isosurface.

2 Definitions

For each smooth point q, let η_q be the normal at q which points in the direction of increasing f. Let B_p^ε be the ball of radius ε centered at point p and let $\partial\psi$ be the boundary of a surface patch ψ. The *dihedral angle* between two smooth surface patches ψ_1 and ψ_2 at point $p \in \partial\psi_1 \cap \partial\psi_2$ is $\lim_{\varepsilon\to 0} \inf\{\angle(\eta_{q_1}, -\eta_{q_2}) : q_1 \in \psi_1 \cap B_p^\varepsilon \text{ and } q_2 \in \psi_2 \cap B_p^\varepsilon\}$. Note that the normal η_{q_2} is reversed. The *dihedral angle* of an edge γ of Σ between ψ_1 and ψ_2 is the maximum dihedral angle over all points $p \in \partial\psi_1 \cap \partial\psi_2$.

3 Related Work

The most well known isosurface construction algorithm is MARCHING CUBES algorithm by Lorensen and Cline [17]. The algorithm places isosurface vertices on grid edges. Because isosurface vertices are restricted to grid edges, the MARCHING CUBES isosurface cannot represent sharp edges or corners.

Dual contouring is an isosurface construction technique which places isosurface vertices inside grid cubes intersected by the isosurface. For each grid edge intersected by the isosurface, dual contouring constructs a quadrilateral whose vertices lie in the four grid cubes containing the grid edge.

Gibson [8, 9] gave the first dual contouring algorithm for isosurface construction. Her algorithm placed only one isosurface vertex in a grid cube. Nielson's DUAL MARCHING CUBES algorithm [20] permits multiple isosurface vertices inside a grid cube. Each isosurface vertex lies on a different isosurface patch in the grid cube.

Kobbelt et al. [16] modified the MARCHING CUBES algorithm [17] to produce isosurfaces with sharp features. Input to their algorithm is a grid of directed distances

to an implicit surface. They constructed surface normals from this grid of directed distances and used the surface normals to position isosurface vertices on sharp features.

Ju. et al. [15, 22] constructed isosurfaces with sharp features using dual contouring [8, 9]. Input to their algorithm is a scalar grid combined with the set of surface normals at the intersection of the surface and the regular grid edges. They coined the term "hermite data" to describe such input. Ju [14] wrote a program Polymender for fixing meshing errors in polygonal meshes. Polymender contains an implementation of the dual contouring algorithm from [15].

Varadhan et al. [27] and Zhang et al. [28] gave dual contouring algorithms which produce isosurfaces which may intersect a grid edge multiple times. These algorithms model thinner features than the dual contouring algorithm by Ju et. al.

Algorithms by Ho et al. [12] and Ashida and Badler [3] construct a polygonal curve representing the intersection of the surface and a grid cube. They connect the curve to an isosurface vertex inside the grid cube, placing the isosurface vertex on sharp features.

Schaefer and Warren [23] and Manson and Schaefer [18] use a different approach to constructing isosurfaces with sharp features. They construct polygonal or tetrahedral grids whose vertices and edges lie on sharp isosurface features. They then extract the isosurface from these grids using MARCHING CUBES. The resulting isosurface mesh contains many triangles with angles near 180°.

All the above algorithms require a directed distance field or exact surface normals as part of the input.

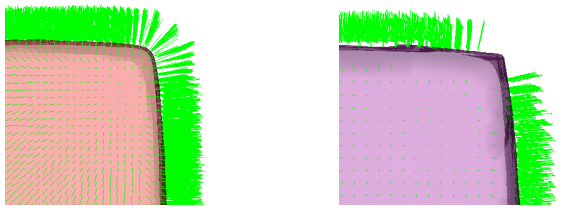
In [1], we gave in algorithm called MERGESHARP for constructing isosurfaces with sharp features from gradient data. The algorithm is briefly described in Section 5. Input to our algorithm is gradient grid data, scalar values and gradient vectors at each vertex of a regular grid. Our algorithm is much more robust than the previous algorithms, and can handle noise in the gradient vectors and missing gradient vectors.

Salman et al. [21] and Dey et al. [7] reconstructed piecewise smooth surfaces with sharp features from point cloud data by separately placing mesh vertices on the sharp features and vertices inside the smooth patches. They place "protecting ball" around mesh vertices on sharp features so that no vertices inside the smooth patches are placed near the sharp features. Algorithm MERGESHARP does something similar, merging grid cubes around sharp features so that isosurface vertices on sharp features are "isolated" away from other vertices.

Formulas for improving the numerical accuracy of gradient computations from scalar data are given in [2, 13, 19]. These formulas assume the gradient vector field is smooth and do not work when there are discontinuities in the vector field. They also do not work when there is noise in the input scalar data.

Anisotropic diffusion is a technique by which the filtering of surface normals or field gradients changes based on local curvature. Gradients or normals in low curvature regions are moved to agree with their neighbors. Gradients or normals in high curvature regions are moved only slightly. Anisotropic diffusion for mesh smoothing is described in [4, 6, 24, 25]. Tasziden et. al. [26] used anisotropic diffusion to preserve features in isosurface reconstruction.

Features in papers on anisotropic diffusion are high curvature regions, not regions with normal or gradient discontinuities (infinite curvature.) Anisotropic diffusion ap-



(a) Central difference gradients. (b) Correct gradients

Figure 2: Gradient computation around a sharp surface edge. (a) The central difference formula produces incorrect gradients near the sharp edge. (b) Gradients which are not near the sharp

plied to surfaces or gradient fields with discontinuities will filter noise from smooth regions, but it will not improve estimations at discontinuities or assist in identifying such discontinuities.

4 Determining Correct Gradients

The first part of our reconstruction algorithm computes correct gradients from a regular grid of scalar values. We assume that the scalar values represent the values at the grid vertices of a piecewise smooth scalar field f. A scalar field $f : \mathbb{R}^3 \to \mathbb{R}$ is *piecewise smooth* if $\mathbb{R}^3$ can be partitioned into a finite set of piecewise smooth regions, such that f has derivatives of all orders on each region. Gradients are computed at some, but not all, of the grid vertices. In particular, gradients are not computed at grid vertices on or adjacent to points where the scalar field is not smooth.

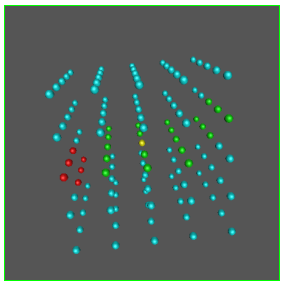
To compute a gradient in a piecewise smooth scalar field, we need all the scalar values used in the computation to be from a single smooth portion of the scalar field. Thus, we want to use a small basis for our gradient computation and not extend our gradient computation over many grid vertices. We use the central difference formula

$$\partial f(x)/\partial(x_d) \approx (f(x+u_d) - f(x-u_d))/(2|u_d|), \tag{1}$$

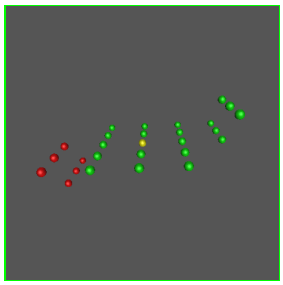
where x is the location of a grid vertex and u_d is the vector to the adjacent vertex in direction d. Figure 2a shows the result of computing gradient using the central difference formula.

If spacing between grid vertices is the same in all directions, then the grid can be rescaled so that u_d is a unit vector in all directions. However, CT scans often have non-uniform spacing, with the z or slice direction different from the x and y directions. In that case, u_x and u_y will have different magnitudes from u_z.

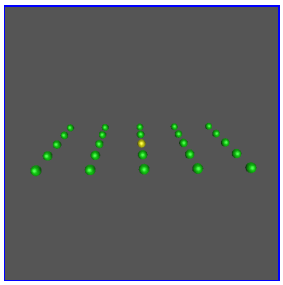
(a)



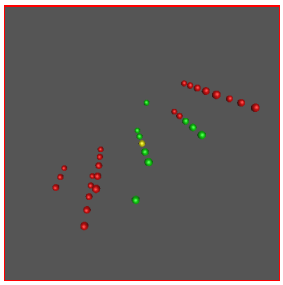
(b)



(c)



(d)



(e)

Figure 3: Results from scalar predictor test. Each yellow vertex is the center of a $5 \times 5 \times 5$ subgrid. Cyan vertices are farther than 0.5 units from plane π_v. Green vertices are at most 0.5 units from plane π_v and at most 0.4 units from plane $\tilde{\pi}_{v,v'}$. Red vertices are at most 0.5 units from π_v but farther than 0.4 units from $\tilde{\pi}_{v,v'}$. (b) shows a $5 \times 5 \times 5$ subgrid around green vertex in (a). Fig (c) shows only the vertices from Fig (b) which are near the plane π_v.(d) shows vertices from subgrid around red vertex in Fig (a). Finally, (e) shows vertices from subgrid around blue vertex in (a).

```
   ANGLETEST(Grid, v)
 1 numAgree ← 0;
 2 if (MAGNITUDE(g_v) ≤ 0.001) then return (false);
 3 foreach grid vertex v' adjacent to v do
 4 |   if (MAGNITUDE(g_v) ≥ 0.001) then
 5 |   |   if (ANGLE(g_v, g_v') ≤ 20) then
 6 |   |   |   numAgree ← numAgree + 1;
 7 |   |   end
 8 |   end
 9 end
10 if (numAgree ≥ 4) then return (true);
11 else return (false);
```

Algorithm 1: Algorithm ANGLETEST

```
   SCALARTEST(Grid, v)
 1 R ← 5 × 5 × 5 subgrid centered at v;
 2 τ ← the afine transformation mapping Grid to the uniform grid with edge
   length one;
 3 π_v ← plane {x : (x − τ(v)) · τ(g_v) = 0};
 4 foreach grid vertex v' in R do
 5 |   if (DISTANCE(τ(v'), π_v) ≤ 0.5) then
 6 |   |   π̃_{v,v'} ← plane {x : s_v + (x − τ(v)) · τ(g_v) = s_v'};
 7 |   |   errorDist ← DISTANCE(τ(v'), π̃_{v,v'});
 8 |   |   if (errorDist > 0.4) then return (false);
 9 |   end
10 end
11 return (true);
```

Algorithm 2: Algorithm SCALARTEST

The central difference formula estimates the gradient at a vertex v from six neighboring vertices of v which share a grid edge with v. If v and its six neighbors lie in the same smooth region of f, then errors in the gradient approximation computed by Equation 1 depend upon the curvature of f in the region and errors in the scalar values at the grid vertices. Assuming that f has low curvature in the given region and scalar value errors are small, the central difference formula will be a good approximation of the gradient. By low curvature, we mean that the difference between the gradient direction and magnitude at v and any of its six neighbors is small (less than 10 degrees variation.) Scalar value errors are small if they are a small fraction of the gradient (less than 10 percent.) If v and its six neighbors do not all lie in the same smooth region of f, then the central difference formula may be a very poor approximation of the gradient. We wish to identify such gradients and mark them as incorrect.

We use two tests to determine if a gradient g is correct, an angle test (Algorithm 1) and a scalar predictor test (Algorithm 2). For the angle test, we compare g with its

six immediate neighbors, gradients at vertex locations $x+u_d$ and $x-u_d$. If the angle between the two gradients is less than 20 degrees, then we say that the two gradients agree. If a gradient agrees with four out of its six neighbors, then it passes the angle test. We consider gradients whose magnitude is less than 0.001 to effectively be zero gradients and ignore them.

Comparing a gradient with only its immediate neighbors can give false positives, where multiple incorrect gradients agree. It can also give false negatives, since not only must a gradient be correct, but four of its six neighbors must also be correct. Therefore, we also use a scalar predictor test.

Each gradient vector g, its vertex position p and the scalar value s_p at p determines a scalar field

$$f_{p,g}(x) = s_p + (x-p)\cdot g. \tag{2}$$

The level sets $f_{p,g}(s)^{-1}$ of that field are planes in $\mathbb{R}^3$. Let q be the position of some vertex in the scalar grid with scalar value s_q. The plane $f_{p,g}^{-1}(s_q)$ represents locations of points with scalar value s_q as predicted by gradient vector g. For uniform grids with grid edge length one, the prediction error is the distance from q to $f_{p,g}^{-1}(s_q)$.

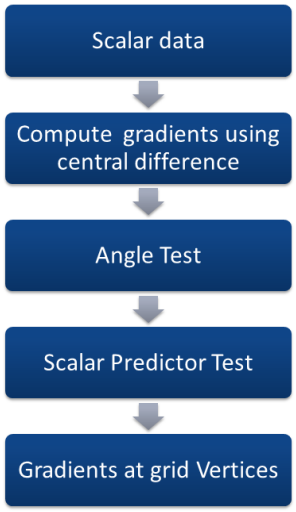


Figure 4: Algorithm RELIABLE-GRAD.

For non-uniform grids or uniform grids with grid edge length other than one, the distance from q to $f_p^{-1}(s_q)$ depends upon the edge lengths. To eliminate this dependence, we apply an affine transformation τ mapping the grid to a uniform grid with edge length one. The prediction error is the distance from $\tau(q)$ to $f_{\tau(p),\tau(g)}^{-1}(s_q)$.

We apply the scalar predictor test not just to immediate neighbors of the grid vertex at position p, but to vertices which are in a $5\times5\times5$ subgrid of vertices centered at p. However, we only wish to apply the scalar predictor test at grid vertices which are near the isosurface. Thus we compute the plane $f_{\tau(p),\tau(g)}^{-1}(s_p)$ through $\tau(p)$ and apply the scalar predictor test to vertices in the $5\times5\times5$ subgrid which are within distance 0.5 of this plane. Note that at least one endpoint of every grid edge intersected by this plane is within distance 0.5 of the plane.

Figure 3a shows a $5\times5\times5$ subgrid around a grid vertex v (yellow). The gradient g_v at v defines a plane π_v through v orthogonal to g_v. Subgrid vertices which are farther than 0.5 units from plane π_v are colored cyan.

The gradient g_v and scalar value s_v define a scalar field f_v (Equation 2). Each subgrid vertex v' with scalar value $s_{v'}$ defines a plane $\tilde{\pi}_{v,v'} = f_v^{-1}(s_{v'})$. A vertex v' which is within 0.5 units of π_v but at distance more than 0.4 from $\tilde{\pi}_{v,v'}$ is colored red. A vertex v' which is within 0.5 units of π_v and 0.4 units of $\tilde{\pi}_{v,v'}$ is colored green. If there are no red vertices, then the central yellow vertex is marked correct.

The yellow vertex in Figure 3b is the same as the yellow vertex in Figure 3a. The

yellow vertex is near a surface edge but about two units away from that edge. The gradient at the yellow vertex correctly predicts gradients to its right but not to its left. The yellow vertex in Figure 3c is on a smooth part of the surface. It correctly predicts all gradients in its vicinity. The yellow vertex in Figure 3d is almost on the surface edge. It does not correctly predict gradients on its right or its left.

Figure 2b displays the gradients from Figure 2a which passed the angle test and the scalar test. Figure 5 (c) shows gradients which passed both tests near a sharp surface corner.

The angle test compares a gradient with the gradient of its immediate neighbors. Since gradients at neighboring vertices are computed using scalar values of their neighbors, the angle test depends on the same $5 \times 5 \times 5$ subgrid as the scalar test. We could use larger neighborhoods for both angle and scalar test which would increase the reliability of predicting a gradient correct, but might mean misclassifying some correct gradients as incorrect.

Algorithm RELIABLEGRAD computes a gradient at each grid vertex using the central difference formula and then calls ANGLETEST and SCALARTEST on each gradient. (See Figure 4.) Gradients which fail either test are reset to $(0,0,0)$ indicating that the gradient at that vertex is unknown.

Routines ANGLETEST and SCALARTEST have a number of fixed parameters. Parameter choices are discussed in Section 8.

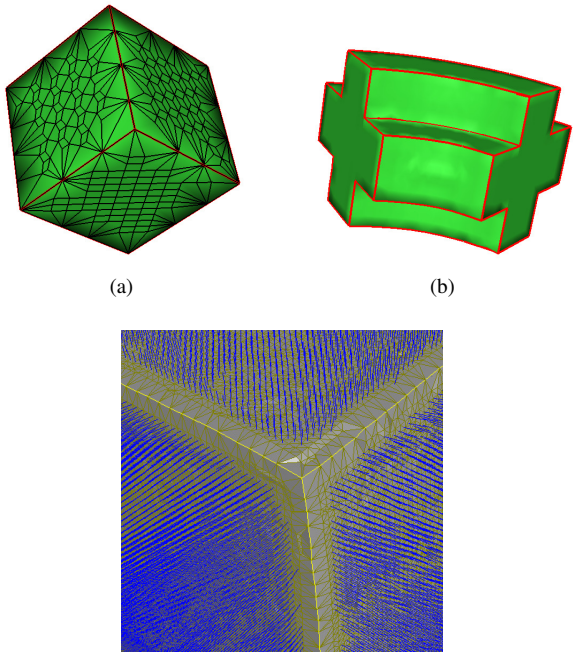
5 MergeSharp

The second part of our reconstruction algorithm reconstructs an isosurface from a set of correct gradients at grid vertices. We use our algorithm MERGESHARP described in [1]. For completeness, we give a brief description here.

Algorithm MERGESHARP has four steps. First, MERGESHARP computes isosurface vertex locations for each grid cube $\mathbf{c}$ intersected by the isosurface. These isosurface vertices are identified as lying on sharp corners or sharp edges or smooth regions of the isosurface. Second, MERGESHARP selects a set of "sharp" isosurface vertices on sharp corners and edges. Third, MERGESHARP applies Nielson's DUAL MARCHING CUBES algorithm [20] to construct the dual contouring isosurface. Lastly, MERGESHARP merges each selected "sharp" isosurface vertex with nearby unselected isosurface vertices.

MERGESHARP computes the location of isosurface vertex $w_{\mathbf{c}}$ for grid cube $\mathbf{c}$ as follows. The gradient g and scalar value s_v at each grid vertex v define a scalar field $f_{v,g}$ (Equation 2) and a plane $\pi_v = f_{v,g}^{-1}(\sigma)$ with isovalue σ. MERGESHARP computes these scalar fields $f_{v,g}$ and planes π_v for each vertex of $\mathbf{c}$ and its neighboring cubes. It places isosurface vertex $w_{\mathbf{c}}$ at the position which minimizes the least squares distance to those planes.

In computing the least squares distance to the planes π_v, MERGESHARP computes the singular values of an array A formed from the plane normals. If A has only one large singular value, then vertex $w_{\mathbf{c}}$ is in a smooth region of the surface. If A has two large singular values, then vertex $w_{\mathbf{c}}$ is on a sharp edge of the surface. If A has three large singular values, then vertex $w_{\mathbf{c}}$ is on surface corner.



(a) (b)

(c)

Figure 5: (a) Isosurface mesh constructed by MERGESHARP. Each isosurface vertex on a sharp corner or edge (red) is far from its neighbors. (b) Red lines are mesh edges with dihedral angle less than 140°. (c) Correct gradients which passed both the angle test and the scalar test

MERGESHARP selects a set of "sharp" isosurface vertices by selecting cubes which generated sharp isosurface vertices. If a cube is selected, then the 26 adjacent cubes are not selected. The 26 adjacent cubes are "covered" by the selected cube. MERGESHARP repeatedly selects cubes generating sharp isosurface vertices until all remaining cubes generating sharp vertices are covered by some selected cube. Of course, the selected set depends upon the order of selection. MERGESHARP first selects cubes generating sharp corners, and then selects cubes generating sharp edges. Among each class of cubes, MERGESHARP prefers cubes which generate isosurface vertices which are closer to the cube center. MERGESHARP also avoids selecting cubes whose isosurface vertices would create degenerate, zero-area triangles with already selected isosurface vertices.

Let $w_{\mathbf{c}}$ be the isosurface vertex generated by a selected cube $\mathbf{c}$. MERGESHARP merges the isosurface vertices generated by the 26 neighboring cubes of $\mathbf{c}$ with $w_{\mathbf{c}}$. If a cube is covered by more than one selected cube, it is merged with the isosurface vertex generated by the first selected cube. Figure 5 (a) contains an example of part of a surface created by MERGESHARP. Note that each isosurface vertex on a sharp corner or sharp edge is relatively far from its neighbors.

In [1], MERGESHARP is described for uniform grids. However, the algorithm can be easily adjusted for non-uniform grids. Coordinates of grid vertices are computed in a world space. Isosurface vertex locations are computed from the grid vertex locations and gradient in a world space. As described in Section 4, let τ be the affine transformation mapping the grid to a uniform grid with edge length one. When distances between points or planes must be compared or tested against thresholds, the points or planes are transformed using τ and the distances are computed in the transformed space.

6 Extending the Neighborhood in MergeSharp

When gradients are computed from scalar data as in Section 4, the gradients near non-smooth points in the scalar field will be incorrect. These computed gradients will fail either ANGLETEST or SCALARTEST and be reset to $(0,0,0)$. If a sharp feature intersects a grid cube, then many or all of the gradients at its vertices will be $(0,0,0)$. Many gradients at neighboring cubes may also be $(0,0,0)$. These gradients are ignored by MERGESHARP. Thus for gradients computed from scalar data, we must use a larger neighborhood of gradients in computing isosurface vertex positions.

We select vertices from an $8 \times 8 \times 8$ subgrid centered at $\mathbf{c}$ to compute $w_{\mathbf{c}}$. To choose the vertices from the subgrid we apply a number of tests. 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 $3 \times 3 \times 3$ centered at $\mathbf{c}$ and only choose a vertex v if the plane π_v intersects $\mathbf{c}'$.

In smooth, curved surfaces, choosing gradients at vertices far from $\mathbf{c}$ can cause the creation of non-existent sharp features in the isosurface. Thus, we wish to only choose a vertex which is far from $\mathbf{c}$ if the closer vertices are not chosen.

Let Q be the set of vertices of the subgrid whose gradients are $(0,0,0)$. 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

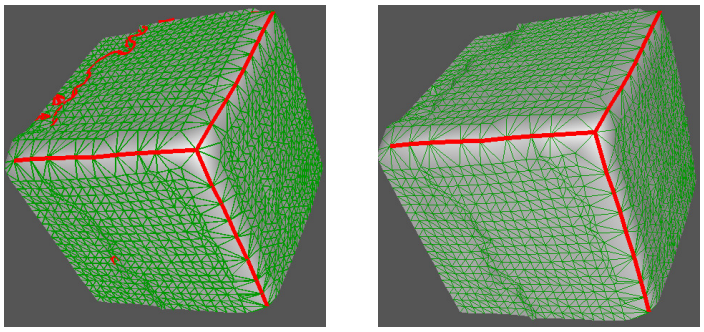
(u,v) where (u,v) is a grid edge. We find the connected component G' of $G_{\mathbf{c}}$ containing $Q_{\mathbf{c}}$. A grid vertex $u \notin V(G')$ is on the boundary of G' if (u,v) is a grid edge and v is in $V(G')$. We only a choose a vertex if it is in $Q_{\mathbf{c}}$ or if it is on the boundary of G'.

Applying the three tests gives a set of vertices and their gradients which define planes. We place isosurface vertex $w_{\mathbf{c}}$ at the position which minimizes the least squares distance to those planes.

MERGESHARP has two major parameters which determine its behavior: a singular value threshold and the subgrid size for computing $w_{\mathbf{c}}$. The singular value threshold determines which singular values are considered significant. The singular values are normalized by dividing by the largest value. Singular values below the threshold are set to 0. The default threshold is 0.1 and this threshold is used throughout this paper and in [1].

As in RELIABLEGRAD, the subgrid size poses the most problems. In fact, this subgrid size should be set in conjunction with the subgrid size used in ANGLETEST. It should be large enough to get correct gradients in the neighborhood of sharp features, but not so large that it finds gradients which are unrelated to the feature. Subgrid sizes are discussed further in Section 9.

Algorithm MERGESHARP assumes that all the edges of the grid cubes have equal length. As mentioned in the previous section, we subsampled the industrial CT data so that grid cube edge lengths were approximately equal.

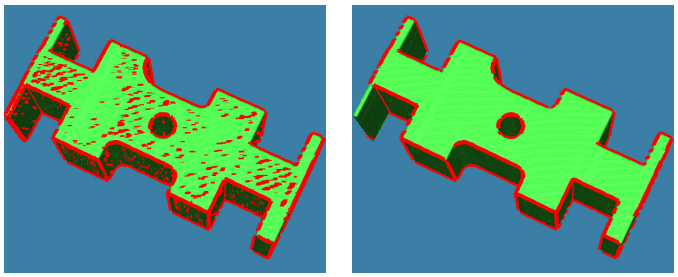


(a) (b)

Figure 6: (a) Red lines are edges with dihedral angle less than 140°. (b) Red lines are edges with dihedral angle less than 140° whose endpoints are identified as sharp by MERGESHARP.

7 Visualization of Sharp Edges

As described in [1], sharp edges of $f^{-1}(\sigma)$ can be easily identified and visualized by selecting mesh edges with dihedral angle under less than a threshold. Figure 5 (b) is an example of an isosurface whose mesh edges with dihedral angle less than 140° are drawn in red. On isosurfaces constructed from synthetic data such as Figure 5,

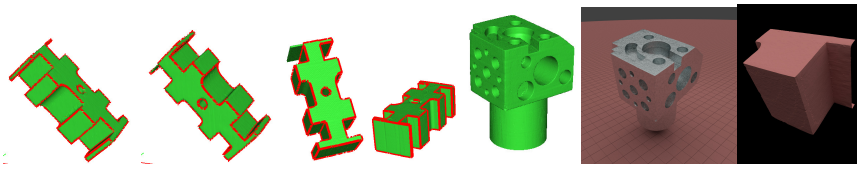


(a) (b)

Figure 7: Isosurface produced by RELIABLEGRAD and MERGESHARP from the Honda rectangular calibration data set. (a) Red lines are edges with dihedral angle less than 140°. (b) Red lines are edges with dihedral angle less than 140° whose endpoints are identified as sharp by MERGESHARP.

FINDSHARP does not return any mesh edges in the smooth regions of the isosurface. However, industrial CT data contains noise and sampling artifacts which can create produce mesh edges with high dihedral angle in the smooth portions of the isosurface. For instance, the red edges in Figure 6 (a) are mesh edges with dihedral angle less than 140°. The surface edge and corner is correctly represented by the red edges. However, there are also spurious red curves on smooth portions of the surface. These spurious curves are sampling artifacts and should not be identified as sharp features.

As described in Section 5, MERGESHARP categorizes each isosurface vertex as smooth or sharp. We can use this categorization to eliminate mesh edges with dihedral angle less than 140° whose endpoints are not sharp. The red edges in Figure 6 (b) are mesh edges with dihedral angle less than 140° whose endpoints are sharp. Figure 7 shows the result on a full dataset. Note, in Figure 7(b) the spurious edges have not been detected.



(a) (b) (c) (d) (e) (f) (g)

Figure 8: The Honda calibration pieces. (a-d) shows the honda rectangular piece scanned in four orientations,(a) Honda-45. (b) Honda-45-sideways. (c) Honda-45-45. (d) Honda-flat. Mesh edges with dihedral angle less than 140° and sharp endpoints are colored red. (e) shows the Honda cylindrical piece, (f-g) shows Blender renderings the Honda cylindrical piece and a corner of the Honda rectangular piece, generated from ReliableGrad and MergeSharp.

8 Parameters

RELIABLEGRAD has six parameters which could potentially be changed: a magnitude threshold (0.001), an angle threshold (20 degrees), a minimum agreement number (4), the maximum distance to $\tilde{\pi}_v$ (0.5), the maximum distance to $\tilde{\pi}_{v,v'}$ (0.4) and the subgrid size ($5 \times 5 \times 5$). Gradients with magnitude less than the magnitude threshold are treated as zero gradients and ignored (steps 2 and 4 in ANGLETEST). Isosurfaces representing object boundaries tend to have high gradients in their vicinity so this parameter has little effect on their reconstruction.

Gradients whose angles are within the angle threshold are counted as agreeing. We used an angle threshold of 20 degrees. Consider a uniform grid with edge length one. If v and w are adjacent grid vertices on an isosurface sphere of radius three (principal curvature 1/3), then the angle between the gradients at p and q is about 20 degrees. If p and q are on isosurface spheres of radii four, five, or six, then the angles between the gradients are approximately 15, 12, and 10 degrees, respectively. Thus, even with approximation and measurement errors, gradients near smooth surfaces with curvature less than 1/6 pass the angle test. This curvature bound must be adjusted for non-uniform grids or uniform grids with edge length other than one, but curvatures are always scale dependent.

We require the number of gradients which agree with g_v in ANGLETEST (step 11) to be at least 4. Since we test gradients in six directions, this means that for some direction d, gradients at $x + e_d$ and $x - e_d$ both agree with g_v.

In SCALARTEST, we only test g_v on vertices which are within the maximum distance of 0.5 to π_v. By using a max distance of 0.5, we ensure that for each grid edge intersecting π_v, we test g_v against at least one endpoint of that grid edge.

To test g_v on the scalar value at w, we compute a plane $\tilde{\pi}_{v,w}$ based on the scalar value at w. Plane $\tilde{\pi}_{v,w}$ is defined using the affine transformation τ which maps the grid to the uniform grid with edge lengths one. Gradient g_v passes the test if $\tau(w)$ is within distance 0.4 of $\tilde{\pi}_{v,w}$. The units are cube edges so g_v is predicting v' to within 2/5 of its actual location. Note that this parameter is independent of the magnitude of the gradient or the length of grid edges.

The last parameter, the size of the subgrid R used in SCALARTEST, is the most difficult to choose and is the most dependent on the input data. Small subgrids may cause the misidentification of gradients as correct, misleading MERGESHARP in positioning isosurface vertices and identifying them as sharp. Large subgrids will increase the accuracy and reliability of gradients marked correct but cause many more gradients which are marked as incorrect. Increasing the number of incorrect gradients may mean that there are no correct gradients on a thin facet, causing MERGESHARP to miss the sharp edges on those facets. Parameter choices are discussed further in Section 9 describing experimental results.

9 Experimental Results

We applied our algorithm on a number of industrial CT data sets at different orientations and sampling resolutions.We use, four different CT scans of a rectangular cali-

bration piece from Honda American Manufacturing (Figure 8 (a-d),(g)), two different scans from Honda of a cylindrical calibration piece (Figure 8 (e-f)), and the 400 volt connector data set described in [11]. The Honda calibration pieces are machined aluminum used for calibration of industrial CT scanners. Honda data sets were acquired using a 1536 matrix flat panel detector. In the first Honda data set (Honda-flat), the rectangular calibration piece is sampled so that its faces are parallel to the *xy*, *yz* and *xz* planes of the regular grid. In the second Honda data set (Honda-45), the calibration pieces are oriented in the (1, 0, 1) direction so some faces are parallel to the *xz*-plane of the regular grid. The orientation in the third Honda data set (Honda-45-sideways) is the same as the second, but the rectangular piece is rotated 90 degrees around the (1, 0, 1) axis. In the fourth Honda data set (Honda-45x45), the calibration piece are oriented in the (1, 1, 1) direction so no faces are parallel to any coordinate planes. The first cylindrical calibration piece is sampled so that its flat faces are parallel to the *xy*, *yz* and *xz* planes of the regular grid. In the second scan object is tilted and no faces are parallel to the planes of the regular grid. The 400volt data is also oriented along (1,1,1).

Dataset	Sample Res (mm)	Grid Dimensions
Honda-flat	0.0201x0.0201x0.031	325x660x140
Honda-45	0.110x0.110x0.031	1200x600x453
Honda-45-sideways	0.110x0.110x0.031	1150x500x448
Honda-45x45	0.0201x0.0201x0.031	630x600x382
Honda-cyl	0.0201x0.0201x0.031	450x450x396

Table 1: Sample resolutions along with scalar grid dimensions.

Table 1 contains sampling resolutions. Sampling resolution in the slice direction was always different than in the *x* and *y* directions.

Figures 8 and 12 show the results of constructing isosurfaces with sharp features from scalar data using RELIABLEGRAD and MERGESHARP. The red edges are edges with dihedral angle less than 140°.

As can be seen from Figure 8 (a-e) and 12, our algorithm did a good, but not perfect, job of identifying sharp surface edges in the Honda calibration pieces and representing them by sharp mesh edges. In all isosurfaces, there are some discontinuities along the sharp mesh edges. As expected, the best results are for Honda-flat and Honda-cyl, which both had axis aligned orientations. As discussed in [1], reconstruction of sharp features on axis-aligned isosurfaces is easier than reconstruction on isosurfaces which are not axis-aligned.

Our reconstruction algorithm correctly constructed sharp edges on even the thinnest facet in Honda-flat. It also correctly reconstructs the thinnest facet in all the non axis-aligned data sets except for honda-45x45, where it incorrectly reconstructs the sharp edges on the thinnest facet. The algorithm positions isosurface vertices on these sharp edges giving the appearance of a correctly reconstructed sharp edge. However this edge is actually covered by numerous degenerate, zero area triangles instead of being represented by a sequence of edges with dihedral angle near 90°. This failure shows some of the limitations of our algorithm.

The edges with dihedral angle less than 140° and their endpoints can be viewed as a

Dataset		RELIABLEGRAD Times (sec)			
Name	# Cubes	Central Diff	AngleTest	ScalarTest	Total
Honda-flat	30030K	1	3.45	9.82	13.27
Honda-45	326160K	10.92	36	77.77	113.77
Honda-45-sideways	257600K	9.5	29	73	102
Honda-45x45	7640K	4.65	14.76	11.82	26.58
Honda-cyl	80190K	2.68	9	15	24

Table 2: All times are in seconds. Second column is number of cubes in the scalar grid used as input to RELIABLEGRAD and MERGESHARP.

Dataset		MERGESHARP Times (sec)			
Name	# Cubes	Position	Merge	Trimesh	Total
Honda-flat	30030K	44.37	0.15	0.65	45.17
Honda-45	326160K	124.02	0.55	6.53	131.1
Honda-45-sideways	257600K	122.47	0.48	4.96	127.91
Honda-45x45	7640K	64.63	0.25	2.36	57.24
Honda-cyl	80190K	93.57	0.3	1.57	95.44

Table 3: All times are in seconds. Second column is number of cubes in the scalar grid used as input to RELIABLEGRAD and MERGESHARP.

graph embedded in $\mathbb{R}^3$. In [1], the authors measure the correctness of a reconstruction based on this graph. In a perfect reconstruction, the number of vertices with degree $k \neq 2$ should match the number of sharp corners with k incident sharp edges in $f^{-1}(\sigma)$. The difference between the degree counts for the reconstruction and $f^{-1}(\sigma)$ gives a useful, quantitative measure of the deviation of the sharp features of the reconstruction from $f^{-1}(\sigma)$.

Reconstructions from industrial CT data contain many edges with dihedral angle less than 140° which do not correspond to sharp surface edges. As suggested in Section 7, we filter edges with dihedral angle less than 140° by requiring their endpoints to also be identified as sharp by MERGESHARP. This reduced set of edges can also be viewed as a graph embedded in $\mathbb{R}^3$. We compare the number of vertices with degree $k \neq 2$ to the number of sharp corners with k incident sharp surface edges to measure the correctness of sharp features in our reconstruction.

In Table 4, we report the degree counts of vertices for sharp edges of the Honda rectangular calibration piece. The sharp edges have dihedral angle less than 140 degrees and sharp endpoints. The graph of sharp edges should have 48 degree 3 vertices. All other vertices should have degree 2. The high number of degree 1 vertices and the presence of some vertices with degree above 3 indicates that our reconstruction is still far from perfect.

We know of no prior, published experimental results of constructing isosurfaces with sharp features from scalar data. However, we wished to compare our algorithm (Figure 1c) with a variation where MERGESHARP was replaced by the dual contouring algorithm of Ju et. al. [15, 14, 22] or gradients were computed just by central dif-

DataSet	Degree 1	Degree 3	Degree >3	Total
Honda-flat	201	76	19	296
Honda-45	1578	414	209	2201
Honda-45-sideways	2066	436	187	2689
Honda-45x45	373	399	116	888
Honda-cyl	832	301	64	1197

Table 4: Degree counts on isosurfaces from the Honda data sets. Sharp edges were extracted using SELECTIVEFINDSHARP.

Gradient Computation	Reconstruction Algorithm	Degree 1	Degree 3	Degree >3	Total
Central Difference	Dual Contouring	5343	1296	678	7317
RELIABLEGRAD	Dual Contouring	6516	13793	277	20586
Central Difference	MERGESHARP	1072	45	53	1170
RELIABLEGRAD	MERGESHARP	373	399	116	888

Table 5: Degree counts on isosurfaces from different algorithms on the honda-45-45 data set. The Honda-45-45 dataset is challenging, because it is not axis aligned and does not have isotropic resolution. Sharp edges were extracted using SELECTIVEFIND-SHARP.

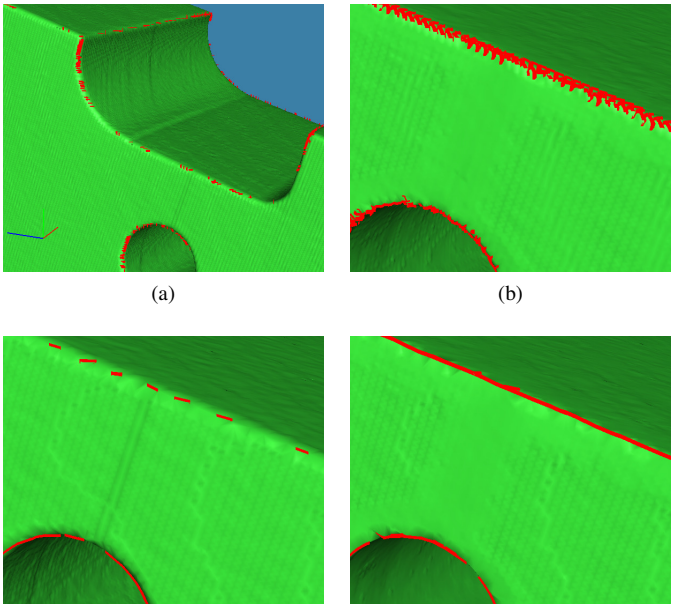
ference, not RELIABLEGRAD. Dual contouring as described in [15] requires surface normals to calculate isosurface vertex locations. Instead of computing such surface normals from gradients, we directly calculate the isosurface vertex locations directly from gradients as described in [1]. The step of computing surface normals from gradients can only add errors to the calculation of isosurface vertex locations.

Table 5 contains results from different variations of our algorithm. The first two rows give results for the dual contouring algorithm of [15]. The third row shows MERGESHARP applied to gradients computed by central difference. The last row is the results of this algorithm. Figure 9 shows samples of the results in table 5. This test was run on other non-axis aligned datasets with similar results.

While there have been many other algorithms proposed for constructing isosurface with sharp features, they all rely upon exact placement of isosurface vertices based on exact surface normals. We think they will have similar problems to dual contouring when isosurface vertex locations can only be approximated.

Next, we evaluate gradients generated from RELIABLEGRAD with gradients from synthetic data sets. For this, we generate a flange dataset fig.10 (a). When compared with the gradients whose magnitudes are above a minimum value, 261 gradients had difference more than 5 degrees between the RELIABLEGRAD results and the true gradients. Only 6 gradients had difference more than 10 degrees and non above 16 degrees (the maximum being 15.7 degrees). In other data sets we found similar results.

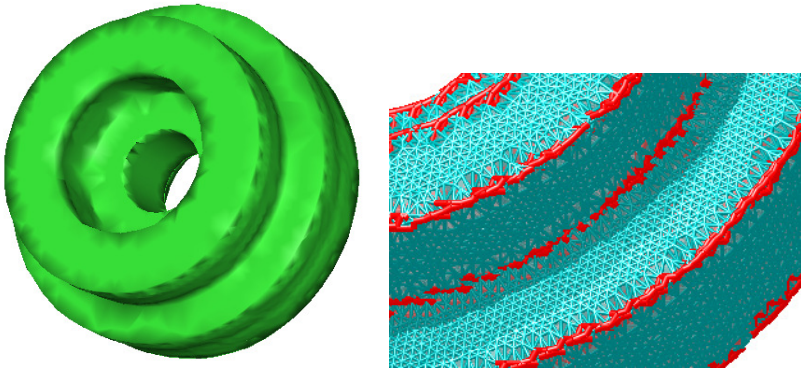
Figure 10 evaluates the effect of adding noise to this data. In Figure 10 (a) we see a close-up of the result of running RELIABLEGRAD and MERGESHARP on the flange data. The scalars were perturbed by adding noise from an uniform distribution between -0.2 and 0.2. Figure 10 (b) shows the result of running Central Difference followed by



(a) (b)

(c) (d)

Figure 9: (a) Shows the sharp edges extracted after running Central Difference and Dual Contouring, due to the smooth change in direction of gradients computed using Central Difference across edges, dual contouring captures few sharp edges or corners. Thus, the high degree 1 count in Table 5. (b) ReliableGrad followed by dual contouring captures a lot of spurious small edges as it is unable to select the correct gradients which leads to erroneous location of the dual vertex.(c)shows the result of applying MergeSharp on the results of Central Difference. (d) shows the result of the current algorithm which gives superior results.



(a) (b)

Figure 10: Effect of adding uniform noise. (a) shows the flange dataset, the scalar values of (a) are perturbed uniformly within a given bound.(b) shows the result of applying ReliableGrad and MergeSharp (c) shows the result of applying Central Difference and Dual Contouring.

Dual Contouring. We note in the second case the sharp edges are jagged and noisy.

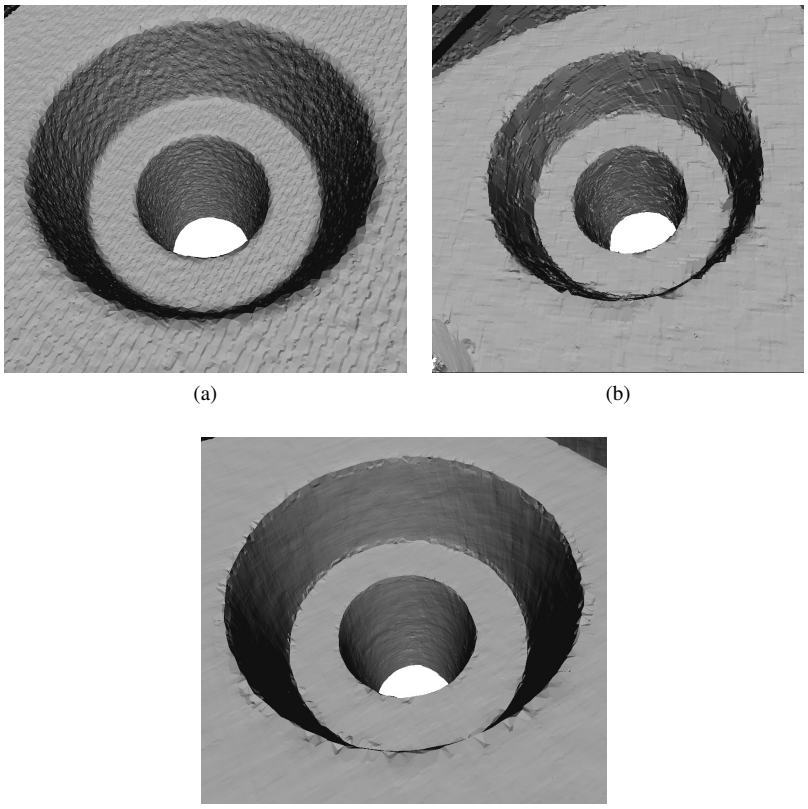
We also wanted to check how RELIABLEGRAD and MERGESHARP performs in noise added to actual CT data. For this we added noise from an uniform distribution between (-100.0 to 100.0) to the 400volt dataset. On the original data set, which is not axis aligned the reconstruction of the individual sockets is very good (Figure11 (c) for a close up , Figure 12 (a) for an overview). In Figure 11(a) we show the result of running Central Difference with MERGESHARP, next, in Figure 11 we show the result of running RELIABLEGRAD with MERGESHARP. This shows even with significant noise, RELIABLEGRAD with MERGESHARP performs better than currently utilized techniques

We end this section with a discussion of subgrid sizes used in RELIABLEGRAD and MERGESHARP and the maximum distance to $\tilde{\pi}_{v,v'}$ used in RELIABLEGRAD. We constructed the isosurface, extracted mesh edges with dihedral angle under 140° and sharp endpoints and counted the degrees of the resulting graph.

If the maximum distance to $\tilde{\pi}_{v,v'}$ was decreased (for example to 0.2) then large portions of sharp edges would be missed (Figure 3 (c,d)). In our experiments varying the RELIABLEGRAD subgrid size or the MERGESHARP subgrid size did not create significant changes in output be it simulated data or CT data from different sources. We do note that decreasing the RELIABLEGRAD subgrid size to very small, would identify most gradients as incorrect and give poor results.

9.1 Timings

Running times of RELIABLEGRAD and MERGESHARP on the different data sets are given in Table 3, 2 respectively. Running times were on an intel machine with 16GB RAM and a 3Ghz processor. The running time of RELIABLEGRAD is proportional to



(a) (b)

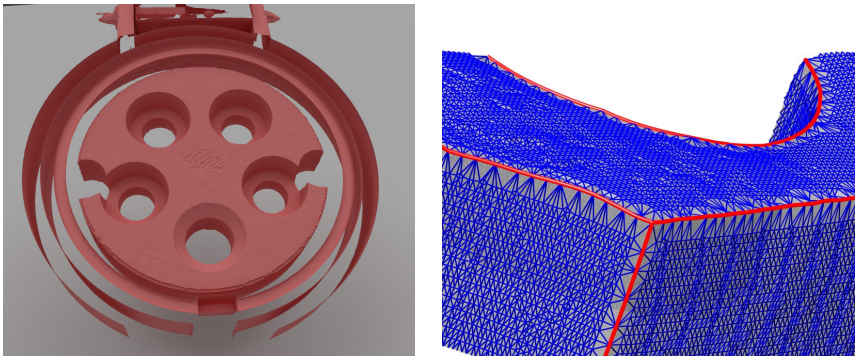
(c)

Figure 11: Results of adding noise to a part of the 400v dataset. (a) Central Difference and MergeSharp on noisy data, (b) ReliableGrad and MergeSharp on noisy data.(c) ReliableGrad and MergeSharp on original data.

the grid size. The SCALARTEST subroutine is much more expensive than ANGLETEST because it iterates over a $5 \times 5 \times 5$ neighborhood around each vertex.

RELIABLEGRAD needs to be run only once in a preprocessing step to compute gradients at all grid vertices. MERGESHARP can then be applied to the scalar and gradient data to compute isosurfaces at different isovalues.

The running time of MERGESHARP is proportional to the number of grid cubes intersected by the isosurface. The time to calculate isosurface vertex positions far outweighs the running times of the other steps in MERGESHARP. Computing isosurface vertex positions is a common step of all algorithms which reconstruct sharp features.



(a) (b)

Figure 12: (a) A rendering of the bottom of the 400volt data set, generated by applying ReliableGrad and MergeSharp. (b) shows a close-up from the Honda cylindrical piece.

10 Conclusion and Future Work

We showed that isosurfaces with sharp features can be reconstructed from industrial CT scalar data. This is a major improvement over previous algorithms which require exact surface normals or gradients. The reconstruction can be used to render and visualize the sharp isosurface features.

The reconstructed isosurface produced by MERGESHARP is not necessarily a manifold. We plan to modify MERGESHARP so that it guarantees manifold isosurfaces. The problem is to guarantee that isosurfaces are manifolds without substantially affecting the reconstruction of sharp features.

Our test data was of surfaces whose smooth faces met at 90 degree angles. We plan to experiment on surfaces with other angles. As the sharp dihedral angle grows smaller, the sharp edge becomes harder to distinguish from a smooth region with high curvature. On the other hand, as the sharp dihedral angle becomes larger, there are fewer samples at the tip of the angle and it becomes harder to compute and represent that tip.

References

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

[2] U. Alim, T. Moller, and L. Condat. Gradient estimation revitalized. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1495–1504, Nov. 2010.

[3] 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.

[4] C. L. Bajaj and G. Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graph.*, 22:4–32, January 2003.

[5] E. W. Cheney and D. R. Kincaid. *Numerical Mathematics and Computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2007.

[6] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In *Proc. of IEEE Visualization 2000 (VIS'00)*, pages 397–405, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[7] 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.

[8] 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.

[9] 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.

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

[11] C. Heinzl, J. Kastner, and E. Groller. Surface extraction from multi-material components for metrology using dual energy ct. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1520–1527, 2007.

[12] 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.

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

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

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

[16] 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.

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

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

[19] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In *Proceedings of the 8th conference on Visualization '97*, VIS '97, pages 19–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

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

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

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

[23] 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.

[24] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 125–132, Washington, DC, USA, 2002. IEEE Computer Society.

[25] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM Trans. Graph.*, 22(4):1012–1033, Oct. 2003.

[26] T. Tasdizen and R. T. Whitaker. Anisotropic diffusion of surface normals for feature preserving surface reconstruction. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM 2003)*, pages 353–360, 2003.

[27] 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.

[28] 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.