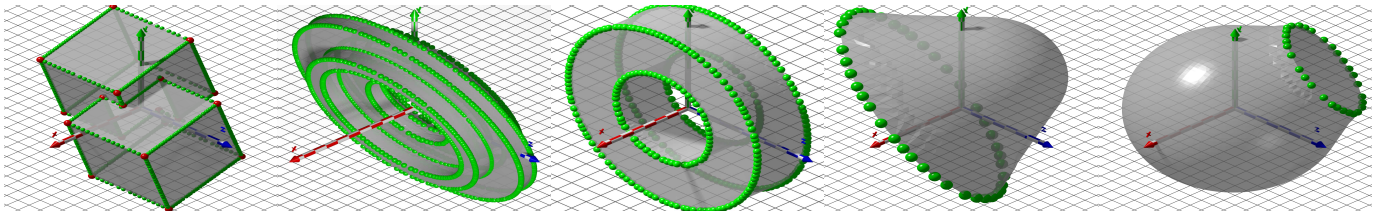
Computing Reliable Gradients from Scalar Data - Technical Report

Arindam Bhattacharya and Rephael Wenger



(a) TwoCube (b) Flange (c) Annulus (d) Cone (e) Cannon

Fig. 1: Sharp isosurface mesh and features generated by our algorithm. Sparse edge features are in green and corners in red.

Abstract— Sharp surface edges and corners often provide important visual information about the surface. However, algorithms for isosurface visualization often smooth sharp edges and corners, hiding instead of highlighting the sharp features. Previous algorithms for identifying sharp isosurface features required gradients or surface normals to be provided as input to the algorithm. We present an algorithm for determining sharp isosurface edges and corners from scalar data on a regular grid. Our algorithm uses the central difference formula to construct gradient approximations in the scalar field, and then identifies which gradient approximations are reliable. Because our algorithm uses only local information to determine sharp features, it is fast and easily parallelizable. Our algorithm can be used to generate an isosurface with sharp features or to visualize or highlight the sharp features.

Index Terms—Isosurface, sharp features, reconstruction, gradient, scalar field

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 modeled as samples of some scalar field $f : \mathbb{R}^3 \to \mathbb{R}$. Object boundaries can be visualized by direct volume rendering or by visualizing isosurfaces (mesh representations of $f^{-1}(\sigma)$) representing the object boundaries. Both approaches have difficulties in representing sharp edges and corners in the object boundaries. Sharp edges and corners are best represented as discontinuities in the gradient field of f. However, standard direct volume rendering and isosurface construction algorithms implicitly assume some continuity in the gradient field of f.

In this paper, we will describe a fast, local algorithm for reliably reconstructing the gradient field of f and constructing points on sharp edges and corners of an isosurface. The points can be rendered in conjunction with isosurface visualizations to highlight sharp features or they can be joined to form a skeleton representation of the sharp feature. They can also be used as input to isosurface or surface meshing algorithms which are designed to handle surfaces with sharp features.

Previous algorithms to construct isosurfaces with sharp features relied upon exact surface normals being provided to the algorithm [3, 11, 12, 14, 15, 17, 26, 22, 27]. The algorithm in [1] constructs isosurfaces with sharp features from gradient grid data where gradients are provided at each grid vertex. We know of no work on constructing isosurfaces with sharp features directly from scalar 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 [7] or higher order approximations [2, 13, 18] assume the gradient is continuous at the given location. Anisotropic diffusion [6, 8, 23, 24, 25] 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. We give an algorithm for identifying correct gradients based on their agreement with neighboring gradients. The algorithm produces enough correct gradients in the neighborhood of sharp features to generate points on those sharp features.

The basic steps of our algorithm are given in Figure 2. The first two steps produce a set of reliable gradients. The algorithm then selects a set of reliable gradients around each cube, computes a set of isosurface tangent planes from those gradients, and finds a point at the "intersection" of those tangent planes. It simultaneously identifies whether that point lies on a sharp edge or corner of the isosurface or on a smooth region of the isosurface. The algorithm sparsifies the set of isosurface vertices on sharp features and returns the sparsified set. The sparsified set can be used for generating an isosurface mesh using feature preserving algorithms such as MergeSharp [1] or Weighted Cocone [9]. Alternatively, it can be used to construct a representative sharp feature curve or can be rendered directly to highlight sharp features on the isosurface.

The focus of this paper is on constructing a set of reliable gradients from scalar data in the presence of gradient discontinuities. As part of our work, we developed a theory (Section 3 giving bounds on gradient approximation errors based on comparing gradients to nearby gradients.

There is a substantial amount of work on computing surfaces with sharp features from point cloud data. Some of the proposed algorithms (e.g. [9, 20]) compute points on sharp features and then create representative curves from those points. Scalar data might be turned into point cloud data by computing a set of points which approximate the intersection of the isosurface and the grid edges. Any of the point

- *The Ohio State University. E-mail: wenger.4@osu.edu.*
- *bhattaca@cse.ohio-state.edu.*

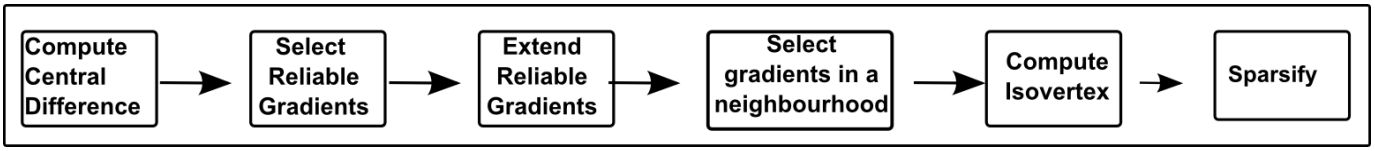


Fig. 2: Constructing points on sharp features.

cloud algorithms which reconstruct sharp features or surfaces with sharp features can then be applied to this point cloud data.

We strongly believe that there is a big advantage in constructing sharp features directly from scalar data without (the extra step of) converting it to point cloud data. First, converting scalar data to point cloud data ignores the grid structure of the scalar data. This grid structure can be employed both for constructing sharp features and for meshing those features. Generally, grid based methods are faster than their point-cloud counterparts. Second, point cloud data is extremely noisy so the point cloud reconstruction algorithms average over *large* neighborhoods. It is extremely difficult to predict the results of the algorithms or to guarantee that the algorithms are not ignoring features in the data. In contrast, our algorithm runs over a small, local neighborhood, so that all features of the data (for better or for worse) are represented in the output. Finally, because our algorithm is local, it is very fast and easily parallelizable.

Our research contributions in this paper are:

1. An algorithm for constructing reliable gradients from scalar data in the presence of gradient discontinuities.
2. Proofs of bounds on the angle between the approximate gradients and the true gradients.

We use our algorithm to:

1. Construct samples points on sharp edges and corners of an isosurface.
2. Construct isosurface meshes with good representations of sharp edges and corners.

We note that the algorithm for constructing the reliable gradients and the sample points is completely local, and thus fast and easily parallelizable.

2 Related Work

Algorithms for constructing isosurfaces with sharp features are given in [3, 12, 14, 15, 17, 21, 22, 26, 27]. All these algorithms require exact surface normals as part of the input.

MergeSharp [1] is an algorithm for constructing isosurfaces with sharp features from gradient data. Input to the algorithm is gradient grid data, scalar values and gradient vectors at each vertex of a regular grid. The algorithm can handle noise in the gradient vectors and missing gradient vectors.

Salman et al. [20] and Dey et al. [9] 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.

Fleishman et al. [10] introduced a least-squares technique to reconstruct a piecewise smooth surface. The sharp features are reconstructed as intersection of these smooth regions. Oztireli et al. [19] extended the moving least square reconstruction to sharp features using kernel regression. The strength of robust kernel regression, makes this method robust to noise. Avron et al. [4] used a l1-sparse approach to reconstruct sharp features from point set.

Formulas for improving the numerical accuracy of gradient computations from scalar data are given in [2, 13, 18]. 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.

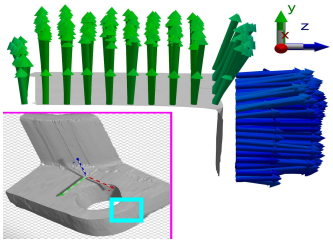


Fig. 3: Gradient computation around a sharp surface edge in a CT data. Expanded view of the cyan rectangle, shows the central difference gradients at grid vertices which intersect the isosurface. Those parallel to Y axis are colored green, those parallel to Z are colored blue, the rest are linearly interpolated. The central difference formula produces incorrect gradients near the sharp edge. Gradients which are not near the sharp edge are correct.

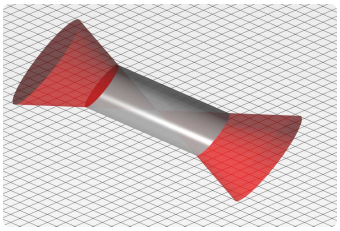


Fig. 4: A double cone (red) representing the gradient discontinuities of a scalar field f. The field f is the maximum of the distance to a line and to a plane orthogonal to that line. The isosurfaces of f are boundaries of cylinders. A sample isosurface is shown in grey. The double cone separates $\mathbb{R}^3$ into three regions. The field f is continuous within each region.

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 [6, 8, 23, 24]. Tasziden et. al. [25] 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 applied 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.

3 Determining Correct Gradients

We assume that the scalar values s_v represent the values at the grid vertices v of a continuous, piecewise smooth scalar field f. The difference between s_v and $f(v)$ is the "*noise*" in the data.

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, $A_1, A_2, \ldots, A_k$

such that f has derivatives of all orders on each region A_i. Because f is only piecewise smooth, the gradient field of f may have discontinuities. Such discontinuities occur only on the boundaries, ∂A_i, of the A_i.

Figure 4 contains an example of a continuous piecewise smooth field, consisting of three piecewise smooth regions separated by two cones (a "double cone"). The isosurface for this field is the boundary of a cylinder.

The gradient at a point p is the vector $(\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)$ at p. 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.

3.1 Definitions

We define the neighborhood of a grid vertex v. Let set $N_1(v)$ be v and the six grid vertices which share a grid edge with v. Recursively define set $N_k(v)$ as:

$$N_k(v) = \{v' : v' \in N_1(v'') \text{ for some } v'' \in N_{k-1}(v)\}.$$

We sometimes use $N(v)$ as an abbreviation for $N_1(v)$.

A *collinear sequence of adjacent grid vertices* is a sequence $(v_1, v_2, \ldots, v_k)$ of distinct grid vertices such that $v_i \in N(v_{i-1})$ for $i = 2, \ldots, k$ and all the v_i are collinear.

An *interior grid vertex* of a region A is a grid vertex v such that $N(v)$ is a subset of A. Set $\mathcal{I}_{\mathcal{V}}(A)$ is the set of all interior grid vertices of A.

3.2 Central Difference Formula

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

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

where x is the location of a grid vertex and u_d is the vector to the adjacent vertex in direction d. Figure 3 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.

Let g_v be the gradient at a vertex v and let $\tilde{g}_v$ be the gradient approximation produced by the central difference formula. There are three types of errors in the approximation of g_v by $\tilde{g}_v$. First, there are errors caused by noise in the data, i.e. the difference between the scalar value s_v and it "true" value $f(v)$. Second, there are errors caused by using the central difference formula as an approximation to the gradient. Such errors occur even if we used the exact values $f(v)$ and if the field was smooth everywhere. Finally, if v and one of its neighbors $v' \in N(v)$ lie in different smooth regions, then there may be a discontinuity in the gradient along edge (v, v'). This discontinuity will also contribute to errors in $\tilde{g}_v$. (Numerical error is a fourth contributor to errors in $\tilde{g}_v$, but it is insignificant compared to the errors caused by noise in the data.)

Replacing the central difference formula by an equation which relies on more vertices will reduce the first two sources of error but increase the effect of gradient discontinuities on the gradient approximation. Anisotropic filtering can be used to decrease noise in the scalar data without affecting the discontinuities, but it will not reduce the error caused by gradient discontinuity. Anisotropic filtering can also be used directly on the gradients. Again, it will improve gradients in the smooth regions, but it won't correct major errors caused by discontinuities. (It might correct minor ones.)

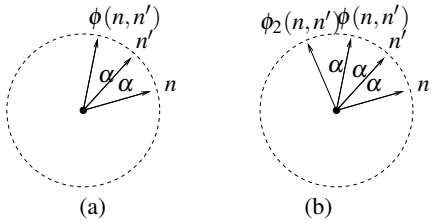


Fig. 5: (a) Vector $\phi(n,n')$ predicted by n and n'. (b) Vector $\phi_2(n,n')$ predicted by n and n'.

3.3 Reliable Gradients

For each vertex v, let $\tilde{n}_v = \tilde{g}_v/|\tilde{g}_v|$ be the unit vector in the direction of the central difference gradient $\tilde{g}_v$. We can try to determine if the gradient direction $\tilde{n}_v$ at vertex v is reliable by comparing it with the gradient directions $\tilde{n}_{v'}$ at all the neighboring vertices $v' \in N(v)$. If $\angle(\tilde{n}_v, \tilde{n}_{v'})$ is less than some constant α for all vertices $v' \in N(v)$, then we can mark the gradient at v as reliable.

Determining reliable gradients from $\angle(\tilde{n}_v, \tilde{n}_{v'})$ will work for flat regions but will fail if the scalar field has any significant curvature. If α is set to a small value, then the algorithm will fail to detect correct gradients in curved regions. On the other hand, if α is set to a large value, then the algorithm will mark incorrect gradients as correct. Instead of comparing $\tilde{n}_v$ to neighboring vertices, we use pairs of vertices to predict the gradient direction at v and compare $\tilde{n}_v$ to this predicted direction.

Let $n_v = g_v/|g_v|$ be the unit vector pointing in the direction of the true gradient g_v. Assume (v, v', v'') is a collinear sequence of adjacent vertices. Vectors n_v and $n_{v'}$ lie in a plane h. If the gradient changes at a constant rate along line segment (v, v''), then $n_{v''}$ also lies in plane h and $\angle(n_{v'}, n_{v''})$ equals $\angle(n_v, n_{v'})$.

Let n and n' be unit vectors in $\mathbb{R}^3$ lying plane h. Let $\phi(n,n')$ be the unit vector in h other than n whose angle with n' is $\angle(n,n')$. We say that the $\phi(n,n')$ is the vector *predicted by* n and n'. (See Figure 5(a).) More precisely, define $\text{Orth}(n,n')$ as $n - (n \cdot n')n'$, the component of n orthogonal to n. Define $\phi(n,n')$ as:

$$\phi(n,n') = n - 2 \times \text{Orth}(n,n') = n - 2(n - (n \cdot n')n') = 2(n \cdot n')n' - n.$$

As defined above, unit vector $\tilde{n}_v = \tilde{g}_v/|\tilde{g}_v|$ points in the direction of the central difference gradient. We determine reliable gradients by testing $\angle(\phi(\tilde{n}_{v''}, \tilde{n}_{v'}), \tilde{n}_v)$ against a constant α.

Input : Vertex v, Angle bound α.

foreach grid vertex $v' \in N(v)$ **do**
  Let $v'' \in N(v')$ be the vertex such that (v, v', v'') is a collinear sequence of adjacent vertices;
  if $(\angle(\phi(\tilde{n}_{v''}, \tilde{n}_{v'}), \tilde{n}_v) > \alpha)$ **then return** (**false**);
end
return (**true**)

Algorithm 1.

Assume $N_3(v)$ is a subset of a smooth region A_i so that $v, v', v'' \in \mathcal{I}_{\mathcal{V}}(A_i)$ for all the neighbors v' of v. If all the second order partial derivatives of function f in A_i are constant, then the gradients change at a constant rate along any direction and $\phi(n_{v''}, n_{v'})$ equals n_v. Moreover, if all the second order partial derivatives are constant and there is no noise ($s_v = f(v)$ for all v), then the central difference gradient $\tilde{g}_v$ equals the exact gradient g_v (up to numerical error.) (See Proposition 10 in the appendix for a proof.)

In the more general case, the second order partial derivatives are not

constant and there is noise in the data. To analyze this case, define:

$$\mu = \max\{\angle(n_v, \tilde{n}_v) : v \in \mathcal{I}_{\mathcal{V}}(A_i) \text{ for some } A_i\}.$$
$$\Lambda(n_v, n_{v'}, n_{v''}) = \angle(\phi(n_v, n_{v'}), n_{v''}).$$
$$\lambda = \max\{\Lambda(n_v, n_{v'}, n_{v''}) : v, v', v'' \in A_i \text{ for some } A_i\}.$$

Value μ is a bound on the angle between the approximate and exact gradient directions in the smooth regions of the field. $\Lambda(n_v, n_{v'}, n_{v''})$ is the difference between the prediction $\phi(n_v, n_{v'})$ and $n_{v''}$. This difference is caused by changes in the curvature of f. Value λ is a bound on this difference over vertices in the interiors of the A_i. Note that Algorithm 1 computes $\Lambda(\tilde{n}_{v''}, \tilde{n}_{v'}, \tilde{n}_v)$, not $\Lambda(n_v, n_{v'}, n_{v''})$.

The following proposition bounds $\Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v''})$ and $\angle(n_{v''}, \tilde{n}_{v''})$.

Proposition 1. *Let (v, v', v'') be a collinear sequence of adjacent vertices contained in A_i for some smooth region A_i.*

1. *If $v, v', v'' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then*
   $\Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v''}) \leq \Lambda(n_v, n_{v'}, n_{v''}) + 4\mu \leq \lambda + 4\mu$.
2. *If $v, v' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then $\angle(n_{v''}, \tilde{n}_{v''}) \leq \Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v''}) + 3\mu + \lambda$.*

Outline of proof of 1.1: Perturbing n_v by at most μ, changes $\phi(n_v, n_{v'})$ by at most μ. Since angles $\angle(n_v, \tilde{n}_v)$ and $\angle(n_{v''}, \tilde{n}_{v''})$ are at most μ, angle $\angle(\phi(\tilde{n}_v, \tilde{n}_{v'}), \tilde{n}_{v''})$ is at most $\angle(\phi(n_v, \tilde{n}_{v'}), n_{v''}) + 2\mu$.

Perturbing $n_{v'}$ by at most μ changes $\phi(n_v, n_{v'})$ by at most 2μ. Thus,

$$\begin{aligned}\Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v''}) &= \angle(\phi(\tilde{n}_v, \tilde{n}_{v'}), \tilde{n}_{v''}) \\ &\leq \angle(\phi(n_v, \tilde{n}_{v'}), n_{v''}) + 2\mu \\ &\leq \angle(\phi(n_v, n_{v'}), n_{v''}) + 2\mu + 2\mu \\ &= \Lambda(n_v, n_{v'}, n_{v''}) + 4\mu.\end{aligned}$$

□

Outline of proof of 1.2: By the triangle inequality,

$$\begin{aligned}\angle(n_{v''}, \tilde{n}_{v''}) &\leq \angle(\phi(n_v, n_{v'}), n_{v''}) + \angle(\phi(n_v, n_{v'}), \tilde{n}_{v''}) \\ &\leq \lambda + \angle(\phi(n_v, n_{v'}), \tilde{n}_{v''}).\end{aligned}$$

As discussed above, perturbing n_v by at most μ changes $\phi(n_v, n_{v'})$ by at most μ. Perturbing $n_{v'}$ by at most μ changes $\phi(n_v, n_{v'}$ by at most 2μ. Thus,

$$\begin{aligned}\angle(n_{v''}, \tilde{n}_{v''}) &\leq \lambda + \angle(\phi(n_v, n_{v'}), \tilde{n}_{v''}) \\ &\leq \lambda + \angle(\phi(\tilde{n}_v, \tilde{n}_{v'}), \tilde{n}_{v''}) + 3\mu \\ &= \Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v''}) + \lambda + 3\mu.\end{aligned}$$

□

More complete versions of these proofs are in the appendix.

Assume that parameter α in Algorithm 1 is at least $\lambda + 4\mu$. By the first inequality, if $N_3(v) \subseteq A_i$ for some A_i, then $\Lambda(\tilde{n}_{v''}, \tilde{n}_{v'}, \tilde{n}_v) \leq \lambda + 4\mu \leq \alpha$ for all neighbors v' of v. Thus Algorithm 1 returns true.

On the other hand, assume that Algorithm 1 returns true and that $N_3(v)$ intersects at most two regions. Let A_i be the region containing v. Under the assumption that the boundary between these two regions is planar, some $v' \in N(v)$ is in $\mathcal{I}_{\mathcal{V}}(A_i)$. If (v, v', v'') is a collinear sequence of adjacent vertices, then v'' is also in $\mathcal{I}_{\mathcal{V}}(A_i)$. By the second inequality, $\angle(n_v, \tilde{n}_v) \leq \Lambda(\tilde{n}_{v''}, \tilde{n}_{v'}, \tilde{n}_v) + 3\mu + \lambda \leq \alpha + 3\mu + \lambda$. Thus, if Algorithm 1 returns true, then the angle between the approximate gradient direction $\tilde{n}_v$ and the true gradient direction n_v is bounded by $\alpha + 3\mu + \lambda$.

So far we've made the assumption that the surface between two smooth regions is planar. If we drop that assumption, then it is no longer true that for every vertex $v \in A_i$ some vertex in $N(v)$ is in $\mathcal{I}_{\mathcal{V}}(A_i)$. For instance, in the 2D example in Figure 6, no vertex in $N(v)$ lies in $\mathcal{I}_{\mathcal{V}}(A_1)$. Without this property, we can no longer guarantee a bound on $\angle(n_v, \tilde{n}_v)$ when Algorithm 1 returns true.

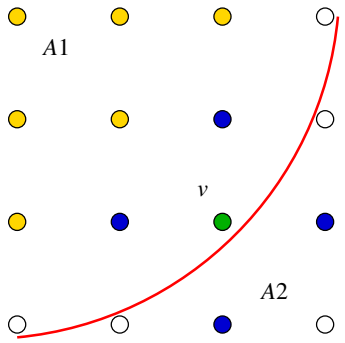


Fig. 6: Red curve is boundary separating regions A_1 and A_2. Green vertex v is in A_1 but no vertex of $N(v)$ lies in $\mathcal{I}_{\mathcal{V}}(A_1)$. Vertices in $N(v)$ are colored blue. Vertices in $\mathcal{I}_{\mathcal{V}}(A_1)$ are colored yellow.

To handle curved boundaries of the A_i, we must modify our algorithm to use vertices at edge distance 3 from v. We define $\phi_k(n_v, n_{v'})$ as the normal direction predicted by n_v and $n_{v'}$ at the vertex which is edge distance k from $n_{v'}$.

$$\begin{aligned}\phi_0(n, n') &= n', \\ \phi_1(n, n') &= \phi(n, n') = 2(n \cdot n')n' - n, \\ \phi_k(n, n') &= \phi(\phi_{k-2}(n, n'), \phi_{k-1}(n, n')), \\ \Lambda_k(n, n', n'') &= \angle(\phi_k(n, n'), n'').\end{aligned}$$

We replace ϕ in Algorithm 1 with ϕ_2.

Input : Vertex v, Angle bound α.

foreach grid vertex $v' \in N(v)$ **do**
  Let $v'', v''' \in N_3(v)$ be the vertices such that (v, v', v'', v''') is a collinear sequence of adjacent vertices;
  if $(\angle(\phi(\tilde{n}_{v''}, \tilde{n}_{v'}), \tilde{n}_v) > \alpha)$ **then return** (**false**);
  if $(\angle(\phi_2(\tilde{n}_{v'''}, \tilde{n}_{v''}), \tilde{n}_v) > \alpha)$ **then return** (**false**);
end
return (**true**)

Algorithm 2.

Let A_i be the smooth region containing vertex v. Let $\mathcal{X} = \cup_{A_j} \partial A_j$ be the union of all the boundaries of smooth regions A_j. If there is a sufficiently large ball containing v and not intersecting $\mathcal{X}$, then $v'', v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$ for some collinear sequence (v, v', v'', v''').

Proposition 2. *Let Γ be a regular grid whose edges all have the same length L. If some ball $\mathbb{B}$ of radius $(5/2)\sqrt{3}L$ contains grid vertex $v \in A_i$ and does not intersect $\mathcal{X}$, then there is a collinear sequence (v, v', v'', v''') of adjacent grid vertices such that $v'' \in \mathcal{I}_{\mathcal{V}}(A_i)$ and $v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$.*

To prove Proposition 2, we show that there $\mathbb{B}$ contains $N_v(v'')$ and $N_v(v''')$ for some collinear sequence (v, v', v'', v'''). The proof is in the appendix.

The following proposition bounds $\Lambda_2(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v'''})$ and $\angle(n_{v'''}, \tilde{n}_{v'''})$.

Proposition 3. *Let (v, v', v'', v''') be a collinear sequence of adjacent vertices contained in A_i for some smooth region A_i.*

1. *If $v, v', v'', v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then*
   $\Lambda_2(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v'''}) \leq \Lambda_2(n_v, n_{v'}, n_{v'''}) + 6\mu \leq 3\lambda + 6\mu$.
2. *If $v, v' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then*
   $\angle(n_{v'''}, \tilde{n}_{v'''}) \leq \Lambda(\tilde{n}_v, \tilde{n}_{v'}, \tilde{n}_{v'''}) + 3\lambda + 5\mu$.

Proofs of these relationships are in the appendix.

As in the discussion, of Algorithm 1, Property 3 can be used to show that if $N_4(v) \subset A_i$ for some A_i, then Algorithm 2 returns true. On the other hand, assume Algorithm 2 returns true. By Proposition 2, there is a collinear sequence of grid vertices (v, v', v'', v''') such that $v'', v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$. By Proposition 3, the angle between the approximate gradient direction $\tilde{n}_v$ and the true gradient direction n_v is bounded by $\alpha + 3\lambda + 5\mu$.

```
DOESORTHMATCHA(v, α1, α2)
1  if (N^O(v) = ∅) then return (true);
2  foreach grid vertex v' ∈ N^O(v) do
3  |   Let v'', v''' ∈ N3(v) be the vertices such that (v, v', v'', v''') is
   |   a collinear sequence of adjacent vertices
   |   flagMatch ← false;
4  |   if (∠(ñv, φ(ñv'', ñv')) ≤ α2) and
5  |      (∠(ñv, φ2(ñv''', ñv'')) ≤ α2) then flagMatch ← true ;
6  |   if (∠(ñv, ñv') ≤ α1) and
7  |      (∠(ñv, ñv'') ≤ α1) then flagMatch ← true ;
8  |   if (flagMatch = false) then return (false) ;
9  end
10 return (true)
```

Algorithm 3. Algorithm DOESORTHMATCHA.

```
DOESORTHMATCHB(v, α1)
1  foreach grid vertex v' ∈ N^O(v) do
2  |   if (v ∈ N^O(v')) then
3  |   |   if (∠(ñv, ñv') ≤ α1) then return (true);
4  |   end
5  end
6  return (false)
```

Algorithm 4. Algorithm DOESORTHMATCHB.

```
FINDRELIABLE(v, α1, α2)
/* α1 and α2 are angle bounds                                   */
1  foreach grid vertex v' ∈ N_v^T do
2  |   Let v'', v''' ∈ N3(v) be the vertices such that (v, v', v'', v''') is
   |   a collinear sequence of adjacent vertices;
3  |   if (∠(ñv, φ(ñv'', ñv')) > α2) then return (false);
4  |   if (∠(ñv, φ2(ñv''', ñv'')) > α2) then return (false);
5  end
6  if DOESORTHMATCHA (v, α1, α2) then return (true) ;
7  else if DOESORTHMATCHB (v, α1) then return (true);
8  else return (false);
```

Algorithm 5. Algorithm FINDRELIABLE

3.4 CT Data

Algorithm 2 has two problems when applied to CT data. First, in CT data scalar values near gradient discontinuities are very unreliable. At such vertices, the angle between the true gradient n_v and the estimated gradient direction $\tilde{n}_v$ is also very unreliable. Thus, we cannot assume that angle is bounded by a constant μ on vertices near gradient discontinuities. Second, gradient magnitudes drop off quickly away from the surface boundaries and gradient directions are quickly meaningless. This is particularly true if the gradient direction is the z-direction and the CT data is reconstructed in planar x-y slices. We address each of these problems.

The first problem with CT data is that scalar values near gradient discontinuities are very unreliable. At such vertices, the angle between n_v and $\tilde{n}_v$ can be much greater than in the rest of the data set. Unfortunately, gradients at vertices near gradient discontinuities are exactly the gradients which we need to generate sharp features.

Gradients generated from scalar data rely upon the accuracy of the scalar data. CT scanners do not measure scalar values directly at each grid vertex. Instead, they measure the intensity of rays passing through the scanned object. The resulting measurements are called projection data. The projection data is transformed into scalar data by using a Radon or similar transformation of by solving a large set of linear equations. The resolution of the scalar data is usually set to equal the resolution of the projection data.

The process of determining scalar values at grid vertices is provably reliable in regions where field gradients vary slowly and continuously. However, it is highly unreliable near discontinuities in the field gradients. The result is that scalar values at grid vertices adjacent to gradient discontinuities are highly unreliable and the angle between the true gradient and the estimated gradient can be much larger than in the rest of the data.

Fortunately, the scalar errors drop off quickly away from the gradient discontinuities. We found that scalar values at a grid vertex v was reliable within an acceptable tolerance as long as no edge incident on v intersected a gradient discontinuity. Equivalently, the scalar value at $v \in A_i$ was reliable as long as $N(v)$ was a subset of A_i. Under this assumption, if $N_2(v)$ is contained in some smooth region A_i, then the scalar values at vertices in $N(v)$ are close to their true values. This implies that the angle between n_v and $\tilde{n}_v$ is small.

Let A_i be the smooth region containing grid vertex v. Assume that the boundary of A_i is flat around v. We can no longer assume that the angle between $n_{v'}$ and $\tilde{n}_{v'}$ is small for some $v' \in N(v)$. However, there is a collinear sequence (v, v', v'', v''') of adjacent vertices such that $N_2(v'')$ and $N_2(v''')$ are in A_i. Thus, $\tilde{n}_{v''}$ and $\tilde{n}_{v'''}$ are good estimates of $n_{v''}$ and $n_{v'''}$, respectively. By comparing $\tilde{n}_v$ with the direction $\phi_2(\tilde{n}_{v'''}, \tilde{n}_{v''})$ predicted by $\tilde{n}_{v'''}$ and $\tilde{n}_{v''}$, we can determine if $\tilde{n}_v$ is a reliable gradient. Note that this exactly what Algorithm 2 does.

The argument above assumes the boundary of A_i is flat around v. Without this assumption, we can no longer conclude that $N_2(v'')$ and $N_2(v''')$ are in A_i for some collinear sequence (v, v', v'', v''').

Assume that all grid edges have the same length L. Replace the assumption that a scalar value at $v \in A_i$ is reliable if $N(v) \subseteq A_i$ by the assumption that a scalar value at v is reliable if a ball $\mathbb{B}_{0.5L}(v)$ of radius $0.5L$ around v is contained in A_i. Under this assumption, if $\mathbb{B}_{1.5L}(v)$ is contained in some smooth region A_i, then the scalar values at the vertices $N(v)$ are all close to their true values. This implies that the angle between n_v and $\tilde{n}_v$ is small.

If $\mathbb{B}'$ is a sufficiently large ball containing v and A_i contains $\mathbb{B}'$, then there is a collinear sequence (v, v', v'', v''') of adjacent grid vertices such that A_i contains $\mathbb{B}_{1.5}(v'')$ and $\mathbb{B}_{1.5}(v''')$. Thus, $\tilde{n}_{v''}$ and $\tilde{n}_{v'''}$ are reliable gradients. Algorithm 2 determines if $\tilde{n}_v$ is reliable from $\tilde{n}_{v''}$ and $\tilde{n}_{v'''}$.

The second problem with CT data is that gradient magnitudes drop off quickly away from the surface boundaries. To address this problem, we divide the vertex neighbors of v into two sets. Define the the tangent neighbor set and the orthogonal neighbor set of v as:

$$
\begin{aligned}
N^T(v) &= \{v' \in N(v) : 20^\circ \leq \angle(\tilde{n}_v, (v' - v)) \leq 160^\circ.\} \\
N^O(v) &= N(v) - N_v^T \\
&= \{v' \in N(v) : \angle(\tilde{n}_v, (v' - v)) < 20^\circ \text{ or} \\
&\qquad \angle(\tilde{n}_v, (v - v')) < 20^\circ.\}
\end{aligned}
$$

$(v' - v)$ is the vector from v to v'. The orthogonal neighbor set may be empty.

Assuming that $\tilde{n}_v$ is relatively close to n_v, vertices in $N^T(v)$ are near the tangent plane at v and close to the isosurface through v. We handle gradient directions at those vertices as in Algorithm 2. Vertices in $N^O(v)$ are (relatively) far from the tangent plane and from the isosurface through v. Fortunately, the gradient directions at the vertices in $N^O(v)$ are much more reliable than the gradient directions at vertices in $N^T(v)$. Thus, we can use gradients at vertices in $N^O(v)$ to determine whether gradient $\tilde{n}_v$ is reliable. We reduce the number of nearby vertices whose gradient directions must match $\tilde{n}_v$ in three ways.

First, we note that there is little curvature along the gradient direction in the scalar field represented by a CT scan. Thus, in addition to the comparison of $\tilde{n}_v$ and $\phi_2(\tilde{n}_{v'''}, \tilde{n}_{v''})$, we can simply compare $\tilde{n}_v$ and

$\tilde{n}_{v''}$. If angle $\angle(\tilde{n}_v, \tilde{n}_{v''})$ is below some threshold α_1, then $\tilde{n}_{v''}$ can be a guarantor of the reliability of $\tilde{n}_v$. (See Algorithm 3.)

Second, instead of comparing $\tilde{n}_v$ and $\tilde{n}_{v''}$, we compare $\tilde{n}_v$ to its immediate neighbor $\tilde{n}_{v'} \in N(v)$ (Algorithm 4.) We still compare $\tilde{n}_v$ with vertices at edge distance 3 in the tangent directions. If some sufficiently large ball contains $v \in A_i$ and does not intersect $\mathcal{X} = \cup_{A_j} \partial A_j$, then either there is a $v' \in N^T(v)$ and collinear sequence (v, v', v'', v''') where $v'', v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$ or there is a $v' \in N^O(v)$ such that $v' \in \mathcal{I}_{\mathcal{V}}(A_i)$.

Third, in Algorithm 4 we replace the requirement that gradient directions of both vertices in $N^O(v)$ "match" $\tilde{n}_v$ by a requirement that the gradient direction of one vertex in $N^O(v)$ matches $\tilde{n}_v$. Let $N^O(v) = \{v'_1, v'_2\}$. We require that either the $\angle(\tilde{n}_v, \tilde{n}_{v'_1})$ or that $\angle(\tilde{n}_v, \tilde{n}_{v'_2})$ is small. To make sure that we are only applying the single check to gradients which truly point along an axis, we require also that $\tilde{n}_{v'_1}$ or $\tilde{n}_{v'_2}$ be close to the axis direction.

Algorithm 3 contains the check in both orthogonal directions. Algorithm 4 contains the additional check in one orthogonal direction. Because of the check $v \in N^O(v')$ in line 2 of Algorithm 4, there are cases where Algorithm 3 may return true while Algorithm 4 returns false. Algorithm 5 is the full algorithm.

Algorithm 5 loosens the conditions under which a gradient is identified as reliable. By Proposition 3, if $N_4(v) \subset A_i$ for some A_i, then Algorithm 5 returns true. What about the converse, i.e. if Algorithm 5 returns true?

We can show that for each $v \in A_i$, there is either a collinear sequence of grid vertices (v, v', v'', v''') such that $v' \in N^T(v)$ and $v'', v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$ or there is a vertex $v' \in N^O(v)$ such that $v' \in \mathcal{I}_{\mathcal{V}}(A_i)$. (See Proposition 9 in the appendix.) In the first case, the angle between the approximate gradient direction $\tilde{n}_v$ and the true gradient direction n_v is bounded by $\alpha_2 + 3\lambda + 5\mu$. In the second case, the angle between $\tilde{n}_v$ and n_v is bounded by $\angle(\tilde{n}_v, \tilde{n}_{v'}) + \kappa + \mu$ where κ is a bound on curvature. (See Prop 6 in the appendix.) If $\angle(\tilde{n}_v, \tilde{n}_{v'}) \leq \alpha_1$, then $\angle(\tilde{n}_v, n_v) \leq \alpha + \kappa + \mu$. However, Algorithm DOESORTHMATCHB (Algorithm 4), only guarantees that $\angle(\tilde{n}_v, \tilde{n}_w) \leq \alpha_1$ for one vertex $w \in N^O(v)$. What if w is not the vertex of $N^O(v)$ which is in $\mathcal{I}_{\mathcal{V}}(A_i)$?

We think that if $\angle(\tilde{n}_v, \tilde{n}_w) \leq \alpha_1$ for one $w \in N^O(v)$ and $\angle(\tilde{n}_v, \phi_2(\tilde{n}_{w'''}, \tilde{n}_{w''})) \leq \alpha_2$ for all collinear sequences (v, w', w'', w''') where $w' \in N^T(v)$, then $\angle(\tilde{n}_v, n_v)$ is small. However, we don't yet have a proof.

```
   EXTENDRELIABLE(α2, numIter)
1  for k ← 1 to numIter do
2  |  S ← ∅;
3  |  foreach vertex v do
4  |  |  foreach vertex v' ∈ N^T(v) do
5  |  |  |  Let v'', v''' ∈ N3(v) be the vertices such that
   |  |  |  (v, v', v'', v''') is a collinear sequence of adjacent
   |  |  |  vertices;
6  |  |  |  if (v', v'' and v''' are marked reliable) then
7  |  |  |  |  if (∠(ñv, φ(ñv'', ñv')) < α2) and
   |  |  |  |  (∠(ñv, φ2(ñv''', ñv'')) < α2) then
8  |  |  |  |  |  S ← S ∪ {v};
9  |  |  |  |  end
10 |  |  |  end
11 |  |  end
12 |  end
13 |  Mark each vertex v ∈ S as reliable;
14 end
```

Algorithm 6. Algorithm EXTENDRELIABLE

3.5 Extending Reliable Gradients

Once we've identified reliable gradients by Algorithm 5, we can use those reliable gradients to identify vertex neighbors with reliable gradients. By Proposition 3, if v'' and v''' are reliable and $\angle(\phi_2(\tilde{n}_{v'''}, \tilde{n}_{v''}), \tilde{n}_v)$ is small, then $\angle(\tilde{n}_v, n_v)$ is small. This is particularly helpful near gradient discontinuities, where Algorithm 5 will fail to identify correct gradients as reliable because some of their neighbors are incorrect.

Pseudo code for Algorithm EXTENDRELIABLE is given in Algorithm 6. Because the initial set of reliable gradients is computed If the initial set of reliable gradients is computed using (v, v', v'', v'''), the value of numIter is set to 2.

```
   RELIGRAD(α1, α2, numIter)
1  foreach grid vertex v do
2  |  FINDRELIABLE(v,α1,α2)
3  end
4  EXTENDRELIABLE(α2, numIter)
```

Algorithm 7. Algorithm RELIGRAD

3.6 Algorithm RELIGRAD

Our final algorithm, named RELIGRAD, applies Algorithm FINDRELIABLE to each vertex and then calls EXTENDRELIABLE. Pseudo code is in Algorithm 7.

4 SELECTING GRADIENTS

Once we have identified reliable gradients, we use those gradients to compute planes tangent to the isosurface in the neighborhood of each cube $\mathbf{c}$. Algorithm MergeSharp in [1] uses gradients at the vertices of $\mathbf{c}$ and cubes adjacent to $\mathbf{c}$. Unfortunately, if cube $\mathbf{c}$ intersects a gradient discontinuity, this neighborhood may contain no or few reliable gradients. We can only guarantee that a vertex $v \in A_i$ will pass the angle test in Step 3 of FINDRELIABLE (Algorithm 5), if $N(v''') \subseteq A_i$ for each collinear sequence (v, v', v'', v''') where $v' \in N^T(v)$. Thus, vertex v should be at least four edges from the boundary of A_i in each of the tangent directions.

Algorithm EXTENDRELIABLE (Algorithm 6) increases the number of reliable gradients close to the gradient discontinuities. On the other hand, if a cube contains an isosurface corner, then the distance to reliable gradients may be even larger than four. Thus, we look for reliable gradients in a $9 \times 9 \times 9$ region around cube $\mathbf{c}$ or distance 4 from the vertices of $\mathbf{c}$.

We are interested only in selecting gradients which determine isosurface tangent planes near $\mathbf{c}$. We use three tests on vertices in the $9 \times 9 \times 9$ region to select such gradients. First, we are only interested in vertices which are near the isosurface. Thus, we only choose vertices from edges where one endpoint has scalar value below the isovalue and one endpoint has scalar value at or above the isovalue. Second, we are only interested in vertices whose gradients generate planes which are close to $\mathbf{c}$. Let h_v be the tangent plane generated by the gradient at vertex v. We construct a cube $\mathbf{c}'$ of size $1.5 \times 1.5 \times 1.5$ centered at $\mathbf{c}$ and only choose a vertex v if the plane h_v intersects $\mathbf{c}'$. (We use a cube $\mathbf{c}'$ which is slightly larger than $\mathbf{c}$ because noise and approximation errors can cause a tangent plane h_v to slightly miss $\mathbf{c}$.)

Finally, we only want to choose a vertex which is far from $\mathbf{c}$ if a closer vertex is not chosen. Let Q be the set of vertices which do NOT have reliable gradients and are in the $9 \times 9 \times 9$ subgrid centered at $\mathbf{c}$. 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 reliable gradients around cube $\mathbf{c}$. As described in the next section, we use these gradients to determine the locations of isosurface vertices on sharp features.

5 COMPUTING POINTS ON SHARP FEATURES

Once we have reliable gradients, we can use those gradients to compute the location of isosurface vertices on sharp edges and corners. The gradients can also be used to compute the location of isosurface

vertices on smooth regions of the isosurface. The algorithm for computing isosurface vertices from gradients is given in [1]. It is a modification of Lindstrom's algorithm in [16], with surface normals replaced by 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 $h_i = \{x : g_i \cdot (x - p_i) + s_i = \sigma\}$ is a plane in 3D. Equivalently, plane h_i is $\{x : g_i \cdot x = \sigma - (g_i \cdot p_i + s_i)\}$.

Given a set $\{(p_i, g_i, s_i)\}$ of k points and their associated gradients and scalar values, define a matrix M whose i'th row is $g_i/|g_i|$ and a column vector b whose i'th element is $(\sigma - (g_i \cdot p_i + s_i))/|g_i|$. (We divide by $|g_i|$ so that all normal directions have equal weight.) This gives 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 overdetermined so we wish to find the least squares solution.

We use the singular valued decomposition (SVD) of $M^T M$ to find an approximate solution x^* to $Mx = b$. We use the number of large singular values of A to determine whether x^* is on a sharp corner, a sharp edge or a smooth region on the surface. Details are in Appendix B.

6 Sparsifying Isosurface Vertices

The algorithm in the previous section produces one isosurface vertex for each grid cube intersected by the isosurface. Points on sharp edges and corners are identified as such by the algorithm. However, isosurface vertices on sharp features may lie very close together. If those vertices are used for mesh generation or are joined together to form sharp curves, they need to sparsified so there is spacing between them. Fortunately, the grid structure makes this easy. We use a procedure from [1] to sparsify the set of isosurface vertices on sharp features.

Let S be the list of isosurface vertices on sharp features sorted by increasing distance from the grid cube centers. Select the first vertex v in S. Let $\mathbf{c}$ be the cube containing v. Delete any vertices of S which lie in $\mathbf{c}$ or in any of the 26 grid cubes which share a vertex with $\mathbf{c}$. Repeat until S contains no more vertices. If all grid edges have length L, then the resulting isosurface vertices are never closer than distance L. Figure 11(b) shows the original sharp edge vertices and figure 11(c) shows the sparse set for an edge in a real CT dataset.

7 Experimental Results

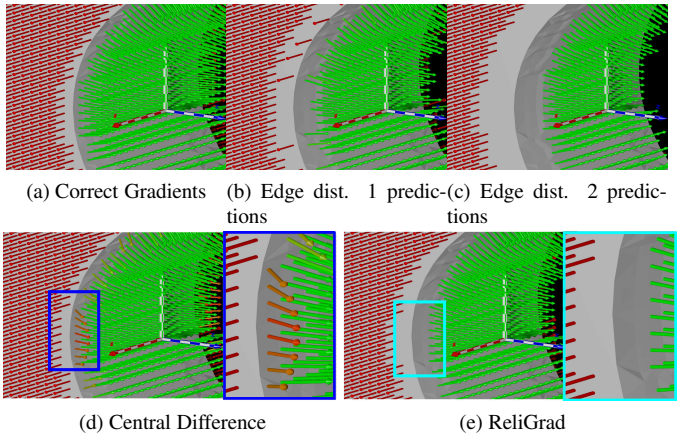
7.1 Synthetic datasets

Name	ASize	Spacing	ARotation
Cube	100	1 1 1	1 1 1
Annulus	100	1 1 1	1 0 0
TwoCube	150	0.2 0.2 0.4	1.732 0.577 0
Flange	150	0.2 0.2 0.4	1.732 0.577 0
Smooth-tip Cone	100	1 1 1	-1 1 1
Cannon	100	1 1 1	-1 1 1

Table 1: Synthetic dataset information, the size of the datasets is $ASize^3$. Each data set is centrally symmetric around the axis given by ARotation. Faces of the *Cube* and *TwoCube* datasets are orthogonal or parallel to the axis given by ARotation.

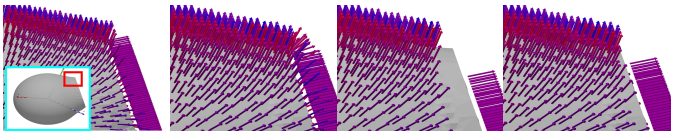
We tested our algorithm on a large number of synthetic datasets with varying spacings and axis rotations. Table 1 and Figure 1 show six representative elements.

Description: The *Cube* dataset samples a scalar field $f :: \mathbb{R}^3 \rightarrow \mathbb{R}$, where $f(p)$ is the minimum of the L_∞ distance to a single point. Tilted cubes are made from orthogonal frames other than the standard x,y and z axis. The isosurfaces in the *TwoCube* dataset contain sharp corners and sharp saddle points. The *Annulus* dataset is minimum of the distance to a cylinder and the distance to a plane orthogonal to the cylinder. By adding constants to these distances, we can create annulli with arbitrary heights and radii. The *Flange* datasets is the minimum of two *Annulus* data sets, one with flat, wide annulii and one with tall thin annulii. Isosurfaces in these datasets are flanges with sharp concave and convex edges. The *Cone* data set is the minimum of the distance to a cone and to an orthogonal plane. The distance to the cone near the tip is the distance to the point at the tip of the cone. The isosurface around the tip is part of a sphere. The *Cone* dataset has acute dihedral angles of 60°. The *Cannon* dataset is the combination of the distance to a cone, the distance to an orthogonal plane and the distance to a point. The *Cannon* dataset has obtuse dihedral angles of 120°. *Flange* and *TwoCube* are challenging datasets, the spacings on these are not uniform and they are not axis aligned. *Cone* and *Cannon* are also not axis aligned (Table1).



(a) Correct Gradients (b) Edge dist. 1 predictions (c) Edge dist. 2 predictions (d) Central Difference (e) ReliGrad

Fig. 7: Gradient results on a part of the *Annulus* dataset (Figure 1(c)). (a) Correct gradients. (b) Gradients marked correct by Algorithm 1 which uses $n_{v''}$ and $n_{v'}$ to predict n_v. (c) Gradients marked correct by Algorithm 2 which uses $n_{v'''}$, $n_{v''}$, and $n_{v'}$ to predict n_v. (d) Central difference gradients. The blue inset on the right contains a magnified view. (e) Gradients marked correct by ReliGrad. The cyan inset shows a magnified view.

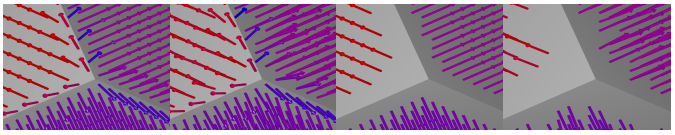


(a) Correct (b) CDiff (c) FindReliable (d) Religrad

Fig. 8: Cannon dataset gradients (Curved edge, non 90° angles). (a) correct gradients, (b) central difference gradients, (c) gradients marked correct by FindReliable. (d) gradients at grid vertices marked correct by ReliGrad. Inaccurate CDiff gradients Figure(b) across the discontinuity are not marked correct, hence not shown.

Reliable Gradients: Figure 7 shows gradients at grid vertices for grid cubes which intersect the isosurface on a zoomed in section of the *Annulus* dataset. To get the colormap for the gradient vectors, we projected all gradient vectors on to the XY plane, the angle of the resulting vector to the X-Axis is mapped between red-green. Gradients close to X-Axis are red, those perpendicular are green. Figure 7(a) shows the known correct gradients at all the grid vertices. Figure 7(d), shows the central difference (CDiff) gradients at all the grid vertices. The blue inset shows a magnified view: CDiff gradients are inaccurate along discontinuities. Figure 7(b) shows gradients only at vertices marked correct by Algorithm 1, which uses $n_{v''}$ and $n_{v'}$ to predict n_v. Figure 7(c) shows results from Algorithm 2, which uses $n_{v'''}$, $n_{v''}$, and $n_{v'}$ to predict n_v. Figure 7(e) shows gradients marked reliable by ReliGrad, the cyan inset shows a close-up, the inaccurate gradients are marked unreliable by ReliGrad and not shown.

Figure 8 shows the gradients around the edge of the *Cannon* dataset (Figure 8(a) red rectangle shows the location). The dataset is not axis-parallel, the dihedral angle around the curved edge is 120°. Figure 8(a) shows known correct gradients at grid vertices which are intersected by the isosurface. Figure 8(b) shows the central difference gradients, at grid vertices of the intersected cubes. Note that along the discon-



(a) (b) (c) (d)

Fig. 9: Effect of uniform noise on *cube* gradients. Uniform 0.1 noise was added to vertex scalars. (a) central difference gradients on the original *cube*, (b) central difference gradients computed on the noisy cube. (c) gradients marked correct by RELIGRAD on the original *cube*. (d) gradients at the vertices marked correct by RELIGRAD on the noisy *cube*.

tinuity the central difference gradients are inaccurate compared to the correct gradients (Figure 8(a)). Figure 8(c) shows gradients at grid vertices marked reliable by FINDRELIABLE. Figure 8(d) shows gradients marked correct by RELIGRAD. Figure 9 shows the gradients at corner of the *cube* dataset. Figure 9(a) shows the central difference gradients. As expected, the gradients across the discontinuity are erroneous. Figure 9(c) shows gradients at grid vertices marked correct by RELIGRAD. The problems of central difference gradients becomes compounded if the scalar values are noisy. We added 0.1 uniform noise to grid vertex scalars of *cube*. Figure 9(b) shows the central difference gradients computed from the noisy scalars, Figure 9(d) shows the result from RELIGRAD.

7.2 Quantitative Analysis

For the synthetic tests cases we know the correct gradients at each grid vertex. Consequently, we can quantitatively compare them with reliable gradients results. Table 2 shows the maximum angle difference

Dataset	CDiff	Algo 1	Algo 2	Find Reliable	Reli Grad
Cube	48.3	0.7	0.0	0.0	0.0
Annulus	44.03	2.79	0.02	0.02	0.02
Flange	45.4	2.1	0.04	0.04	9.8
TwoCube	61.86	6.1	0.0	0.0	19.8
Cannon	29.7	12.3	0.5	0.5	0.5
Cone	57.5	1.6	1.11	1.11	13.1

Table 2: Maximum Angle difference compared to correct gradients in°s. α,α2 is set to 20°. A large number of vertices with high angle difference to the correct gradients, means erroneous gradients are being marked correct. Low angles mean the gradients marked reliable are very close to the correct gradient which is desired outcome.

between the known correct gradients and those computed as reliable gradients at each grid vertex v. Intuitively this captures false positives. Large maximum angle would mean poor gradients are being marked as correct. Algorithm 2 which uses vertices at edge distance 3 for n_v has lower angles than Algorithm 1 for all the test cases thus performing better. Figure 7(b) (Algorithm 1) showed one particular example where gradients near the discontinuity were marked correct compared to (Algorithm 2) Figure 7(c). Central difference as expected generates erroneous gradients near the edges. For the synthetic datasets, FINDRELIABLE, performs similarly to Algorithm 2. RELIGRAD which extends the FINDRELIABLE gradients by using Algorithm 6 generates larger maximum angle than FINDRELIABLE.

Next we look at the maximum of the L1 distances from each vertex v to the closest grid vertex with a reliable gradient. Table 3 shows the results. This test intuitively captures false negatives. Large distances mean more vertices with reliable gradients are being marked unreliable. It is desirable for algorithms which use the reliable gradients results, that the L1 distance of an unreliable vertex v to its closest reliable vertex v_2 be as small as possible. For *Annulus* there is a grid vertex at maximum L1 edge distance of 4 from each v when using Algorithm 2. For *Cube* there is a vertex with correct gradient within L1

Dataset	Algorithm 1	Algorithm 2	RELIGRAD
Annulus	3	4	2
Cube	3	4	3
Flange	4	6	4
Cannon	2	4	2
Cone	3	4	2
TwoCube	4	5	3

Table 3: Maximum of L1 distances to closest grid vertices with exact gradients.

edge distance 4 from each v when using Algorithm 2. When we extend the reliable gradients and use RELIGRAD, the maximum L1 distance decreases to 2 and 3 respectively. Experimentally we found that the distances for *Cube* is maximum near the corners. (See Figure 9(c).)

Noise: To test the effect of noise on RELIGRAD, uniform noise was added to the datasets. Table 4 shows the results. With 0.1 uniform noise RELIGRAD performs well. The maximum L_1 distance to reliable gradients in the *Cube* dataset is 5. This distance was achieved at a vertex near the isosurface corner (Figure 9(d).) With a high uniform noise of 0.2 and α_2 set to 30 degrees, RELIGRAD performs reasonably well.

dataset	noise	α_2 in °	Max AngleDiff in °	Dist 2Grad
cube	0.1	20	7.5	5
annulus	0.1	20	8.1	3
cone	0.1	20	12.7	3
cube	0.2	30	12.8	7
annulus	0.2	30	14.1	5
cone	0.2	30	19	6

Table 4: Results after adding 0.1 and 0.2 uniform noise to the datasets using RELIGRAD. To handle 0.2 noise we set α_2 to be 30°s. MaxAngleDiff, measures the maximum angle difference between known correct gradients and those marked correct by RELIGRAD. Dist2Grad measures the maximum of the l1 distance of the unreliable vertices to vertices marked correct by RELIGRAD.

7.3 Industrial CT data

We also evaluate our algorithm on a number of industrial CT data sets (table 5).

Description: The *CMM* dataset is a CT of solid alumimum shape used to calibrate CT devices. *CMM* stands for "coordinate measuring machine". Figure 15(b) shows an isosurface from this data set. Figure 15(a) shows a single slice along the XY plane. The *Engine Cylinder* dataset is the CT of two motorcycle engine cylinders. Figure 13(a) shows a single slice in an XY plane. The slice exhibits streaking artifacts caused by beam hardening. The *Intake* dataset is a CT of a motorcycle engine intake valve. Figure 17(b) shows an isosurface of dataset, along with a single slice in an XY plane. The *Socket* dataset is a CT of 440 volt converter. Figure 16(c) shows a single slice in the YZ plane using the hsv color scale. Table 5 provides the size and spacing info on all these data sets.

Reliable Gradients Figure 10 shows the central difference gradients at all grid vertex location around an edge of the *engine cylinder*

Name	Axis Size	Spacing
CMM	500 500 196	0.2 0.2 0.31
Engine cylinder	201 130 63	0.27 0.27 0.68
Intake	150 220 201	0.27 0.27 0.68
Socket	411 431 61	1 1 1

Table 5: CT dataset information

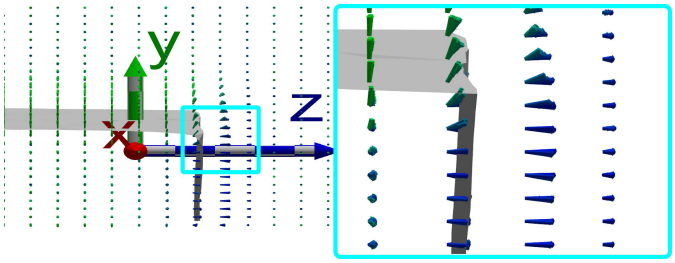
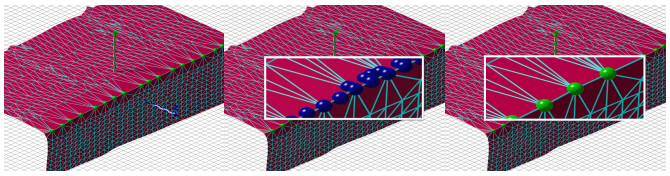


Fig. 10: All the central difference gradients on grid vertices of a small portion of the *engine cylinder* dataset (Figure 3 shows the same regions, but only the gradients at vertices intersected by isosurface). The gradient magnitudes are proportional to length of the vectors. On the right we see a magnified section of the cyan rectangle. The grid magnitudes drop off quickly, and the gradient directions become meaningless especially along the Z-axis.



(a) (b) (c)

Fig. 11: (a) Sharp mesh of a part of (Figure 13). (b) All the sharp edge vertices generated. (c) Sparse sharp vertex set

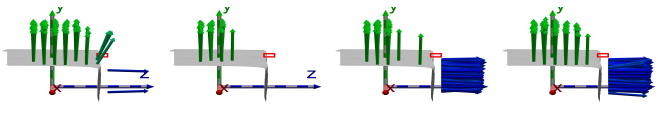
dataset. The length of the vectors are proportional to the magnitude of the gradients. Note that the gradients quickly fall off away from the isosurface. The problem is more severe along the Z-Axis. The magnified portion shows that there are about two good columns of gradients after which the gradient magnitude becomes too small and the gradient directions meaningless. Figure 12(a), 12(b) show the results of Algorithm 1 and Algorithm 2 respectively. Unlike the simulated datasets, almost no gradients along the Z axis are marked as reliable. This shows the need for dividing the vertex set into tangential and orthogonal neighbor sets. This lead us to Algorithm 7. Figure 12(c) shows the corresponding gradients of the FINDRELIABLE technique. Figure 12(d) shows the RELIGRAD gradients. For reference, Figure 3 shows the Central Difference gradients.

7.4 Visualization

7.4.1 Sparse feature point detection

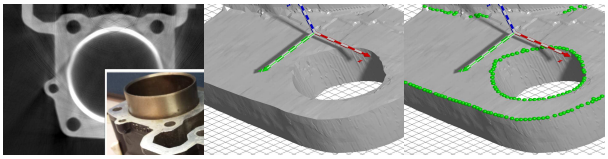
From the reliable gradients, we can select a subset in the neighborhood of each cube (Section 4), compute points on the sharp features (Section 5), and select a well-spaced subset of those points (Section 6).

Figure 11(b) shows the points on a portion of the *engine cylinder* dataset. Figure 11(c) shows a well-spaced subset of those points.



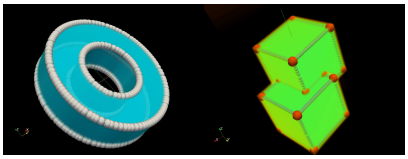
(a) Algorithm 1 (b) Algorithm 2 (c) FindReliable (d) ReliGrad

Fig. 12: The gradients generated on a portion of the *engine cylinder* dataset. The red rectangle shows the size of a grid cube with spacing (0.27, 0.27, 0.68). (a) Gradients marked reliable by Algorithm 1. (b) Gradients marked reliable by Algorithm 2. (c) Gradients marked reliable by FINDRELIABLE (Algorithm 5). (d) Gradients marked reliable by RELIGRAD.



(a) Single image slice, original machine part (b) Sharp Isosurface (c) Sparse feature points

Fig. 13: *Engine Cylinder* dataset. (a) a slice of the original CT image, note the streaking artifacts introduced during the scanning process. The inset shows the original machine part. Figure (b), (c), shows the sharp mesh, and the extracted sparse feature points.



(a) Annulus (b) Twocube

Fig. 14: Integrating results into ParaView [5]. Volume rendering of the Annulus and TwoCube dataset, overlayed with sparse feature skeleton.

7.4.2 Point Skeleton Visualization

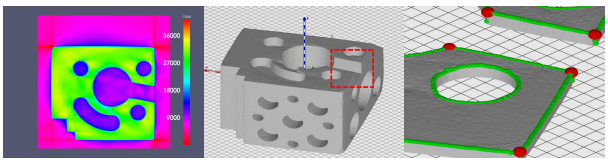
The sparse feature vertices are used to create a sharp feature point skeleton. The point set skeleton is visualized for the synthetic dataset in Figure 1. The number of large Eigenvalues determines if the sharp feature is on an edge or on a corner. Points on the edges are marked in green and corner points in red. The skeleton is correctly extracted on challenging non-axis aligned datasets. We also do well on acute and obtuse angles (Figure 1(e), 1(d)).

Figure 13(c) shows the extracted sparse point skeleton of a part of the *engine cylinder* dataset. Figure 15(c) shows the point skeleton the CMM dataset. Figure 16(b) shows the point skeleton of the *socket* dataset.

Apart from direct point visualization, The extracted feature points can be quickly integrated into commonly used frameworks. For example, Figure 14(a), 14(b) shows the volume rendering of the datasets using ParaView [5], overlay-ed with the detected edge and corner points. The simple addition of the edge points makes the rendering more comprehensible.

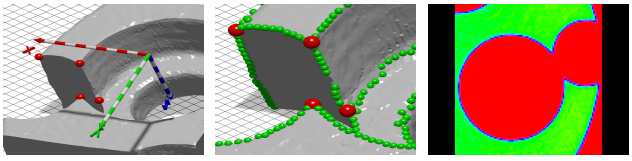
7.5 Isosurface Reconstruction

The point skeleton generated by our algorithm can be input to mesh generation algorithms to construct an isosurface mesh. All the meshes in Figure 1 are generated using MERGESHARP applied to the sharp points from our algorithm. Figure 13b shows the extracted isosurface of a part of the *engine cylinder* dataset. Figure 15(b) shows the



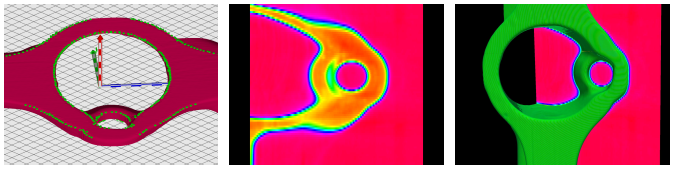
(a) Single XY slice, hsv color scale. Intensity range is 0-41000 (b) MergeSharp Surface (c) Close Up of red square

Fig. 15: CMM dataset, (a) single slice XY plane. (b) MergeSharp isosurface. (c) magnified red rectangle. Table 5 provides size and spacing.



(a) Socket (b) Socket Close-up (c) XY Slice

Fig. 16: (a) sharp mesh of a part of the *socket* dataset. (b) Close up. Corners are marked in red, sparse selected edge vertices are marked in green. (c) XY slice (hsv color scale, ParaView).



(a) intake valve (b) Slice (c) Context

Fig. 17: (a) sharp surface and point skeleton of the socket dataset. (b) a single slice of the YZ axis using the hsv color scale. (c) the slice along with the dataset, to provide context.

isosurface generated from the CMM data set. Figure 15(c) shows close up of the red rectangle in Figure 15b. The corner vertices are marked in red and the sparse edge vertices are marked in green. Figure 16(a), 16(b) shows an extracted isosurface of the socket dataset. Figure 17(a) shows a MERGESHARP extracted mesh along with the sparse feature point skeleton of the *intake* dataset.

The algorithm, WeightedCocone, by Dey et al. [9] constructs a surface mesh from a point cloud which includes as input an identified set of points on sharp features. We applied WeightedCocone on the point skeleton from the flange dataset. We also supplied sample points on the smooth parts of the surface. Figure 18(a) shows the sparse point skeleton and the resulting surface. Other techniques which take as input feature curves/point can also potentially use our results.

7.6 Parameters

Algorithm RELIGRAD has three parameters. Among these α_2 is the major parameter. α_2 is a bound on the error in our prediction of a gradient at a grid vertex against the actual gradient at the vertex location. This includes the inherent noise in the data and the change in curvature. Table 6 shows the effect of changing α_2 on maxAngleDiff and

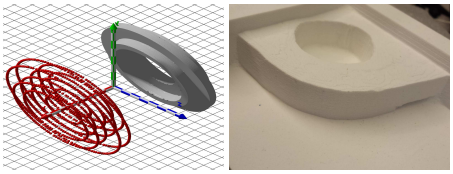
α_2 in $^\circ$	Flange		TwoCube	
	MaxAngleDiff in $^\circ$s	Dist 2Grad	MaxAngleDiffin $^\circ$s	Dist 2Grad
5	4.11	5	3.3	5
10	9.8	5	8.0	5
15	14.2	4	14.4	4
20	19.7	4	19.2	3
25	21.9	4	37.3	3
30	25.0	4	73.1	3

Table 6: Effect of changing parameter α_2 in RELIGRAD.

Dist2Grad on the *flange* and *TwoCube* dataset. As expected, the maximum angle difference to the known correct gradients increases with increasing α_2. While the maximum angle is 73.1 degrees, for *TwoCube* with α_2 set to 30 degrees, there were only 2 vertices with angle greater than 30 degrees. The MERGESHARP reconstruction showed no visible errors. α_1 decides if gradients at two vertices agree with each other. This is a much tighter threshold. For all our experiments this was set to 5°s. We also always fixed the numIter, which decides how many times the reliable gradients are extended to 2, for all our tests. The time to run the algorithm on all the datasets is in seconds.

8 CONCLUSION

We showed that sharp features can be extracted directly from industrial CT scalar data, without extra information. This is a major improvement over previous algorithms which require exact surface normals or gradients. We stress on this being a complete framework, starting directly from the CT scan, to constructing isosurface meshes with good representation of sharp edges and corners. Figure 18(b) shows a 3D printing of a part of the *cylinder engine* dataset from the mesh extracted in Figure 13(b).



(a) (b)

Fig. 18: 18(a) shows the result of SingularCocone using the sparse selected vertices as input and the corresponding output mesh on the Flange dataset. Figure 18(b) shows a 3D print of Figure 13(b).

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] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or. L 1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG)*, 29(5):135, 2010.

[5] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware Clifton Park, NY, 2015.

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

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

[8] 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.

[9] 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.

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

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

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

[15] 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.

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

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

[18] 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.

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

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

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

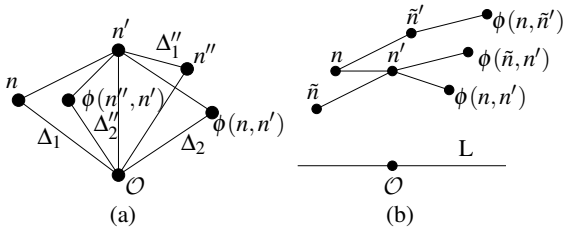


Fig. 19: (a) Triangles Δ_1, Δ_2, Δ_1'' and Δ_2''. (b) Rotating n and $\tilde{n}'$ around line L.

A PROOFS

A.1 Angle Lemmas

Let n and n' be unit vectors in $\mathbb{R}^3$. Recall the following definitions, from Section 3.3.

$$\begin{aligned}\text{Orth}(n,n') &= n-(n\cdot n')n'\\ \phi(n,n') &= n-2\times\text{Orth}(n,n') = 2(n\cdot n')n'-n.\end{aligned}$$

Vector $\text{Orth}(n,n')$ is the component of n orthogonal to n. Vector $\phi(n,n')$ is the vector *predicted by* n and n'.

$$\begin{aligned}\cos(\angle(\phi(n,n'),n')) &= (2(n\cdot n')n'-n)\cdot n'\\ &= 2(n\cdot n')(n'\cdot n')-(n\cdot n')\\ &= 2(n\cdot n')-(n\cdot n') \text{ (since } n' \text{ is a unit vector)}\\ &= n\cdot n' = \cos(\angle(n,n')).\end{aligned}$$

Thus, $\angle(\phi(n,n'),n')$ equals $\angle(n,n')$.

Lemma 4. *If n, n' and n'' are unit vectors, then*

1. $\Lambda(n,n',n'') = \Lambda(n'',n',n)$*, and*
2. $\angle(n,n'') = \angle(\phi(n,n'),\phi(n'',n'))$.

Proof. Let $\mathcal{O}$ be the origin $(0,0,0)$. Let Δ_1, Δ_2, Δ_1'' and Δ_2'' be the triangles $\Delta(\mathcal{O},n',n)$, $\Delta(\mathcal{O},n',\phi(n,n'))$, $\Delta(\mathcal{O},n',n'')$ and $\Delta(\mathcal{O},n',\phi(n'',n'))$, respectively. (See Figure 19.(a).)

As noted above, $\angle(n,n')$ equals $\angle(\phi(n,n'),n')$. Similarly, $\angle(n',n'')$ equals $\angle(\phi(n'',n'),n')$. Since n, n' and $\phi(n,n')$ have the same (unit) length, triangles Δ_1 and Δ_2 are congruent. Since n', n'' and $\phi(n'',n')$ have the same (unit) length, triangles Δ_1'' and Δ_2'' are congruent.

Since n, n' and $\phi(n,n')$ are co-planar and n'', n' and $\phi(n'',n')$ are co-planar, the dihedral angles between Δ_1 and Δ_2'' equals the dihedral angle between Δ_2 and Δ_1''. Thus, tetrahedron $(\mathcal{O},n,n',\phi(n'',n'))$ is congruent to tetrahedron $(\mathcal{O},n',n'',\phi(n,n'))$ and $\angle(n,\phi(n'',n'))$ equals $\angle(n'',\phi(n,n'))$ and $\angle(n,n'')$ equals $\angle(\phi(n,n'),\phi(n'',n'))$. □

The following corollary describes how $\angle(\phi(n,n'))$ and $\Lambda(n,n',n'')$ change when unit vector n is replaced by $\tilde{n}$.

Corollary 4.1. *If n, $\tilde{n}$, n' and n'' are unit vectors and $\angle(n,\tilde{n}) \le \mu$, then*

1. $\angle(\phi(n,n'),\phi(\tilde{n},n')) \le \mu$*;*
2. $|\Lambda(n,n',n'') - \Lambda(\tilde{n},n',n'')| \le \mu$.

Proof. By Lemma 4, angle $\angle(n,\tilde{n})$ equals $\angle(\phi(n,n'),\phi(\tilde{n},n'))$. (Let n'' represent $\tilde{n}$ in Lemma 4.) Thus,

$$\begin{aligned}\angle(\phi(n,n'),\phi(\tilde{n},n')) &= \angle(n,\tilde{n}) \le \mu, \qquad \text{and}\\ |\Lambda(n,n',n'') - \Lambda(\tilde{n},n',n'')| &= |\angle(\phi(n,n'),n'') - \angle(\phi(\tilde{n},n'),n'')|\\ &\le \angle(\phi(n,n'),\phi(\tilde{n},n')) \le \mu.\end{aligned}$$

□

In the following corollary, vector n' is replaced by $\tilde{n}'$. The bounds are slightly different than the bounds in the previous corollary.

Corollary 4.2. *If n, n', $\tilde{n}'$, n'' and n''' are unit vectors and $\angle(n',\tilde{n}') \leq \mu$ and $\angle(\phi(n,n'),n'') \leq \gamma$, then*

1. $\angle(\phi(n,n'),\phi(n,\tilde{n}')) \leq 2\mu$;
2. $|\Lambda(n,n',n'') - \Lambda(n,\tilde{n}',n'')| \leq 2\mu$;
3. $|\Lambda(n',n'',n''') - \Lambda(n',\phi(n,n'),n''')| \leq 2\gamma$.

Proof. Let L be the line through the origin orthogonal to n' and $\tilde{n}'$. Rotating $\tilde{n}'$ and n by angle $\angle(\tilde{n}',n')$ around L maps $\tilde{n}'$ to n' and n to a unit vector $\tilde{n}$ where $\angle(n,\tilde{n}) \leq \mu$ (Figure 19.(b).) By Corollary 4.1, $\angle(\phi(n,n'),\phi(\tilde{n},n')) \leq \mu$. Rotating $\tilde{n}$, n' and $\phi(\tilde{n},n')$ by angle $-\angle(\tilde{n}',n')$ around L maps $\tilde{n}$ back to n, vector n' to $\tilde{n}'$ and $\phi(\tilde{n},n')$ to $\phi(n,\tilde{n}')$. Since $|-\angle(\tilde{n}',n')| \leq \mu$, angle $\angle(\phi(\tilde{n},n'),\phi(n,\tilde{n}')) \leq \mu$. Thus,

$$\begin{aligned}\angle(\phi(n,n'),\phi(n,\tilde{n}')) &\leq \angle(\phi(n,n'),\phi(\tilde{n},n')) + \angle(\phi(\tilde{n},n'),\phi(n,\tilde{n}')) \\ &\leq 2\mu,\end{aligned}$$

and

$$\begin{aligned}|\Lambda(n,n',n'') - \Lambda(n,\tilde{n}',n'')| &= |\angle(n'',\phi(n,n')) - \angle(n'',\phi(n,\tilde{n}'))| \\ &\leq \angle(\phi(n,n'),\phi(n,\tilde{n}')) \leq 2\mu.\end{aligned}$$

Replacing n, n', $\tilde{n}'$, n'' and μ by n', n'', $\phi(n,n')$, n''' and γ in the formula above, gives

$$|\Lambda(n',n'',n''') - \Lambda(n',\phi(n,n'),n''')| \leq 2\gamma.$$

□

Lemma 5. *If n, n' and n'' are unit vectors, then*

1. $\phi_2(n,n') = \phi_1(n',\phi_1(n,n'))$, *and*
2. $\Lambda_2(n,n',n'') = \Lambda_1(n',\phi_1(n,n'),n''))$.

Proof. By definition,

$$\phi_2(n,n') = \phi_1(\phi_0(n,n'),\phi_1(n,n')) = \phi_1(n',\phi_1(n,n')).$$

Thus,

$$\begin{aligned}\Lambda_2(n,n',n'') &= \angle(\phi_2(n,n'),n'') \\ &= \angle(\phi_1(n',\phi_1(n,n')),n'') = \Lambda_1(n',\phi_1(n,n'),n'').\end{aligned}$$

□

Corollary 5.1. *If n, n' and n'' are unit vectors and $\angle(n,\tilde{n}) \leq \mu$, then $|\Lambda_2(n,n',n'') - \Lambda_2(\tilde{n},n',n'')| \leq 2\mu$.*

Proof. By Lemma 4,

$$\angle(\phi_1(n,n'),\phi_1(\tilde{n},n')) = \angle(n,\tilde{n}) \leq \mu.$$

Thus,

$$\begin{aligned}\Lambda_2(n,n',n'') &= \Lambda_1(n',\phi(n,n'),n'') && \text{(Lemma 5)} \\ &\leq \Lambda_1(n',\phi(\tilde{n},n'),n'') + 2\mu && \text{(Corollary 4.2)} \\ &= \Lambda_2(\tilde{n},n',n'') + 2\mu.\end{aligned}$$

Swapping n and $\tilde{n}$ gives $\Lambda_2(\tilde{n},n',n'') \leq \Lambda_2(n,n',n'') + 2\mu$. Thus, $|\Lambda_2(n,n',n'') - \Lambda_2(\tilde{n},n',n'')| \leq 2\mu$. □

Corollary 5.2. *If n, n' and n'' are unit vectors and $\angle(n',\tilde{n}') \leq \mu$, then $|\Lambda_2(n,n',n'') - \Lambda_2(n,\tilde{n}',n'')| \leq 3\mu$.*

Proof. By Lemma 4,

$$\angle(\phi_1(n,n'),\phi_1(n,\tilde{n}')) = \angle(n',\tilde{n}') \leq \mu.$$

Thus,

$$\begin{aligned}\Lambda_2(n,n',n'') &= \Lambda_1(n',\phi(n,n'),n'') && \text{(Lemma 5)} \\ &\leq \Lambda_1(\tilde{n}',\phi(n,n'),n'') + \mu && \text{(Corollary 5.1)} \\ &\leq \Lambda_1(\tilde{n}',\phi(n,\tilde{n}'),n'') + 2\mu + \mu && \text{(Corollary 4.2)} \\ &= \Lambda_2(n,\tilde{n}',n'') + 3\mu.\end{aligned}$$

Swapping n' and $\tilde{n}'$ gives $\Lambda_2(n,n',n'') \leq \Lambda_2(n,\tilde{n}',n'') + 3\mu$. Thus, $|\Lambda_2(n,\tilde{n}',n'') - \Lambda_2(n,\tilde{n}',n'')| \leq 3\mu$.

□

Corollary 5.3. *If n, n', n'' and n''' are unit vectors and $\Lambda(n,n',n'') \leq \gamma$ and $\Lambda(n',n'',n''') \leq \gamma$, then $\Lambda_2(n,n',n''') \leq 3\gamma$.*

Proof. By assumption, $\Lambda_1(n,n',n'') \leq \gamma$, so $\angle(\phi_1(n,n'),n'') \leq \gamma$.

$$\begin{aligned}\Lambda_2(n,n',n''') &= \Lambda_1(n',\phi_1(n,n'),n'') && \text{(Lemma 5)} \\ &\leq \Lambda_1(n',n'',n''') + 2\gamma && \text{(Corollary 4.2)} \\ &\leq 3\gamma.\end{aligned}$$

□

A.2 Angle bounds

We now give bounds on Λ and on the angles between the exact and approximated normals.

Proposition 1. *Let (v,v',v'') be a collinear sequence of adjacent vertices contained in A_i for some smooth region A_i.*

1. *If $v,v',v'' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then* $\Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) \leq \Lambda(n_v,n_{v'},n_{v''}) + 4\mu \leq \lambda + 4\mu$.
2. *If $v,v' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then $\angle(n_{v''},\tilde{n}_{v''}) \leq \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) + 3\mu + \lambda$.*

Proof of 1.1. Since v, v' and v'' are interior vertices, angles $\angle(n_v,\tilde{n}_v)$, $\angle(n_{v'},\tilde{n}_{v'})$ and $\angle(n_{v''},\tilde{n}_{v''})$ are all less than μ.

$$\begin{aligned}\Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) &\leq \Lambda(\tilde{n}_v,\tilde{n}_{v'},n_{v''}) + \mu \\ &\leq \Lambda(n_v,\tilde{n}_{v'},n_{v''}) + \mu + \mu && \text{(Corollary 4.1)} \\ &\leq \Lambda(n_v,n_{v'},n_{v''}) + 2\mu + \mu + \mu && \text{(Corollary 4.2)} \\ &= \Lambda(n_v,n_{v'},n_{v''}) + 4\mu \leq \lambda + 4\mu.\end{aligned}$$

□

Proof of 1.2.

$$\begin{aligned}\angle(n_{v''},\tilde{n}_{v''}) &\leq \angle(\phi(n_v,n_{v'}),n_{v''}) + \angle(\phi(n_v,n_{v'}),\tilde{n}_{v''}) \\ &\leq \lambda + \Lambda(n_v,n_{v'},\tilde{n}_{v''}).\end{aligned}$$

Since $\angle(n_v,\tilde{n}_v) \leq \mu$,

$$\Lambda(n_v,n_{v'},\tilde{n}_{v''}) \leq \Lambda(\tilde{n}_v,n_{v'},\tilde{n}_{v''}) + \mu. \qquad \text{(Corollary 4.1)}$$

Since $\angle(n_{v'},\tilde{n}_{v'}) \leq \mu$,

$$\Lambda(\tilde{n}_v,n_{v'},\tilde{n}_{v''}) \leq \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) + 2\mu. \qquad \text{(Corollary 4.2)}$$

Thus,

$$\begin{aligned}\Lambda(n_v,n_{v'},\tilde{n}_{v''}) &\leq \Lambda(\tilde{n}_v,n_{v'},\tilde{n}_{v''}) + \mu \leq \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) + 3\mu, \text{ and} \\ \angle(n_{v''},\tilde{n}_{v''}) &\leq \lambda + \Lambda(n_v,n_{v'},\tilde{n}_{v''}) \leq \lambda + \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) + 3\mu \\ &= \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) + 3\mu + \lambda.\end{aligned}$$

□

The following corollary follows directly from Proposition 1.1.

Corollary 5.4. *Let (v,v',v'') be a collinear sequence of adjacent grid vertices where vertices $v,v',v'' \in A_1$ and v' and v'' are interior vertices in A_1. If $\Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v''}) \leq \alpha$, then*

$$\angle(n_{v''},\tilde{n}_{v''}) \leq \alpha + 3\mu + \lambda.$$

The proofs of Proposition 3 are similar to the proofs of Proposition 1.

Proposition 3. *Let (v,v',v'',v''') be a collinear sequence of adjacent vertices contained in A_i for some smooth region A_i.*

1. *If $v,v',v'',v''' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then*
   $\Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) \leq \Lambda_2(n_v,n_{v'},n_{v'''}) + 4\mu \leq 3\lambda + 6\mu$.
2. *If $v,v' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then*
   $\angle(n_{v'''},\tilde{n}_{v'''}) \leq \Lambda(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) + 3\lambda + 5\mu$.

Proof of 3.1. Since v, v', v'' and v''' are interior vertices, angles $\angle(n_v,\tilde{n}_v)$, $\angle(n_{v'},\tilde{n}_{v'})$, $\angle(n_{v''},\tilde{n}_{v''})$ and $\angle(n_{v'''},\tilde{n}_{v'''})$ are all less than μ.

$$\begin{aligned}
\Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) &\leq \Lambda_2(\tilde{n}_v,\tilde{n}_{v'},n_{v'''}) + \mu \\
&\leq \Lambda_2(n_v,\tilde{n}_{v'},n_{v'''}) + \mu + 2\mu && \text{(Corollary 5.1)} \\
&\leq \Lambda_2(n_v,n_{v'},n_{v'''}) + 3\mu + 2\mu + \mu && \text{(Corollary 5.2)} \\
&= \Lambda_2(n_v,n_{v'},n_{v'''}) + 6\mu \\
&\leq 3\lambda + 6\mu. && \text{(Corollary 5.3)}
\end{aligned}$$

□

Proof of 3.2.

$$\begin{aligned}
\angle(n_{v'''},\tilde{n}_{v'''}) &\leq \angle(\phi_2(n_v,n_{v'}),n_{v'''}) + \angle(\phi_2(n_v,n_{v'}),\tilde{n}_{v'''}) \\
&= \Lambda_2(n_v,n_{v'},n_{v'''}) + \angle(\phi_2(n_v,n_{v'}),\tilde{n}_{v'''}) \\
&\leq 3\lambda + \Lambda_2(n_v,n_{v'},\tilde{n}_{v'''}). && \text{(Lemma 5)}
\end{aligned}$$

Since $\angle(n_v,\tilde{n}_v) \leq \mu$,

$$\Lambda_2(n_v,n_{v'},\tilde{n}_{v'''}) \leq \Lambda_2(\tilde{n}_v,n_{v'},\tilde{n}_{v'''}) + 2\mu. \qquad \text{(Corollary 5.1)}$$

Since $\angle(n_{v'},\tilde{n}_{v'}) \leq \mu$,

$$\Lambda_2(\tilde{n}_v,n_{v'},\tilde{n}_{v'''}) \leq \Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) + 3\mu. \qquad \text{(Corollary 5.2)}$$

Thus,

$$\begin{aligned}
\Lambda_2(n_v,n_{v'},\tilde{n}_{v'''}) &\leq \Lambda_2(\tilde{n}_v,n_{v'},\tilde{n}_{v'''}) + 2\mu \leq \Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) + 5\mu, \text{ and} \\
\angle(n_{v'''},\tilde{n}_{v'''}) &\leq 3\lambda + \Lambda_2(n_v,n_{v'},\tilde{n}_{v'''}) \leq 3\lambda + \Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) + 5\mu \\
&= \Lambda_2(\tilde{n}_v,\tilde{n}_{v'},\tilde{n}_{v'''}) + 3\lambda + 5\mu.
\end{aligned}$$

□

We also give bounds on $\angle(\tilde{n}_v,\tilde{n}_{v'})$ and on $\angle(n_{v'},\tilde{n}_{v'})$. We first define a bound κ on $\angle(n_v,n_{v'})$ in smooth regions of f.

$$\kappa = \max\{\angle(n_v,n_{v'}) v,v' \in A_i \text{ for some } A_i\}.$$

Value κ can be viewed as a bound on the curvature of the field within smooth regions. In comparison, λ is a bound on the change in curvature of the field within smooth regions.

The following proposition gives bounds on $\angle(n_{v'},\tilde{n}_{v'})$ based on $\angle(\tilde{n}_v,\tilde{n}_{v'})$.

Proposition 6. *Let v and v' be adjacent vertices contained in A_i for some smooth region A_i.*

1. *If $v,v' \in \mathcal{I}_{\mathcal{V}}(A_i)$, then $\angle(\tilde{n}_v,\tilde{n}_{v'}) \leq \angle(n_v,\tilde{n}_{v'}) + 2\mu \leq \kappa + 2\mu$.*
2. *If $v \in \mathcal{I}_{\mathcal{V}}(A_i)$, then $\angle(n_{v'},\tilde{n}_{v'}) \leq \angle(\tilde{n}_v,\tilde{n}_{v'}) + \kappa + \mu$.*

Proof of 6.1.

$$\begin{aligned}
\angle(\tilde{n}_v,\tilde{n}_{v'}) &\leq \angle(n_v,\tilde{n}_{v'}) + \angle(n_v,\tilde{n}_v) \leq \angle(n_v,\tilde{n}_{v'}) + \mu \\
&\leq \angle(n_v,n_{v'}) + \angle(n_{v'},\tilde{n}_{v'}) + \mu \\
&\leq \angle(n_v,n_{v'}) + \mu + \mu = \angle(n_v,n_{v'}) + 2\mu \leq \kappa + 2\mu.
\end{aligned}$$

□

Proof of 6.2.

$$\begin{aligned}
\angle(n_{v'},\tilde{n}_{v'}) &\leq \angle(n_v,\tilde{n}_{v'}) + \angle(n_v,n_{v'}) \leq \angle(n_v,\tilde{n}_{v'}) + \kappa \\
&\leq \angle(\tilde{n}_v,\tilde{n}_{v'}) + \angle(n_v,\tilde{n}_v) + \kappa \leq \angle(\tilde{n}_v,\tilde{n}_{v'}) + \mu + \kappa.
\end{aligned}$$

□

A.3 Containment bounds

To prove Proposition 2, we show that if a sufficiently large ball $\mathbb{B}$ contains vertex v, then $\mathbb{B}$ contains $N(v'')$ and $N(v''')$ for some collinear sequence (v,v',v'',v''').

Lemma 7. *Let Γ be a regular grid whose edges all have the same length L. If some ball $\mathbb{B}$ of radius $(5/2)\sqrt{3}L$ contains grid vertex v, then there is a collinear sequence (v,v',v'',v''') of adjacent grid vertices such that $\mathbb{B}$ contains $N(v'')$ and $N(v''')$.*

Proof. Let $\mathbb{B}$ be a ball of radius $(5/2)\sqrt{3}L$ containing grid vertex v. Let q be the center of $\mathbb{B}$. Without loss of generality assume vertex v is at the origin $(0,0,0)$, that the edge length L equals 1, and that q lies in the positive x, y and z orthant. Let $D \leq (5/2)\sqrt{3}$ be the distance from the origin to q. The coordinates of q can be expressed as $D(u_x,u_y,u_z)$ where (u_x,u_y,u_z) is a unit vector.

Without loss of generality, assume that q is closest to the x-axis, i.e., $u_x \geq u_y$ and $u_x \geq u_z$. Let v'' be $(2,0,0)$ and v''' be $(3,0,0)$. We claim that $N_{v''}$ and $N_{v'''}$ are in $\mathbb{B}$.

Let p be the grid vertex with coordinates $(3,-1,0)$. Point p is in $N_{v''}$. Let R equal $(5/2)\sqrt{3}$. Since ball $\mathbb{B}$ contains the origin, distance D is at most R.

$$\begin{aligned}
|q-p|^2 &= (Du_x-3)^2 + (Du_y+1)^2 + (Du_z)^2 \\
&= D^2u_x^2 - 6Du_x + 9 + D^2u_y^2 + 2Du_y + 1 + D^2u_z^2 \\
&= D^2(u_x^2+u_y^2+u_z^2) + 10 - 6Du_x + 2Du_y \\
&= D^2 + 10 - 6Du_x + 2Du_y \\
&\leq D^2 + 10 - 4Du_x \text{ (since } u_y \leq u_x) \\
&= (R+D-R)^2 + 10 - 4(R+D-R)u_x \\
&= R^2 + 2R(D-R) + (D-R)^2 + 10 - 4Ru_x - 4(D-R)u_x \\
&= R^2 + (10-4Ru_x) + (D-R)(2R+D-R-4u_x) \\
&= R^2 + (10-4Ru_x) - (R-D)(D+R-4u_x).
\end{aligned}$$

Since $u_x \geq u_y$ and $u_x \geq u_z$, coordinate u_x is at least $1/\sqrt{3}$.

$$\begin{aligned}
10 - 4Ru_x &= 10 - 4(5/2)\sqrt{3}u_x \\
&\leq 10 - 10\sqrt{3}(1/\sqrt{3}) \text{ since } u_x \geq 1/\sqrt{3} \\
&= 10 - 10 = 0.
\end{aligned}$$

$$D + R - 4u_x \geq R - 4 = (5/2)\sqrt{3} - 4 \geq 4.3 - 4 > 0.$$

Since $\mathbb{B}$ contains the origin, distance D is at most R and $(R-D) \geq 0$. Thus,

$$\begin{aligned}
|q-p|^2 &= R^2 + (10-4Ru_x) - (R-D)(D+R-4u_x) \\
&\leq R^2 + 0 - (R-D)(D+R-4u_x) \text{ (since } (10-4Ru_x) \leq 0) \\
&\leq R^2 \text{ (since } (R-D) > 0 \text{ and } (D+R-4u_x) > 0.)
\end{aligned}$$

so $\mathbb{B}$ contains $(3,-1,0)$.

A similar analysis shows that $\mathbb{B}$ contains $(3,0,-1)$, $(3,1,0)$, $(3,0,1)$, $(2,0,0)$ and $(4,0,0)$. Thus, $\mathbb{B}$ contains $N(v''')$.

A similar analysis shows that $\mathbb{B}$ contains $N(v'')$. □

The bound $(5/2)\sqrt{3}L$ is tight. If we place a ball $\mathbb{B}$ of radius $(2.5-\varepsilon)\sqrt{3}L$ at the point $(2.5-\varepsilon,2.5-\varepsilon,2.5-\varepsilon)L$, then ball $\mathbb{B}$ contains the origin. However, ball $\mathbb{B}$ does not contain $N(v'')$ or $N(v''')$ for any collinear sequence (v,v',v'',v''').

As before, $\mathcal{X}=\cup_{A_j}\partial A_j$ is the union of all the boundaries of smooth regions A_j. Proposition 2 follows directly from Lemma 7.

Proposition 2. *Let Γ be a regular grid whose edges all have the same length L. If some ball $\mathbb{B}$ of radius $(5/2)\sqrt{3}L$ contains grid vertex $v\in A_i$ and does not intersect $\mathcal{X}$, then there is a collinear sequence (v,v',v'',v''') of adjacent grid vertices such that $v''\in\mathcal{I}_{\mathcal{V}}(A_i)$ and $v'''\in\mathcal{I}_{\mathcal{V}}(A_i)$.*

Proof. By Lemma 7, ball $\mathbb{B}$ contains $N(v'')$ and $N(v''')$ for some collinear sequence (v,v',v'',v'''). Since ball $\mathbb{B}$ does not intersect $\mathcal{X}$,

$$N(v'')\subseteq\mathbb{B}\subseteq A_i, \text{ and}$$
$$N(v''')\subseteq\mathbb{B}\subseteq A_i.$$

Thus, $v''\in\mathcal{I}_{\mathcal{V}}(A_i)$ and $v'''\in IV(A_i)$. □

We can also give a reformulation of Lemma 7 to handle the tangent and orthogonal directions separately.

Lemma 8. *Let Γ be a regular grid whose edges all have the same length L. If some ball $\mathbb{B}$ of radius 7L contains grid vertex v, then either there is a collinear sequence (v,v',v'',v''') of adjacent grid vertices such that $v'\in N^T(v)$ and $\mathbb{B}$ contains $N(v'')$ and $N(v''')$ or there is a vertex $v'\in N^O(v)$ such that $\mathbb{B}$ contains $N(v')$.*

The proof is similar to the proof of Lemma 7, and is omitted.

The radius $7L$ is an upper bound on $(\sqrt{179}/2)L$ which is tight. If we place a ball $\mathbb{B}$ of radius $((\sqrt{179}/2)-\varepsilon)L$ at the point $(3.5-\varepsilon,3.5-\varepsilon,3.5-\varepsilon)L$, then ball $\mathbb{B}$ contains the origin. However, ball $\mathbb{B}$ does not contain $N(v'')$ or $N(v''')$ for any collinear sequence (v,v',v'',v''') where v' equals $(1,0,0)$ or $(-1,0,0)$ or $(0,1,0)$ or $(0,-1,0)$. Ball $\mathbb{B}$ also does not contain $N(v')$ for v' equal to $(0,0,1)$ or $(0,0,-1)$.

Let A_j be the smooth region containing vertex v. Let $\mathcal{X}=\cup_{A_j}\partial A_j$ be the union of all the boundaries of smooth regions A_j. Lemma 8 leads to the following proposition.

Proposition 9. *Let Γ be a regular grid whose edges all have the same length L. If some ball $\mathbb{B}$ of radius 7L contains grid vertex $v\in A_i$ and does not intersect $\mathcal{X}$, then either there is a collinear sequence (v,v',v'',v''') of adjacent grid vertices such that $v'\in N^T(v)$ and $v''\in\mathcal{I}_{\mathcal{V}}(A_i)$ and $v'''\in\mathcal{I}_{\mathcal{V}}(A_i)$ or there is a vertex $v'\in N^O(v)$ such that $\mathbb{B}$ contains $N(v')$.*

Proof. By Lemma 7, either ball $\mathbb{B}$ contains $N(v'')$ and $N(v''')$ for some collinear sequence (v,v',v'',v''') where $v'\in N^T(v)$ or ball $\mathbb{B}$ contains $N(v')$ where $v'\in N^O(v)$.

In the first case,

$$N(v'')\subseteq\mathbb{B}\subseteq A_i, \text{ and}$$
$$N(v''')\subseteq\mathbb{B}\subseteq A_i.$$

Thus, $v''\in\mathcal{I}_{\mathcal{V}}(A_i)$ and $v'''\in IV(A_i)$.

In the second case,

$$N(v')\subseteq\mathbb{B}\subseteq A_i.$$

Thus, $v'\in\mathcal{I}_{\mathcal{V}}(A_i)$. □

A.4 The Central Difference Formula

In the following proposition, we claim that the the central difference formula is exact under appropriate conditions. Let u_d be the vector from each vertex to the adjacent vertex in direction d.

Proposition 10. *Let $f:\mathbb{R}^3\to\mathbb{R}$ be a smooth function. If the second order partial derivatives of f are all constant, then the gradient of f at point x is exactly equal to $(\tilde{g}_1,\tilde{g}_2,\tilde{g}_3)$ where*

$$\tilde{g}_d=(f(x+u_d)-f(x-u_d))/2|u_d|.$$

Proof. Since $\partial^2 f/(\partial(x_d))^2$ is constant, the third order partial derivative $\partial^3 f/(\partial(x_d))^3$ is zero. By Taylor's Theorem,

$$\begin{aligned}f(x+tu_d)&=f(x)+t|u_d|\frac{df(x+tu_d)}{dt}+(t^2/2)|u_d|^2\frac{d^2f(x+tu_d)}{dt^2}\\&\quad+(t^3/6)|u_d|^3\frac{d^3f(x+tu_d)}{dt^3}+\ldots\\&=f(x)+t|u_d|\frac{\partial f}{\partial(x_d)}+(t^2/2)|u_d|^2\frac{\partial^2 f}{\partial(x_d)^2}\\&\quad+(t^3/6)|u_d|^3\frac{\partial^3 f}{\partial(x_d)^3}+\ldots\\&=f(x)+t|u_d|\frac{\partial f}{\partial(x_d)}+(t^2/2)|u_d|^2\frac{\partial^2 f}{\partial(x_d)^2}\\&\quad\text{(since }\partial^3 f/\partial(x_d)^3\text{ is zero.)}\end{aligned}$$

Thus,

$$\begin{aligned}\tilde{g}_d&=\frac{f(x+u_d)-f(x-u_d)}{2|u_d|}\\&=\frac{\left(f(x)+|u_d|\frac{\partial f}{\partial(x_d)}+(1/2)|u_d|^2\frac{\partial^2 f}{\partial(x_d)^2}\right)}{2|u_d|}\\&\quad-\frac{\left(f(x)-|u_d|\frac{\partial f}{\partial(x_d)}+(1/2)|u_d|^2\frac{\partial^2 f}{\partial(x_d)^2}\right)}{2|u_d|}\\&=\frac{2|u_d|\frac{\partial f}{\partial(x_d)}}{2|u_d|}=\frac{\partial f}{\partial(x_d)}.\end{aligned}$$

The gradient of f is is:

$$\left(\frac{\partial f}{\partial(x_1)},\frac{\partial f}{\partial(x_2)},\frac{\partial f}{\partial(x_d)}\right)=(\tilde{g}_1,\tilde{g}_2,\tilde{g}_3)=\tilde{g}.$$

Thus the gradient of f equals $\tilde{g}$. □

B Computing Points on Sharp Features

Given a set $\{(p_i,g_i,s_i)\}$ of k points and their associated gradients and scalar values, define a matrix M whose i'th row is $g_i/|g_i|$ and a column vector b whose i'th element is $(\sigma-(g_i\cdot p_i+s_i))/|g_i|$. (We divide by $|g_i|$ so that all normal directions have equal weight.) This gives 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

$$M^TMx=M^Tb.$$

The 3×3 matrix $A=M^TM$ and the column vector $b'=M^Tb$ gives the quadric error measure.

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

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

σ_1, σ_2, and σ_3 are the singular values of A sorted in decreasing order. If all three singular values of A are large, then the M and b define a sharp corner. If two singular values are large, then M and b define a sharp edge. Otherwise, M and b do not define a sharp feature.

$$\sigma_i' = \begin{cases} \sigma_i & \text{if } \sigma_i/\sigma_1 > \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

where 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 $(\sigma_1^+, \sigma_2^+, \sigma_3^+)$. Let q be a point in $\mathbb{R}^3$. Compute:

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

When A has three large singular values, x is the point solving $Ax = b$. When A has two large singular values, x is a the point closest to q on the line $A'x = b$. When A has only one large singular value, x is a point closest to q on the plane $A'x = b$.

The columns of U are the eigenvectors, u_1, u_2 and u_3 of A' where u_i corresponds to σ_i'. Note that σ_i' are sorted in decreasing order. When A has one large singular value, the plane $\{x : A'x = b\}$ is orthogonal to u_1. When A has two singular values, the direction of the line $\{x : A'x = b\}$ is the cross product, $u_1 \times u_2$, of u_1 and u_2.

We use Equation 2 to determine an isosurface vertex in or near each active cube. To apply Equation 2, we need a point q. The center of the cube is a reasonable choice for q. However, as reported in [21], better results are given by computing an approximate location of the isosurface vertex based on linear interpolation and using this approximate location for q. More specifically, for each bipolar grid edge $\mathbf{c} = (p, p')$, we define:

$$w_{\mathbf{c}} = w_{\mathbf{c}} \leftarrow (1 - \alpha)p + \alpha p'$$

where $\alpha = (\sigma - s_p)/(s_{p'} - s_p)$. Point $w_{\mathbf{c}}$ is an approximation of the intersection of the isosurface and the grid edge.

Instead of the singular valued decomposition of the square matrix $M^T M$, we could have used the singular valued decomposition of M. The condition number of $M^T M$ is the square of the condition number of M so using the SVD of M is more numerically stable. However, because the algorithm sets small eigenvalues to zero, the increased numerical stability does not change the results of our algorithm.