# Polygonizing Extremal Surfaces with Manifold Guarantees

Ruosi Li, Lu Liu, Ly Phan, Sasakthi Abeysinghe, Cindy Grimm, Tao Ju
Washington University, St. Louis, MO 63132 USA

## ABSTRACT

Extremal surfaces are a class of implicit surfaces that have been found useful in a variety of geometry reconstruction applications. Compared to iso-surfaces, extremal surfaces are particularly challenging to construct in part due to the presence of boundaries and the lack of a consistent orientation. We present a novel, grid-based algorithm for constructing polygonal approximations of extremal surfaces that may be open or unorientable. The algorithm is simple to implement and applicable to both uniform and adaptive grid structures. More importantly, the resulting discrete surface preserves the structural property of the extremal surface in a grid-independent manner. The algorithm is applied to extract ridge surfaces from intensity volumes and reconstruct surfaces from point sets with unoriented normals.

## 1. INTRODUCTION

Extremal surfaces (ES) are a type of implicit surfaces defined at the extremity of a scalar field restricted to a direction field [6]. Specific examples of ES include ridge surfaces of intensity volumes and point set surfaces, which are of great interest in practical applications. Unlike iso-surfaces, which are closed and orientable implicit surfaces, ES may contain open boundaries and can be un-orientable, which makes ES challenging to compute. In this paper we develop a sound approach for computing discrete approximations of ES. Our grid-based algorithm is simple to implement, applicable to both uniform and adaptive grids, and preserves key structural properties (e.g., manifoldness) of ES.

### 1.1 Definition

Let $s : \mathbb{R}^d \to \mathbb{R}$ and $n : \mathbb{R}^d \to \mathbb{RP}^{d-1}$ (the real projective space consisting of all lines passing through the origin in $\mathbb{R}^d$) be respectively a scalar function and an *unoriented* vector function. Both functions are defined over some domain $D \in \mathbb{R}^d$. ES are then defined as [6]:

$$S = \{x \,|x \in \text{arglocalmin}_{y \in l(x,n(x))} s(y)\} \tag{1}$$


*SPM* '10 Haifa, Israel


where $l(x, n(x))$ denotes the line through $x$ with direction $n(x)$. Intuitively, $S$ consists of all points $x$ at which the scalar function is minimum *along* the direction at $x$. A 2D example is shown in Figure 1 (d), where $s$ is plotted in gray scale (larger values are lighter), $n$ as short lines, and $S$ as the red curve. By choosing appropriate $s$ and $n$, different surface definitions can be formulated as ES:

**Height ridges:** The classical "height ridges" in a scalar function $L$ [4] are the loci where $L(x)$ is maximum along the direction where the second derivative is negative and having greatest magnitude. Such ridges can be defined by Equation 1 with the scalar field $s(x) = -L(x)$ and the vector field $n(x)$ as the eigenvector of the Hessian matrix of $L$ at $x$ with the smallest negative eigenvalue.

**Surface ridges:** Ridge lines on surfaces in 3D can also be formulated as extremal surfaces in 1D (that is, extremal curves). For example, the definitions in [12] and [8] can be expressed by Equation 1 using the sign-inverted maximum curvature (or apparent maximum curvature) as the scalar function and the corresponding curvature direction as the vector function.

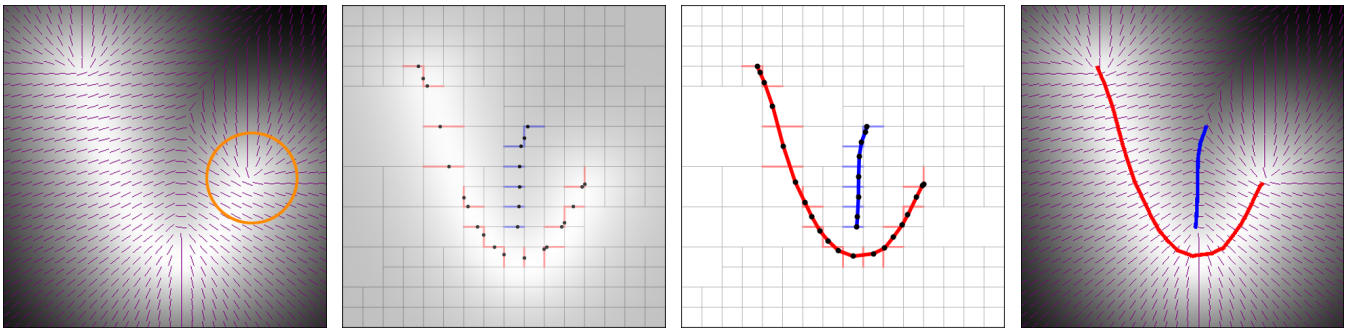
**Point set surfaces:** Amenta and Kil pointed out that the Moving Least Squares (MLS) surface proposed by Levin [10] can be re-formulated as an ES where the scalar function $s(x)$ is extended to a two-parameter function $e(x, v)$ that also depends on a direction $v$ [1], and also proposed several variants of these functions [2].

### 1.2 The structure of ES

One of the key properties that make ES appealing is its continuous structure. We start by considering the *critical surface* (CS) made up of all critical points of $s$ (where the derivative is zero) restricted to the line $l(x, n(x))$. Note that these critical points can be local minima (in which case they are on the ES), local maxima, or inflection points of $s$ along that line. As a result, the ES is a subset of the CS.

It was observed [6, 1] that the CS is *generally a manifold* as long as $s$ is smooth and $n$ is continuous. Here, continuity of an unoriented function $n$ at a point $x$ means that $n(x)$ is well-defined and that the variation in the directions represented by $n$ becomes arbitrarily small given a sufficiently small neighborhood around $x$. To understand the manifold property, take a small neighborhood of $x$ where $n$ is continuous and form a continuous *oriented* vector function $\vec{n}$. Let $\nabla s(x)$ be the gradient vector of $s$. Then the CS is the zero-set of the following scalar function:

$$g(x) = \vec{n}(x) \cdot \nabla s(x) \tag{2}$$



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

**Figure 1: The flow of the algorithm in 2D. From left to right: Given a scalar (gray level in (a)), an unoriented vector function (short lines in (a)), and a spatial grid (the quadtree in (b)), the algorithm first identifies grid edges intersected by *all* critical curves (blue and red edges in (b)), a subset of which are intersected by the extremal curves (red edges). Next, the algorithm constructs a polyline that crosses those grid edges (c). (d) shows the final curves in the original vector field.**

Note that $g(x)$ would be undefined wherever $n$ is discontinuous, hence the CS would exhibit an open boundary there.

Furthermore, assuming $s$ is a smooth function with continuous second derivatives, the ES is bounded on the CS by inflection points of $s$ restricted to lines $l(x, n(x))$. These inflection points generally form a manifold at a lower dimension than ES (e.g., curves in 3D), as they are the intersection of the CS and the zero-set surface of the second derivative of $s$ restricted to $n$.

To summarize, the ES is a $(d-1)$-manifold with open boundaries given a smooth scalar function $s$ in $\mathbb{R}^d$. The boundaries lie either at the discontinuity of $n$, or at the inflection points of $s$ restricted to lines $l(x, n(x))$.

## 1.3 Previous construction methods

To compute ES as explicit surface meshes, most existing works (such as [6, 5, 13, 9, 15]) assume that a global, continuous orientation exists for the input vector function $n$, in which case the ES can be extracted as a subset of the iso-surface of $g(x)$ (Equation 2) using techniques like Marching Cubes [11]. However, unorientable vectors functions are quite common in practice, such as in the scenario of height ridges [14]. A 2D example is shown in Figure 1 (a), where $n$ is not orientable along the highlighted loop (the vectors twist for a total of 180°).

Methods that can handle unoriented vector functions are scarce and are mostly proposed for 2D domains [12, 3, 8], with the exception of a recent work by Schultz *et al.* [14]. Given a spatial grid over the domain, these methods identify grid edges that are intersected by the ES, and connect the intersection points to form curves (in a 2D domain) or surfaces (in a 3D domain).

There are a number of limitations of these algorithms that we aim to address in this work. First, as we shall detail in Section 2.1, the typical criterium used for identifying grid edges intersected by the ES in these methods is highly sensitive to the sampling resolution of the grid. Next, connecting intersection points in 3D to form polygonal pieces is a non-trivial task, especially near the boundary of the ES. The method in [14] involves ad-hoc polygonization rules which are complex and restricted to a uniform cubic grid. More importantly, these methods do not guarantee to preserve the structural properties of the ES (e.g., those stated in Section 1.2). Their results may contain open boundaries or non-manifold features due to insufficient grid resolution, rather than due to the presence of singular features in the input scalar and vector functions.

## 1.4 Our contribution

We present a novel, grid-based algorithm for extracting a discrete extremal surface in 3D. The method handles unoriented vector functions, and makes several improvements over previous works such as [14]:

1. We adopt a robust criteria for identifying grid edges crossed by the ES.
2. We propose a simple polygonization routine that is generally applicable to any grid types.
3. We show that our algorithm preserves the structural property of the ES in a grid-independent manner. In particular, non-manifold features of the resulting polygonal surface appear only in the vicinity of the singular features of the input scalar and vector functions.

# 2. THE ALGORITHM

The input to our algorithm is a pair of scalar and unoriented vector functions, as well as a spatial grid over the domain. Note that the choice of the functions depends on the specific input data and the type of surface to be extracted. Likewise, the choice of grid structure also depends on the desired application. We discuss specific choices in Section 3; the algorithm presented here is applicable to *any* set of choices.

Our algorithm to compute a polygonal approximation of an ES proceeds in two steps, illustrated in 2D in Figure 1 for the input scalar and vector function in (a) and the quadtree grid in (b). First, we identify those grid edges that are crossed by the critical surface (CS). We call these edges *critical edges*, which are highlighted in red and blue in (b). The subset of critical edges intersected by the ES, called *extremal edges*, are the ones colored red. Next, we create a polygonal critical (extremal) surface that crosses the identified critical (extremal) edges (thick curves in (c)). We will first detail each step, then provide an analysis of the structure of the resulting surfaces.

## 2.1 Identifying grid edges

### 2.1.1 Identifying critical edges

As discussed in Section 1.2, critical points of $s$ restricted to $n$ are the loci where $g(x)$ (Equation 2) evaluates to zero, assuming the vector function $n$ can be continuously oriented. If the continuous-orientation assumption holds, a sufficient condition for a grid edge to be critical is that the value of $g(x)$ at the grid ends have different signs, meaning that $g(x)$ became zero somewhere along the line.

We make a key observation here that $n$ is, in general, orientable along any grid edge. To see this, we first note that a continuous $n$ along the edge is always orientable: starting with an arbitrary orientation of $n$ at one end of the edge, we can continuously assign an orientation to every other point by walking along the edge to the other end (a more general statement is proved in the Appendix). Next, we note that the discontinuity of $n$ tends to stay away from the grid edges in most applications (e.g., ridges and point set surfaces), where $n$ is defined as the principle eigenvectors of a tensor field. This is because such discontinuities generally form $d-2$-dimensional structures [16], which do not have stable intersections with 1-D edges. This leads to our test:

PROPOSITION 1. *An edge with end points $\{x_1, x_2\}$ is critical if $g(x_1)g(x_2) < 0$, where $g$ is defined in Equation 2 and $\vec{n}$ is an orientation of $n$ along the edge.*

We compare this test in Figure 2 with that used in previous methods [12, 3, 8, 14]. In these methods, a critical edge is identified as one with a negative dot product $\vec{v}(x_1) \cdot \vec{v}(x_2)$ where $\vec{v}(x) = g(x)\vec{n}(x)$. Intuitively, $\vec{v}(x)$ is the gradient of the scalar function $s(x)$ along the line $n(x)$, and this test checks if $s$ at the two ends of the edge fall off in opposite directions. Note that the value of $\vec{v}$ is independent of the choice of orientation $\vec{n}$, hence this test does not require orienting $n$ along the edge. The dot-product test performs well in the example of Figure 2 (a): $\vec{v}(x_1), \vec{v}(x_2)$ (pink arrows) point away from each other, indicating a critical point (green) where $g(x) = 0$. However, it fails in the example of (b), where $n$ twists significantly along the edge: $\vec{v}(x_1), \vec{v}(x_2)$ still point away from each other, but there is no critical point. In contrast, our test requires the knowledge of the orientation of $n$ along the edge (in order to compute $g$), therefore is able to give accurate prediction. In these two examples, $g(x_1)g(x_2)$ is negative in (a) while positive in (b).

In our implementation, we discretely transport the orientation from one end of the edge to the other over intervals where the vectors $n$ at the two ends of the interval are sufficiently similar. For efficiency, the intervals are found by a top-down binary search. Starting with the entire edge, we divide an interval $\{x_1, x_2\}$ into two halves if the acute angle between $n(x_1), n(x_2)$ is larger than some user-chosen threshold $\epsilon$. The search is guaranteed to stop if $n(x)$ is continuous on the edge. Note that this discrete implementation yields a correctly oriented pair $\vec{n}(x_1), \vec{n}(x_2)$ if $n(x)$ twists no more than $(180^\circ - \epsilon)$ within each interval.

### 2.1.2 Identifying extremal edges

Extremal edges are identified as those critical edges whose critical points are local minimum of $s$ restricted to $n$. Following the strategy in previous works [12, 3, 8, 14], we compute the location of critical point by linear interpolation along the critical edge $\{x_1, x_2\}$ using the magnitude of $g(x_1), g(x_2)$. A critical point $x$ is considered a local minimum if the second derivative of $s$ along $l(x, n(x))$ is positive. Figure 1 (b) shows an example of extremal edges (red) and non-extremal critical edges (blue) identified using our method, along with the locations of critical points.

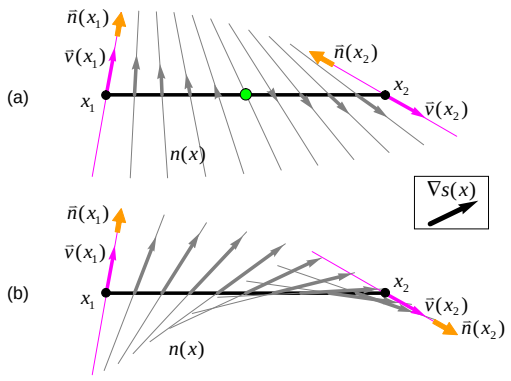



**Figure 2: Where the dot-product test fails: Both (a) and (b) have the same $\vec{v}$ at their end points, whose dot product is negative. Yet only (a) contains a critical point of $s(x)$ (with a constant gradient shown on the right) restricted to $n(x)$ (the gray lines).**

## 2.2 Polygonization

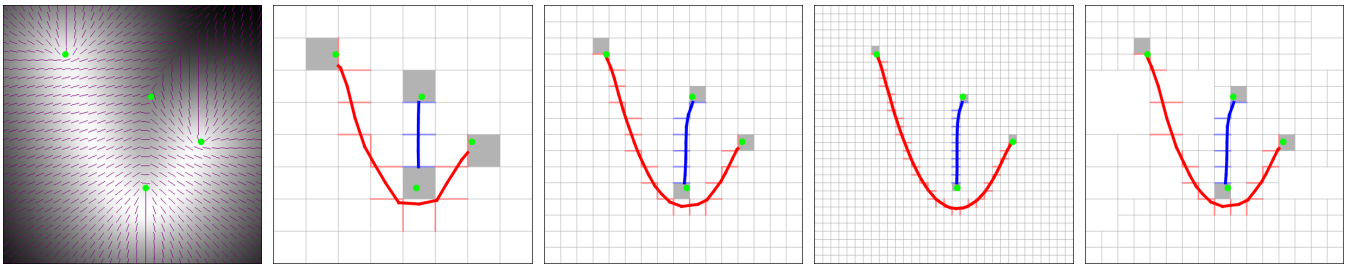
After identifying the extremal edges, we next generate a polygonal surface crossing the extremal edges while approximating the locations of the critical points along those edges. Note that we can build the non-extremal part of the CS in a similar manner by considering the non-extremal critical edges instead.

Our polygonization method is simple to implement and applicable to non-uniform grids in adaptive sampling (e.g., octrees). This is achieved by adapting an iso-surfacing algorithm, Dual Contouring (DC) [7]. DC places vertices *within* grid cells that exhibit a sign change, and constructs polygons that *cross* the grid edges with a sign change. To extract extremal surfaces, we create a vertex within any grid cell that contains an extremal edge, and create a polygon for each extremal edge by connecting the vertices within the cells sharing that edge. To determine the location of the vertex within a grid cell, we take the centroid of the critical points computed previously on all of the cell's extremal edges, then project this point onto the ES using the projection strategy in [1]. A 2D example of the polygonized ES is shown in Figure 1 (c). Note that the algorithm naturally handles boundaries and the adaptive grid structure.

## 2.3 Analysis

As shown in Section 1.2, an ES is a subset of a manifold that is only bounded by singular features of the input functions, such as discontinuities of $n$ or inflection points of $s$ restricted to $n$. As our main result, we will show that the polygonal ES generated by our algorithm possesses a similar structure that is governed by the topology of the input functions rather than the structure of the grid. Let the valence of $e$ be the number of polygons incident to $e$:

PROPOSITION 2. *An edge $e$ on the polygonal ES has an odd valence only if the dual grid face of $e$ contains either a discontinuity of $n$ or an inflection point of $s$ restricted to $n$.*



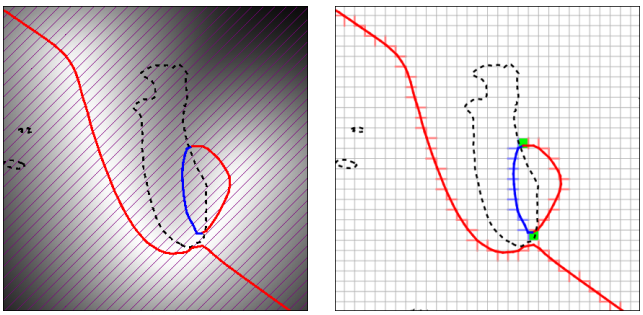
(a) (b) (c) (d) (e)

**Figure 3: (a) Input scalar and vector functions, where green dots highlight the discontinuities in the vector function. (b-d) Critical curves (with extremal curves colored red) computed on various grid types and resolutions, where shaded squares are the dual faces to odd-valence vertices on the critical curves.**

where the *dual grid face* of a polygonal ES edge $e = \{v_1, v_2\}$ is the grid face shared by the two grid cells containing the vertices $v_1$ and $v_2$ respectively. Note that the grid face dual to the odd-valence edge may contain multiple discontinuities of $n$ or inflection points of $s$. The proof is provided in the Appendix.

Intuitively, an odd-valence edge represents a non-manifold surface feature, since the local surface there cannot be topologically decomposed into manifold pieces. The above property ensures that each non-manifold feature on our polygonal surfaces lies in the vicinity of some singular features of the input functions. In this sense, our discrete approximation of ES is always *as manifold as* the actual ES.

To illustrate this property, consider again the 2D example in Figure 1 (a). Note that the vector function has a number of discontinuities, highlighted as green dots in Figure 3 (a). Figures 3 (b-d) show the polygonal extremal curves (red) and the non-extremal part of the critical curves (blue) computed on various grid types. In these pictures, the dual faces of odd-valence vertices on the computed curves are highlighted. Note that each dual face, regardless of the grid type and resolution, contains a discontinuity of the vector function in (a). Figure 4 (a) shows another example where the vector function in Figure 3 (a) is replaced with a constant field without discontinuity. Note that the extremal curve in the middle meets the non-extremal critical curve at two valence-one vertices, whose dual faces (colored green in (b)) intersect with the zero-set curve (dotted) of the second derivative of $s$ restricted to $n$.



(a) (b)

**Figure 4: Critical curves (with extremal curves colored red) computed on a constant vector field. The dotted curve is the zero-set of the second derivative of $s$ restricted to $n$, and the green grid faces are dual to the odd-valence vertices on the extremal curve.**

## 3. RESULTS AND APPLICATIONS

Here we present some results of our algorithm in 3D. We consider two specific kinds of extremal surfaces: ridges from intensity volumes and point set surfaces. In each scenario, we will discuss our choice of scalar and vector functions, as well as the type of spatial grid (e.g., uniform or adaptive) on which the algorithm is applied. In all examples, the threshold used for orienting vectors along a grid edge, which is the only parameter in our algorithm, is set to be $\epsilon = 10°$.

**Ridge surfaces:** We consider a variation of the classical height ridges of intensity images [4]. In the height ridge definition, the direction along which the height maxima is sought is the one that maximizes second derivative magnitude, which can be unstable for near-linear intensity distributions (e.g., distance functions). Instead, we use the direction where the intensity varies most, which relies only on first order derivatives. Specifically, given a discrete image $\mathbf{I}$ consisting of pixels $V$, we compute a discrete *structure tensor* $\mathbf{T}$ at a pixel $p$ as

$$\mathbf{T}(p) = \sum_{q \in V} \theta(\|p - q\|) \langle \nabla \mathbf{I}(q), \nabla \mathbf{I}(q)^T \rangle$$

where $\theta(r) = e^{-r^2/h^2}$ is a Gaussian kernel, $\nabla$ is the gradient operator, and $\langle , \rangle$ is the outer product operator. The eigenvector of $\mathbf{T}(q)$ with the largest eigenvalue represents the direction with greatest intensity variation around $q$. Letting $T$ be the bi-cubic interpolation of $\mathbf{T}$, we set $n(x)$ at an arbitrary space location to be the dominant eigenvector of $T(x)$. Following the height ridge definition, we set $s(x) = -L(x)$ where $L$ is the bi-cubic interpolation of a smoothed image $\mathbf{L}$ whose value at a pixel $p$ is

$$\mathbf{L}(p) = \sum_{q \in V} \theta(\|p - q\|) \mathbf{I}(q)$$

Figure 5 shows two examples of ridge extraction of 3D density volumes, where the algorithm ran on a uniform grid at the same dimension of the input. These data (show on the left) are volumetric reconstructions from 2D micrographs obtained by cryo-electron microscopy (cryo-EM), which capture the electron density around the atoms that make up proteins (shown on the right). Note that our algorithm is able to extract a contiguous ES (shown in the middle) despite its highly non-manifold structure. Also note that the ridge surfaces closely resemble the protein shape, making them a potential starting point for automatic recognition of protein structures from the input volumes. In this example, we color the surfaces by the ratio $\lambda_1/(\lambda_1 + \lambda_2)$ (higher

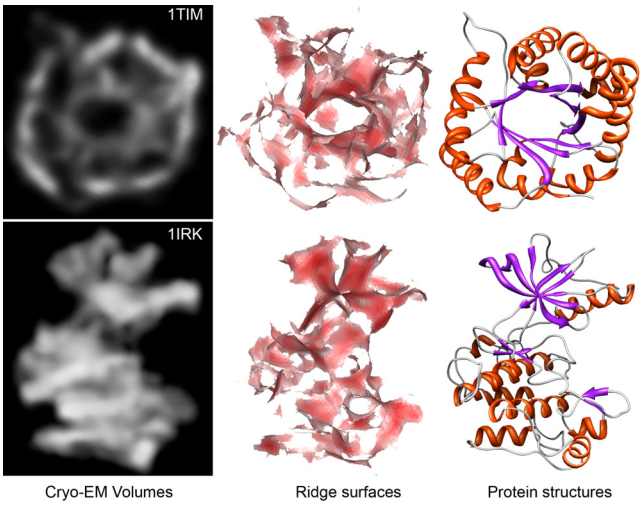



**Figure 5: Ridge surfaces of cryo-EM density volumes of proteins.**

ratios are redder), where $\lambda_1, \lambda_2$ are the largest and second largest eigenvalues of the structure tensor $T(x)$. A higher ratio indicates a more plate-like density distribution.

**Point set surfaces** We consider surface reconstruction from a point cloud associated with unoriented normals. As suggested by Amenta and Kil [1], we construct $n$ by blending the point normals. Given point and normal pairs $\{p_i, n_i\}$, they formulate $n(x)$ at any space location $x$ as the eigenvector of the following tensor with the greatest eigenvalue

$$G(x) = \sum_{p_i} \theta(\|x - p_i\|)\langle n_i, n_i^T \rangle$$

To construct $s$, we consider a recent formulation by Sussmuth et. al. [15] as the sum of (sign-inverted) Gaussian weights:

$$s(x) = -\sum_{p_i} \theta(\|x - p_i\|)$$

This function reaches local minima around the data points and increases smoothly away from the data (which avoids the narrow-domain problem of the weighted least-square formulation in [1]).

For point sets, we ran our algorithm on an octree grid to achieve adapted sampling. Starting from the bounding box of the data as the root cell, we recursively subdivide a cell if it contains more than one data point, if any two vectors $n(x)$ at the cell corners differ by more than $\epsilon$ (the same threshold as we used for orienting vectors along a grid edge), or if the second derivative of $s$ along the lines $l(x, n(x))$ at the cell corners exhibit different signs. The first criteria ensures the grid has sufficient resolution around the data points, while the next two criteria are used to detect possible variation of the ES within the cell. The subdivision terminates at a user-specified maximum depth (10 in our examples).

We show some examples of point set surfaces in Figure 6. Observe that the octree structure captures well the highly curved surface features (e.g., the tip of the bunny ear) using cells at fine resolutions, and the algorithm results in continuous and smooth surfaces on this adaptive grid. Also note that the algorithms can create unorientable surfaces (e.g., the Möbius strip), as it does not require orienting the input vectors $n(x)$. We further demonstrate the stability of our algorithm on input functions with fluctuations in Figure 7, where the input points were synthetically perturbed. Note that the algorithm results in continuous surfaces even though the underlying ES is highly undulating. The jagged boundary in these examples is caused by our implementation restricting the surface computation to the vicinity of the data points, as the ES defined above would otherwise extend to infinity. A better choice of functions resulting in ES within a domain bounded by the data points is an interesting question to pursue in the future.

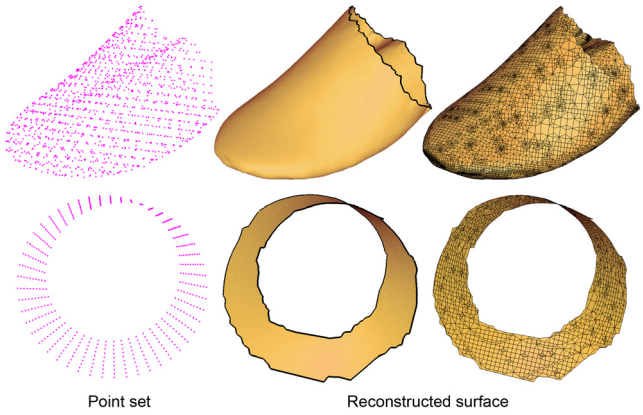



**Figure 6: Point set surfaces computed on adaptive octrees for the ear of the Stanford Bunny (top) and a Möbius strip.**

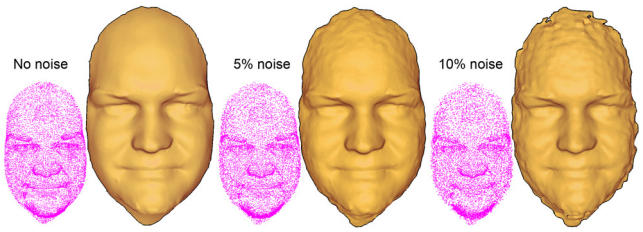



**Figure 7: Reconstruction from points with noise.**

**Performance** The most time consuming part of the algorithm is evaluating the input functions $s, n$, which is particularly frequent during grid subdivision (in the point set surface application) and when orienting vectors along the grid edges for identifying extremal edges. On the larger tests (e.g., Figure 6 (a)), grid subdivision and identifying extremal edges took around 20 minutes, while the polygonization step took no more than 2 seconds.

## 4. CONCLUSION

In this paper, we present a simple, grid-based algorithm for extracting extremal surfaces that does not require orienting the input vectors. A key property of our algorithm is that it preserves the manifold structure of the extremal surface in a grid-independent manner. The algorithm can be applied to reconstruct various types of extremal surfaces, such as ridge surfaces and point set surfaces. In the future, we would like to explore heuristics for generating adaptive spatial grids that can accurately capture the topology and geometry of the ES, in addition to its manifold structure.

Also, we would like to investigate better scalar and vector functions to define extremal surfaces in specific applications.

**Acknowledgement** This work is supported in part by NSF grants IIS-0705538, IIS-0846072, CCF-0702662 and DBI-0743691. We thank Matt Baker and Stanford 3D Scanning Repository for providing the density and point set data.

## 5. REFERENCES

[1] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[2] N. Amenta and Y. J. Kil. The domain of a point set surfaces. In *Eurographics Symposium on Point-based Graphics*, pages 139–147, 2004.

[3] F. Cazals and M. Pouget. Topology driven algorithms for ridge extraction on meshes. Research Report RR-5526, INRIA, 2005.

[4] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *J. Math. Imaging and Vision*, 4:353–373, 1994.

[5] J. D. Furst and S. M. Pizer. Marching ridges. In *Signal and Image Processing (SIP 2001)*, pages 22–26, 2001.

[6] G. Guy and G. Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(11):1265–1277, 1997.

[7] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM Press.

[8] T. Judd, F. Durand, and E. Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[9] G. Kindlmann, X. Tricoche, and C.-F. Westin. Delineating white matter structure in diffusion tensor MRI with anisotropy creases. *Medical Image Analysis*, 11(5):492–502, October 2007.

[10] D. Levin. Mesh-independent surface interpolation. In G. Brunnett, B. Hamann, K. Mueller, and L. Linsen, editors, *Geometric Modeling for Scientific Visualization*. Springer-Verlag, 2003.

[11] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

[12] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 609–612, New York, NY, USA, 2004. ACM.

[13] F. Sadlo and R. Peikert. Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1456–1463, 2007.

[14] T. Schultz, H. Theisel, and H.-P. Seidel. Crease surfaces: From theory to extraction and application to diffusion tensor mri. *IEEE Transactions on Visualization and Computer Graphics*, 16:109–119, 2010.

[15] J. Sussmuth and G. Greiner. Ridge based curve and surface reconstruction. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 243–251, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[16] X. Zheng and A. Pang. Topological lines in 3D tensor fields. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 313–320, Washington, DC, USA, 2004. IEEE Computer Society.

## APPENDIX

## A. PROOF OF PROPOSITION 2

Before we present the proof, we first make the following observation about the orientability of a continuous, unoriented vector function:

LEMMA 1. *Let $D$ be a $k$-manifold in $\mathbb{R}^d$ where $k \leq d$, and $n : \mathbb{R}^d \rightarrow \mathbb{RP}^{d-1}$ be an unoriented vector function continuously defined over $D$. If $D$ is homeomorphic to a $k$-disk, than $n$ is orientable.*

**Proof:** When $k = 1$ (or $D$ is a line segment), the argument obviously holds as we can orient $n$ along the line in a continuous manner from end to the other. We can do the same for $k > 1$, by traversing all points in $D$ in an order such that, at every point $x \in D$, an orientation $\vec{n}(x)$ can be picked that is continuous with those already picked in the neighborhood of $x$. Specifically, due to homeomorphism to a disk, $D$ can be parameterized by $k$ parameters $\{t_1, t_2, \ldots, t_k\}$ where $t_i \in [0, 1]$. We visit points in $D$ in increasing values of each parameter (e.g., from $\{0, 0, \ldots, 0\}$ to $\{0.5, 0, \ldots, 0\}$, then to $\{1, 0, \ldots, 0\}$, then to $\{0, 0.5, \ldots, 0\}$, then to $\{0.5, 0.5, \ldots, 0\}$, then to $\{1, 0.5, \ldots, 0\}$, etc.), each time assigning an orientation for $\vec{n}(x)$ that is continuous with all orientated vectors $\vec{n}(y)$ where $y$ is in an infinitesimal neighborhood of $x$ and has already been visited. Note that such assignment is always possible, as the set of all such $y$ forms a connected (open) set around $x$ due to the visiting order, and the angle formed by $\vec{n}(y)$ and any assignment of $\vec{n}(x)$ would approach 0 or $\pi$ *simultaneously* for all $y$ in this set. □

Now we present the proof of Proposition 2:

**Proof:** (Proposition 2): Due to our polygonization procedure, the valence of an edge $e$ on our polygonal ES is the same as the number of extremal edges contained by the dual face $f$ of $e$. Therefore the odd-valence of $e$ means that there are odd number of extremal edges on grid face $f$, which implies one (or both) of the following two cases:

**Case 1:** $f$ has both types (extremal or non-extremal ) of critical edges. If so, $f$ contains an inflection point of $s$ restricted to $n$. This is because we distinguish extremal and non-extremal edges by the sign of the second derivatives at the critical points. The presence of both types of edges implies that $f$ intersects with the zero set of the second derivatives.

**Case 2:** $f$ has an odd number of critical edges. If so, $f$ contains a discontinuity of $n$. Otherwise, assuming $n$ is continuous along $f$, than $n$ is orientable along the border of $f$ by Lemma 1 (assuming $f$ has a disk-like topology, which is the case for most spatial partitioning schemes). Adopting this orientation, the function $g(x)$ (Equation 2) is a continuous scalar function over the border of $f$, and hence changes signs for an even number of times. As a result, the number of critical edges satisfying our criterium in Proposition 1 is even along the border of $f$, contradicting our assumption.□