Title
A Selective Refinement Approach for Computing the Distance Function of Curves

Authors
Laney, Daniel E.
Duchaineau, Mark A.
Max, Nelson

Publication Date
2001

Peer reviewed

A Selective Refinement Approach for Computing the Distance Functions of Curves

Daniel E. Laney[1], Mark A. Duchaineau[2], Nelson L. Max[1,2]

[1] Department of Applied Science, University of California at Davis
[2] Lawrence Livermore National Laboratory***

Abstract. We present an adaptive signed distance transform algorithm for curves in the plane. A hierarchy of bounding boxes is required for the input curves. We demonstrate the algorithm on the isocontours of a turbulence simulation. The algorithm provides guaranteed error bounds with a selective refinement approach. The domain over which the signed distance function is desired is adaptively triangulated and piecewise discontinuous linear approximations are constructed within each triangle. The resulting transform performs work only where requested and does not rely on a preset sampling rate or other constraints.

1 Introduction

The distance function of a shape encodes the minimum distance to the shape at every point in space. The encoded shape can be extracted by taking the isocontour of the distance function with an isovalue of zero. Distance functions have proven useful in modeling deformations of solid objects [11], representation of medical data [4], modeling swept surfaces and volumes [9], and increasing the performance of ray-casting via space leaping [12]. A closed shape allows a signed distance function to be defined that encodes inside/outside information and enables boolean operations and object morphing. Signed distance representations vary smoothly across shape boundaries, enabling accurate reconstruction of surface properties from the distance function. As an implicit representation, distance functions have the potential to simplify the storage and visualization of time varying surfaces with changing topology because they do not explicitly store the topology.

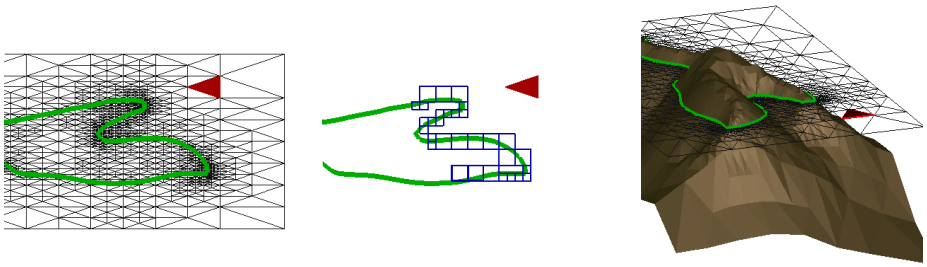
Distance functions are usually represented as sampled scalar fields with values between samples generated by interpolation. A propagation algorithm such as that of Breen [1] is the most commonly used method for obtaining an approximate distance function of a shape. These algorithms initialize a sampled scalar field with closest point information near the boundary of a shape and propagate this information throughout the volume. For large problems it would be advantageous to limit the amount of computation spent on areas with less detail. The adaptive sampling technique of Frisken [3] provides many of the benefits of distance representations without the excessive memory requirements. However, adaptive sampling implies that a distance function exists in closed form, or an approximate distance function exists at a high resolution. Our

*** {laney1, duchaineau1, max2}@llnl.gov

contribution is to provide adaptivity in the distance function itself through a distance transform based on selective refinement.

In this paper we present an adaptive signed distance transform for curves in two dimensions. We demonstrate the algorithm on isocontours of a turbulence simulation [7]. We are investigating distance functions in the plane as a precursor to a full three dimensional method. In two dimensions, the error analysis is simplified, and the behavior of the algorithm and data structures can be clearly visualized. Our algorithm utilizes a selective refinement approach which concentrates computation where a more accurate approximation is required.

The algorithm operates on an adaptive distance approximation represented by a triangulation of the desired 2D domain. Figure 1a shows the adaptive triangulation of the distance function for a small piece of isocontour from the test data. Figure 1b shows the bounding boxes which contain curve dependency information for the highlighted distance triangle. The dependency information reduces the work required to compute distance approximations within each triangle. We require the input curve to provide a bounding box hierarchy so that distance triangles can track potential contributing points efficiently. Figure 1c shows a continuous reconstruction of the distance approximation which satisfies the error bounds of every triangle.



a **b** **c**

Fig. 1. A node in the distance hierarchy (a) contains a list of nodes in the curve hierarchy (b)

The adaptive distance approximation has the following properties:

1. **Triangle bintree representation:** The adaptive distance is defined over a triangle bintree [2, 6]. The refinement rules for triangle bintrees insure that all adjacent triangles meet at a common vertex or a common complete edge. This property allows the construction of continuous functions on the triangle mesh.
2. **Per-triangle linear approximations with guaranteed error bounds:** Selective refinement operations must be as local as possible. Our approach uses per-triangle linear approximations which may be discontinuous with respect to the linear approximations in neighboring triangles. Each triangle also maintains an error bound for the linear approximation which is guaranteed to be correct with respect to the

exact distance function. The per-triangle approach requires an extra step to construct a continuous distance approximation, but this is always possible.

3. **Per-triangle tracking of dependencies on the input curve:** In general the distance function within a triangle depends on a subset of the input curve. Each triangle tracks the subset of the curve that contains every point which potentially contributes to the distance function. Linear approximations are computed with respect to the per-triangle dependency information. The selective refinement approach is based on the observation that the curve dependencies are typically reduced when a triangle is refined.
4. **Easy reconstruction of a continuous per-vertex distance approximation:** The per-triangle error bounds are guaranteed to overlap at the vertices and edges. A simple algorithm is presented which produces distance values at the vertices of the triangle bintree that satisfy the error bounds of the per-triangle approximations. The triangle bintree automatically insures that no cracks are present in the continuous distance approximation.

The remainder of the paper first presents related work in distance transforms. We then describe the curves on which the algorithm was tested, including our method of obtaining a bounding box hierarchy. Finally, we detail the distance transform algorithm itself and characterize the behavior of the algorithm on the test data.

2 Related Work

The closest point propagation technique of Breen [1] is based on the fast marching method of Sethian [10]. The main advantages of propagation algorithms are speed and simplicity which offsets the fact that they are usually applied to sampled volumes for which only a subset of the samples will be used. An exception to this is object morphing which requires operations on the entire volume. One disadvantage is that propagation methods rely on the user to set the appropriate sampling rate for a given object. In addition, the distance computation is sometimes sampled on a much finer lattice and then sub-sampled to the desired resolution of the distance approximation.

Perhaps closer in spirit to the approach of this paper is the surface reconstruction algorithm of Hoppe [5]. They define a signed distance function for scattered point data and reconstruct a plausible surface by taking the zero isosurface of the distance function. At the heart of the algorithm is a distance transform which relies on local best fit planes computed for each surface sample and its neighborhood. The reconstructed surface is extracted as the zero set of the approximate distance function. The approach in the present work requires normal vector information which is not usually present in data from range scanners and was not required by the method of [5].

3 The Curve Hierarchy

For this paper we have chosen to test the algorithm on isocontours of a regularly sampled scalar field [7]. In this section we describe the the curve hierarchy required as input

to the distance transform. We begin by stating the properties of the input curve hierarchy for the general case, then describe in detail the isocontour hierarchy used in this paper.

3.1 Curve Hierarchy Requirements

Let C denote the set of points on the input curve(s) and D denote the domain in $\Re^2$ for which an approximate distance function is desired. The input curve must have the following properties:

1. **Continuity:** The curve(s) must be C^0 continuous everywhere and C^1 continuous everywhere except at a finite number of points.
2. **Partitions the domain:** The curve(s) must partition the domain D into an inside and outside labeled by negative and positive distances. This also implies that the curve end points may only occur on or outside the boundary of D.
3. **Bounding Boxes:** The curve must allow a hierarchy of bounding boxes to be specified. In the remainder of the paper we will index these bounding boxes by α. Each node in the hierarchy must define a bounding box B_α in $\Re^2$ which contains a subset of the curve.
4. **Normal Wedges:** Each curve hierarchy node α must define a bound on the directions of all unit normals of the curve contained in B_α. The bound is represented by a normal wedge $(\boldsymbol{n}_\alpha, \psi_\alpha)$ with central unit normal $\boldsymbol{n}_\alpha$ and opening half angle $0 \leq \psi_\alpha \leq \pi$. A unit vector $\boldsymbol{v}$ is contained in a normal wedge for node α if the following condition is met: $\boldsymbol{v} \cdot \boldsymbol{n}_\alpha \geq \cos(\psi_\alpha)$.

Any number of data structures and curve definitions could be used to construct a curve hierarchy. We have chosen a hierarchy based on scalar field data generated by scientific simulation. The next subsection describes this isocontour hierarchy.

3.2 Constructing An Isocontour Hierarchy

Isocontours of a sampled scalar field with bilinear interpolation satisfy the first two requirements listed above as long as the domain D is the same as the scalar field domain or a subset of it. Two strategies for satisfying items 3 and 4 are possible: (1) Extract the isocontour and build a bounding box hierarchy from the geometry, or (2) Construct a spatial hierarchy on the scalar field and use it to generate bounding box hierarchies for all isocontours. The first strategy will result in the tightest bounding boxes because the geometry is exactly specified. We adopt the second strategy because spatial data structures such as min-max octrees are already commonly used to accelerate isocontouring operations and adding normal wedge information to them is straightforward.

We compute a min-max quadtree on the scalar field and compute bounds on the directions of the isocontour normals using the scalar field gradient. Figure 2 shows an example scalar field and three levels of the quadtree hierarchy. The normal wedges are defined by the scalar field gradients and are drawn inside each node with the opening angles denoted by dotted arcs.

The scalar field hierarchy has loose bounds on the normal direction because the direction bounds are valid for all isocontours within a particular node. This forces the

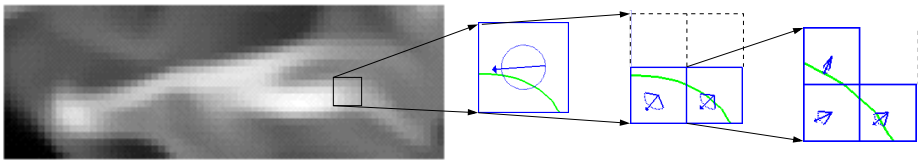


Fig. 2. (Left) A sampled scalar field rendered with one pixel/sample. A single quadtree node is highlighted. (Right) Three levels of the isocontour bounding box hierarchy.

distance transform algorithm to subdivide more often in areas where the original scalar field gradients are widely varying. This hierarchy is similar to the data structure used in [8] in that it involves minimal preprocessing and grants access to all of the isocontours in the scalar field.

4 The Distance Hierarchy

We begin this section by formally defining the signed distance function of a curve or set of curves C as follows:

$$\mathrm{d}(\boldsymbol{x}) := \mathrm{sign}(\boldsymbol{x}) * \min_{\forall \boldsymbol{y} \in C}(\| \boldsymbol{x} - \boldsymbol{y} \|) \tag{1}$$

where C is the set of points on the input curve(s), $\boldsymbol{x}$ and $\boldsymbol{y}$ are points in $\Re^2$, $\boldsymbol{y} \in C$, and $\mathrm{sign}(\boldsymbol{x})$ returns negative if $\boldsymbol{x}$ inside the curve, and positive if outside.

We will use T_k to refer to the triangle associated with a distance node k. All other quantities associated with a node shall be denoted by subscripting them with k.

4.1 Triangle Bintrees

The triangle bintree begins with a right isosceles triangle. Each level of the triangle bintree is created by edge bisection operations. A triangle is split by inserting a vertex at the midpoint of the hypotenuse and connecting it to the opposite vertex of the triangle. This operation is then applied recursively to the children. A new vertex is created when a triangle is split. This creates a crack problem because the triangle sharing the edge may not be split. The crack problem is solved by forcing splits until there are no vertices on any edges. In this paper we begin with a rectangular domain and relax the constraint that each triangle must be right isosceles.

4.2 Linear Approximation

A distance node maintains a list of curve bounding boxes which we will refer to as its *active list*. The bounding boxes in the active list contain all curve points which potentially contribute to the distance function in the node. In the remainder of the paper we will denote the active list of a distance node by A_k and define it as a list of indices of

curve nodes as follows: $A_k := \alpha_0, \alpha_1, \ldots, \alpha_N$. We will define the linear approximation for a node k as:

$$\tilde{d}_k(\boldsymbol{x}) = \boldsymbol{g_k} \cdot \boldsymbol{x} + c_k \tag{2}$$

where the gradient $\|\boldsymbol{g_k}\| = 1$. In addition, the error bound on the distance function within a distance node k must satisfy:

$$\| \max_{\forall \boldsymbol{x} \in T_k} (\mathrm{d}(\boldsymbol{x}) - \tilde{d}_k(\boldsymbol{x})) \| \leq \varepsilon_k \tag{3}$$

In general, we want $\varepsilon_k < \mathrm{uerror}(k)$, where $\mathrm{uerror}(k)$ is a user defined error which depends on k and may depend on other parameters as well. When the distance function is too complicated to be linearly approximated, we fall back on a constant approximation ($\boldsymbol{g}_k = 0$).

5 The Distance Transform

Given a distance node k, the transform algorithm recursively subdivides k until a user defined error criterion is met over the original triangle T_k. The subdivision may cause forced splittings of some triangles outside of T_k as mentioned in section 4.1. We assume that k contains an active list A_k of all curve nodes which may contribute to the distance function within T_k. The algorithm proceeds as follows:

1. **Establish Bounds:** For each curve node in A_k compute conservative lower and upper distance bounds. These bounds are computed with respect to the bounding regions B_α and the distance triangle T_k. These bounds are used to compute lower and upper bounds on the distance within T_k.
2. **Refine and Cull:** Place the curve nodes in A_k in a priority queue. Let the priority $p(\alpha) = 2r \sin(\psi_\alpha)$ be a bound on the error of a linear approximation of a node, where r the length of the node diagonal. This error bound stems from the fact that once a point on the curve is known the normal wedge constrains the possible directions of the curve. In priority order refine the nodes in A_k until $\mathrm{p}(\alpha) < \mathrm{uerror}(k) : \forall \alpha \in A_k$. For each child node added to A_k compute conservative bounds as in step 1 and update the overall bounds on T_k. As the overall bounds tighten, *Bound Cull* any curve node in A_k that is unable to contribute to the distance function within T_k. When a curve node is bound culled label all adjacent curve nodes as gap nodes.
3. **Linear Approximation:** Compute a linear approximation of the distance function over T_k based on the refined active list nodes A_k. A point on the curve is sampled and the opening half angles of the curve normal wedges are used to compute a linear approximation with an error bound. A test is then made on A_k to insure that the bound is correct.
4. **Constant Approximation:** If no linear approximation with guaranteed error was computed, or the error of the linear approximation violates the user supplied error criterion, then compute a constant approximation as follows: Use the conservative bounds on the distance over T_k to compute a c_k and ε_k for equations 2 and 3.
5. **Recurse or End**: If the resulting approximation error does not satisfy the user defined error criterion, split distance node k and copy the active list A_k to both children. Repeat with step 1 for each child of k.

5.1 Bound Culling And Constant Approximations

In this section we describe how conservative guaranteed error bounds are established between a distance triangle T_k and the curve nodes in its active list A_k. We begin by defining when a point on the curve contributes to the distance function:

Definition 1. *A point $\boldsymbol{y} \in C$ contributes to the distance function within the triangle T_k of a distance node k if there exists at least one point $\boldsymbol{x} \in T_k$ for which $\boldsymbol{y}$ is the point of closest approach of the curve C.*

Furthermore, we say a curve hierarchy node α contributes to the distance function within T_k if at least one point $\boldsymbol{y} \in C \cap B_\alpha$ contributes. We define unsigned conservative lower and upper bounds on the distance function with respect to a single curve hierarchy node as $lower_{k,\alpha} = \min(\| \boldsymbol{x} - \boldsymbol{y}) \|$ and $upper_{k,\alpha} = \max(\| \boldsymbol{x} - \boldsymbol{y}) \|$ for all $\boldsymbol{x} \in T_k$ and all $\boldsymbol{y} \in B_\alpha$. The lower and upper distance bounds are are conservative because they rely only on the bounding box B_α, and not on the actual curve contained within B_α. These bounds can be expensive to compute and become more expensive in three dimensions. We use bounding circles for the distance computations for simplicity.

The next step is to bound the distance function due to all curve nodes in the active list A_k. These bounds are given by $d_{lower} = \min(lower_{k,\alpha})$ and $d_{upper} = \min(upper_{k,\alpha})$ for each $\alpha \in A_k$. Thus, we have the following condition on the distance function within distance node k:

$$d_{lower} \leq \mathrm{d}(\boldsymbol{x}) \leq d_{upper} \; ; \; \forall \boldsymbol{x} \in T_k \tag{4}$$

A curve hierarchy node does not contribute to the distance function if the following condition holds: $lower_{k,\alpha} > d_{upper}$. Finally, the lower and upper bounds may be used to construct a piecewise discontinuous constant approximation as $\tilde{d}_k(\boldsymbol{x}) = \frac{1}{2}\mathrm{sign}(\boldsymbol{x})(d_{lower} + d_{upper})$ and $\varepsilon_k = |d_{upper} - \tilde{d}_k(\boldsymbol{x})|$.

5.2 Computing Linear Approximations

The processes of computing a linear approximation and obtaining a guaranteed error bound are decoupled from one another. This is due to the fact that curves with complex foldings, disconnected components, and gaps created during bound culling produce distance fields which are difficult to bound correctly. In this section we describe how to compute linear approximations and error bounds. In the next section we describe how to guarantee that a bound is correct.

Equation (2) implies that the zero set of a linear approximation is the equation of a line with normal $\boldsymbol{g}_k$. We set $\boldsymbol{g}_k$ to the normalized average of all central normals of the normal wedges in A_k and compute an opening half angle ψ_k with respect to the merged normal wedges in A_k. The constant c_k is obtained by sampling a point $\boldsymbol{p}$ on the curve contained in the active list A_k and solving for the zero set: $\boldsymbol{g}_k \cdot \boldsymbol{p} + c_k = 0$. The point $\boldsymbol{p}$ is computed by choosing an isocontour node in the active list and finding a contour intersection with one of its edges using linear interpolation. We bound the error of the approximation using the normal wedge information of the active list nodes. The extent r of A_k is computed as the maximum distance from $\boldsymbol{p}$ to the boundaries of the nodes in A_k. We compute the error bound as $\varepsilon_k = r \sin(\psi_k)$. In practice, this simple method

works quite well near the curve. It is dependent on the variation of the normal directions of the curve hierarchy and tends to do less well for distance triangles farther from the isocontour.

5.3 Guaranteed Error Bounds

We assume a linear approximation has been computed and an error bound ε_k must be produced. We desire a tighter error bound than that presented in equation (4) . We can achieve this by utilizing information about the shape of the curve contained in A_k. Figure 3 illustrates how an error bound ε_k constrains the location of the contributing curve points. A constant approximation has no estimate of the gradient and only constrains contributing curve points to the annulus centered at $\boldsymbol{x}$ denoted by the dashed circles. A linear approximation with a guaranteed error ε_k further restricts the possible locations of contributing points to the shaded area between the isocontours of the linear approximation at $\pm\varepsilon_k$.

Given a possible ε_k, the distribution of curve points in the active list must be analyzed to insure that at least one curve point falls in the shaded area of Fig. 3 for each point in T_k. Gaps or folds in the curve must be accounted for to insure that the bound on the distance is correct. The general procedure is as follows:

1. Compute a possible error bound ε_k.
2. Insure that the curve does not have folds or loops by requiring that the curve bend no more than 90 degrees from the estimated gradient $\boldsymbol{g}_k$ and is entirely front facing or entirely back facing with respect to $\boldsymbol{g}_k$.
3. Insure that no gaps exist in the curve by requiring that no curve nodes in the active list A_k labeled as gaps exist in the shaded area of figure 3.
4. If both 2 and 3 are satisfied, then the ε_k computed in item 1 is a guaranteed bound.

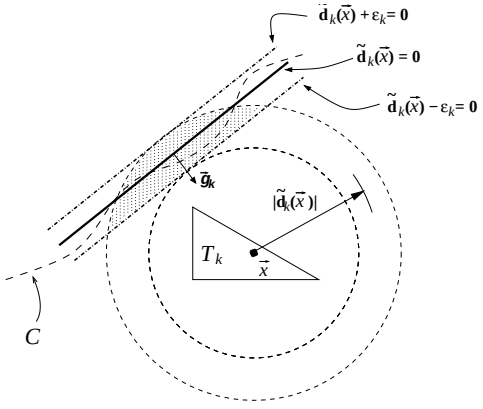


Fig. 3. Points of closest approach must fall inside the shaded area for a linear approximation with guaranteed error.

5.4 Producing a Continuous Approximation

The linear approximation within each triangle defines a range of distance values for each vertex of that triangle. The procedure for constructing a continuous distance function is the following. For each vertex the range of distance values is computed as the intersection of the distance ranges for each incident linear approximation. The midpoint of this range is chosen as the distance value of the vertex. This doubles the error bound on the approximation because the midpoint might occur at a lower or upper bound of one of the incident ranges. It is possible to use a more aggressive algorithm that uses the ranges of neighboring vertices to improve the accuracy of the continuous distance approximation.

6 Results

We tested our algorithm on scalar entropy data from a turbulent mixing simulation [7]. Figure 4 shows a 256×128 slice from one sub-domain of this computation with the isocontour at 50% of the maximum entropy. The error criterion we used in the examples was computed as follows: uerror$(k, \varepsilon_{min}, \lambda) := \max(\varepsilon_{min},\ \lambda d_{lower})$. where d_{lower} is computed for each distance node k as defined in section 5.1. This error criterion allows less accurate approximations farther away from the curve and clamps approximations near the curve to a user defined minimum. We found that on average each distance triangle tracked $15 - 17$ curve nodes in its active list. This value was not dependent on the values of λ and ε_{min}. Our experiments indicate that the execution time and number of triangles varies approximately linearly with $1/\varepsilon_{min}$. That is, reducing the error by a factor of two increases the number of triangles and execution by approximately the same factor.

Figure 5 shows an adaptive distance function and the corresponding triangulation of the isocontour in Fig. 4. The transform took 37 seconds to compute and required approximately 60 megabytes of memory. The final distance triangulation contains 70361 triangles. The implementation was not optimized for memory usage and uses pointer based hierarchies. In figure 6 the same distance approximation is shown.

7 Conclusion

We have implemented a fully adaptive distance transform algorithm that produces piecewise discontinuous linear approximations in a top down fashion. Although the implementation was not coded for speed, it is clear that a three dimensional version of the transform would require more efficient data structures. The algorithm is tuned for approximations near the input curve and tends to rely on constant approximations farther away. This could be improved by using a different approximation strategy for boxes farther from the curve.

The algorithm does not handle discontinuities in the gradient of the distance function in an efficient manner. Gradient discontinuities violate the observation that the curve dependencies are reduced when a distance triangle is refined. An extreme example of this is a triangle which contains the center of a circle. The distance function

within the triangle depends on the entire circle. At least one child triangle will contain the center regardless of the number of refinement operations.

8 Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. We wish to thank Valerio Pascucci for helpful discussions.

References

[1] David E. Breen, Sean Mauch, and Ross T. Whitaker. 3d scan conversion of csg models into distance volumes. In *Proc. 1998 Symposium on Volume Visualization*, pages 7–14, Oct 1998.

[2] Mark Duchaineau, Murray Wolinsky, David Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. Roaming terrain: real-time optimally adapting meshes. In *Proc. Visualization*, pages 81–88, 1997.

[3] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptive sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of SIGGRAPH 2000*, pages 249–254, July 2000.

[4] Sarah F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of the 1999 Conference on Volume Visualization*, pages 23–30, Oct 1998.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuelzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH 92*, pages 71–78, July 1992.

[6] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory A. Turner. Real-time, continuous level of detail rendering of height fields. In *Proc. SIGGRAPH*, pages 109–118, 1996.

[7] A. A. Mirin, R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White. Very high resolution simulation of compressible turbulence on the IBM-SP system. In ACM, editor, *Super Computing 1999, Oregon*, pages ??–?? ACM Press and IEEE Computer Society Press, 1999.

[8] V. Pascucci and C. L. Bajaj. time-critical isosurface refinement and smoothing. In *Proceedings of Volume Visualization and Graphics Symposium 2000*. IEEE and ACM/SIGGRAPH.

[9] William J. Schroeder, William E. Lorensen, and Steve Linthicum. Implicit modelling of swept surfaces and volumes. In *Proc. Visualization*, pages 40–45, 1994.

[10] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Science*, 93:1591–1595, 1996.

[11] Ross T. Whitaker and David E. Breen. Level-set models for the deformation of solid objects. In *Proc. of the 3rd International Workshop on Implicit Surfaces*, pages 19–35, June 1998.

[12] K. J. Zuiderveld, A. H. J. Konig, and M. A. Viergever. Acceleration of ray-casting using 3-d distance transforms. In *Proc. of Visualization in Biomedical Computing*, pages 324–335, 1992.

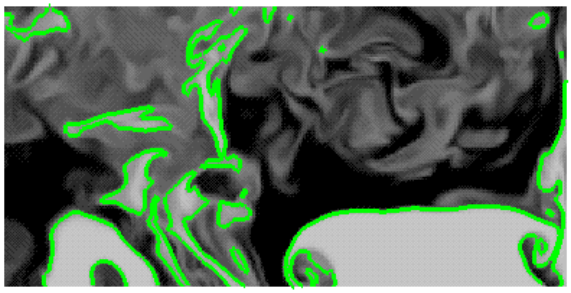


Fig. 4. A slice of the entropy field showing the isocontour used in the color figures.

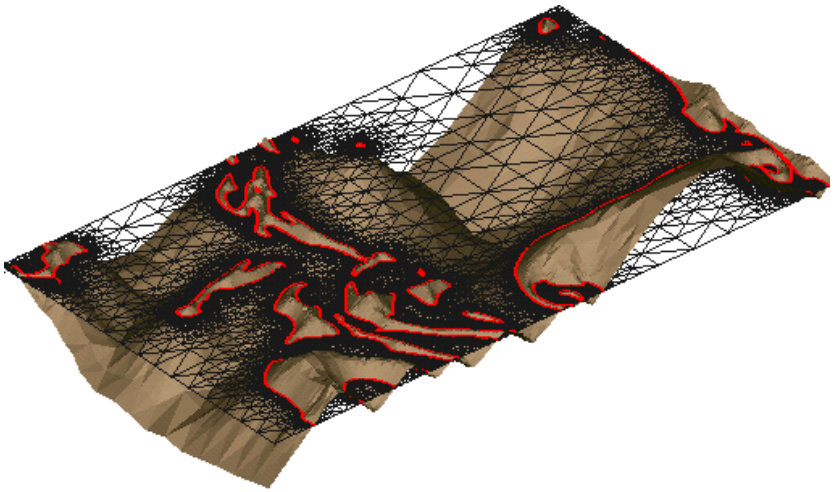


Fig. 5. An adaptive distance transform with $\lambda = 1$ and $\varepsilon_{min} = 0.5$. The adaptive distance triangulation is shown.

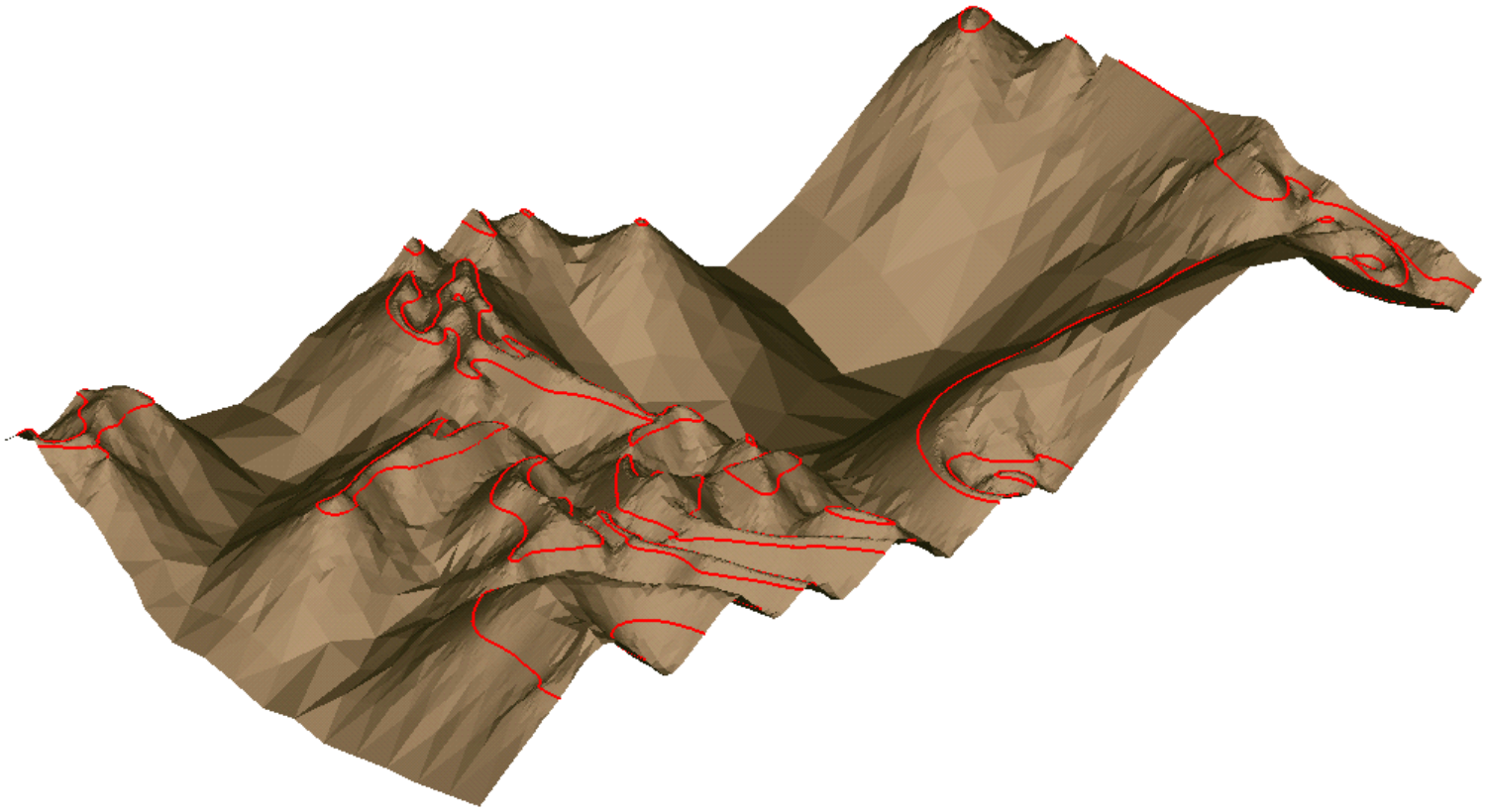


Fig. 6. An adaptive distance transform with $\lambda = 1$ and $\varepsilon_{min} = 0.5$.