Journal of Computational Physics **178,** 175–195 (2002)
doi:10.1006/jcph.2002.7028, available online at http://www.idealibrary.com on IDEAL®

Rapid and Accurate Computation of the Distance Function Using Grids

Yen-hsi Richard Tsai[1]

Department of Mathematics, University of California Los Angeles, Los Angeles, California 90095
E-mail: ytsai@math.ucla.edu

Received October 20, 2000; revised June 11, 2001

We present two fast and simple algorithms for approximating the distance function for given isolated points on uniform grids. The algorithms are then generalized to compute the distance to piecewise linear objects. By incorporating the geometry of Huygens' principle in the reverse order with the classical viscosity solution theory for the eikonal equation $|\nabla u| = 1$, the algorithms become almost purely algebraic and yield very accurate approximations. The generalized closest point formulation used in the second algorithm provides a framework for further extension to compute the distance accurately to smooth geometric objects on different grid geometries, without the construction of the Voronoi diagrams. This method provides a fast and simple translator of data commonly given in computational geometry to the volumetric representation used in level set methods. © 2002 Elsevier Science (USA)

1. INTRODUCTION

Given a compact set K in space, the distance function for K at a point x is defined to be the smallest Euclidean distance of x to the points in K. Fast and accurate computation of the distance function is an important step in many applications. For example, level set methods [14] rely heavily on it to represent an interface, and in particular to keep the level set function well behaved during time evolution. In the surface interpolation model of [28], a surface interpolating a given points set K is constructed by first computing the distance function to K. In the Island Dynamics model of [6] in materials science, one needs the distance function to help understand the aggregation of atoms into islands. Generally, applications in physics, chemistry, molecular biology, and even urban planning that use point pattern analysis require construction of the distance function or (at least) Voronoi diagrams [5, 12]. Furthermore, in applications such as ray tracing and surface rendering where one needs

[1] Research supported by ONR N00014-97-1-0027, DARPA/NSF VIP Grant NSF DMS 9615854 and ARO DAAG 55-98-1-0323.

information about the normals or other geometric quantities of the surfaces of the given objects, the distance function is essential.

In computational geometry, the extraction of distance information is dealt with through the construction of the Voronoi diagram. There is an abundance of literature about this topic. For isolated points, those algorithms usually start out using geometric properties (usually the Euclidean distance between points and some ad hoc observations) and intricate sorting algorithms to construct a representation (e.g., the "winged-edge" data structures [12] of the Voronoi diagram that is optimally linear in the number of data points [1, 24]. For configurations of slightly more general geometric objects, such as isolated points and linear segments (triangles in the planes), there are $\mathcal{O}(M \log M)$ algorithms for its construction and point location [27], where M is the number of segments. However, if we need to evaluate the distance at all the grid points, we still have to go through the data structures on each grid point to compute the information. The overall complexity is at least $\mathcal{O}(N)$ where N is the number of grid points. We would like to emphasize that many applications can be solved more efficiently, both in terms of development time and computation time if we accept the use of grids. One reason is that these applications are formulated and solved on the grids and require only the distance values on the grid nodes. In [22], the author proposed a fast tree-based method that efficiently uses a fast construction of Voronoi diagrams for accurate distance computation. Another efficient approach that uses Voronoi diagram for distance computation on the grids is reported in [11], in which the author considered mainly a single triangulated object.

Thus, we want to provide a framework that can be generalized to compute the distances to a variety of interfaces often encountered in the PDE-level set formulations of the problems. From the perspective of the grid-based numerical methods for PDEs, our method can thus be adopted almost effortlessly. Of course many PDE based applications are discretized on grids, e.g., the surface interpolation of [28]. Even problems involving the solution of PDEs on surfaces can use the level set formulation [3, 10]. In these cases, our algorithms provide fast and simple procedures to accomplish the task of generating the distance function. With our algorithms, the user only has to visit each grid point and perform some simple calculations. The complexity is linear in the number of grid points in the grid that resolves the interface. Only the data structures very similar or identical to the original ones used in the original applications are required. One can also think of our approach as a fast volumetric preprocessing of the distance.

We also note that once we have a second or higher order approximate distance function $d(\bar{x})$, we can easily find the Voronoi generator $\bar{x}^*$ of any given point $\bar{x}$ by $\bar{x}^* = \bar{x} - d(\bar{x})\nabla d(\bar{x})$, which requires $\mathcal{O}(1)$ operations.

The rest of this paper is organized as follows: We begin by presenting the standard PDE approximation methods for finding the distance function with an iterative scheme [23] and the Dijkstra-like fast marching method [26]. Next, we give motivation for the new method based on the geometric consideration of the reverse Huygens's principle. We employ a special sweeping strategy similar to the ones used in [4, 8, 25] to update each grid point. We then reformulate the computation of the distance function by storing the closest points on the interface on each grid point. Finally, we generalize the closest point algorithm to compute the distance accurately to triangulated surfaces commonly used in computational geometry. This provides a fast translator porting data used in computational geometry to the embedded surface formulation used by the level set methods.

For simplicity, we will first discuss the problem and the setup in $\mathbb{R}^2$. This can be directly extended to $\mathbb{R}^3$.

2. THE PROBLEM

Let Ω be a compact connected subset of $\mathbb{R}^2$ and $\Gamma \subset \Omega$ closed. The distance function $u(\bar{x})$ to Γ is defined by

$$u(\bar{x}) = \text{dist}(\bar{x}, \Gamma) := \min_{p \in \Gamma} |\bar{x} - p|$$

as a deterministic optimization problem. However, it can also be formulated as the viscosity solution of the Eikonal equation

$$|\nabla u| - 1 = 0 \quad \text{with} \quad u(\bar{x}) = 0 \quad \text{for} \quad \bar{x} \in \Gamma; \tag{1}$$

i.e., $u : \Omega \to \mathbb{R}$ is continuous,

$$u(\bar{x}) = \lim_{\epsilon \to 0} u^{\epsilon}(\bar{x}),$$

where u^{ϵ} solves the parabolic equations

$$|\nabla u^{\epsilon}| - \epsilon \Delta u^{\epsilon} = 1 \, (\epsilon > 0).$$

The existence and uniqueness of such viscosity solution $u(\bar{x})$ is well known; see e.g., [2]. Additionally, from classical PDE theory, it follows that u is increasing along the characteristics in regions where the characteristics do not cross. This is crucial to the approximation procedures described later in this paper.

Before describing our algorithms, we present some notation which shall be used later.

DEFINITION 2.1. Given $P \in \Omega$ and $\Gamma \subset \Omega$ compact, we define

$$\text{source}(P) = \{Q \in \Gamma : \text{dist}(P, \Gamma) = |P - Q|\}.$$

Clearly, if P is in the interior of a Voronoi region, source (P) contains only a unique point, and we shall call this point P^*. If P is on the Voronoi boundaries, we shall choose one element in source (P), and also call this predetermined point P^* for convenience. Since we are concerned with the distance, any such P^* works. We will also adopt the conventional term used in computational geometry with P^* being called *a Voronoi generator* [12].

We consider $\Omega \subset \mathbb{R}^2$ a rectangular region and Ω_d its discretization with uniform grid size h in both the x- and y-directions. $u_{i,j}$ refers to $u(ih, jh)$.

DEFINITION 2.2 (Neighbors). Let $E = (x_i, y_j) \in \Omega_d$. We say P is a neighbor of E if $P \in \{(x_i, y_{j\pm1}), (x_{i\pm1}, y_j)\}$.

DEFINITION 2.3 ($\sqrt{2}h$ Neighbors). We say Q is a $\sqrt{2}h$ neighbor of E if $Q \in \{(x_{i\pm1}, y_{j\pm1})\}$.

3. APPROXIMATION OF THE DISCRETIZED EQUATION—GODUNOV'S HAMILTONIAN

To put our proposed methods into context, we review briefly the distance computation from the numerical PDE point of view. This provides the motivation and the foundation of our algorithms.

Let $u(x)$ be the viscosity solution of (1). Rouy and Tourin [18] proved convergence to the viscosity solution of an iterative method solving the steady state equation (1) with the Godunov Hamiltonian approximating $|\nabla u|$

$$H_G(p_-, p_+, q_-, q_+) = \sqrt{\max\{p_-^+, p_+^-\}^2 + \max\{q_-^+, q_+^-\}^2}, \tag{2}$$

where $p_\pm = D_\pm^x u_{i,j}$, $q_\pm = D_\pm^y u_{i,j}$, $D_\pm^x u_{i,j} = \pm(u_{i\pm1,j} - u_{i,j})/h$ and accordingly for $D_\pm^y u_{i,j}$. Here $u_{i,j}$ is the approximation to the exact solution, and $x^+ = \max(x, 0)$, $x^- = -\min(x, 0)$. The Godunov Hamiltonian yields a convergent method that is first-order accurate in grid size.

Osher [13] proved that the t-level set of $u(x)$ is the zero level set of the viscosity solution of the evolution equation at time t

$$\phi_t = |\nabla\phi| \tag{3}$$

with appropriate initial conditions. So one can try to solve the time-dependent equation by the level set formulation [16] with a consistent, monotone Hamiltonian [17]. Crandall and Lions proved that the discrete solution obtained this way converges to the desired viscosity solution [7].

Tsitsiklis [26] combined the heap sorting procedure with a variant of Dijkstra's algorithm to solve the steady state equation of the more general problem

$$|\nabla\phi| - R(\bar{x}) = 0 \quad \text{for} \quad R(\bar{x}) > 0. \tag{4}$$

This was later rederived in [20] and in [9] and has become known as the fast marching method. Its complexity is $\mathcal{O}(N\log(N))$, where N is the number of grid points.

Later, Osher and Helmsen gave a criterion on general Hamiltonians so that the fast marching algorithm is applicable and extended the fast marching methods to these more general problems [15, 25].

In [4], Boué and Dupuis suggest a "sweeping" approach to solve the steady state equation, which, in our experience, results in an $\mathcal{O}(N)$ algorithm for the simple problem at hand. We point out here that this strategy when applied to compute the distance to a set of isolated points is very similar to that described in an orginal and quite early paper of Danielsson [8]. However, the author in [8] considered only points lying on the grid and did not provide an extension to the three-dimensional case. This iterative approach has been used successfully in [25] in both 2D and 3D. We shall incorporate it into our algorithms. Using this "sweeping" approach, the complexity of our algorithms drops to $\mathcal{O}(N)$ from $\mathcal{O}(N\log N)$ in the fast marching case, and the implementation of the algorithms becomes a bit easier than the fast marching method that requires heap sort. (Sweeping is not recommended for general $R(\bar{x})$ in Eq. [4].)

Since the fast marching method is, by now, well known, we will not give much detail on its implementation in this paper. In general, this involves a sorting procedure and the solution of

$$H_G(D_-^x u_{i,j}, D_+^x u_{i,j}, D_-^y u_{i,j}, D_+^y u_{i,j}) = 1 \tag{5}$$

for u_{ij} in terms of its four neighboring values.

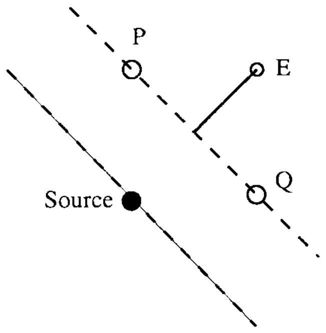


FIG. 1. Godunov solver treats everything as a plane wave.

It is important to note that this method yields the exact distance of a point inside the Voronoi regions induced by lines in 2D and planes in 3D. The normal $n(x, y) = \nabla u(x, y)/|\nabla u|$ of the distance to the lines is constant. So any consistent divided difference of u gives an exact evaluation of its derivatives.

However, this method is inconsistent for computing the distance to isolated points. For instance, if $u_{l,m} = 0$, then the computed $u_{l+1,m} = u_{l,m+1} = h$, and $u_{l+1,m+1} = (1 + 1/\sqrt{2})h$, which is the distance between $u_{l+1,m+1}$ to the line joining $u_{l+1,m}$ and $u_{l,m+1}$ plus h. That is, the Godunov solver treats the values $u_{l+1,m}$ and $u_{l,m+1}$ as though they come from a plane wave, resulting in an early arrival time (Fig. 1). The level sets of the solution look like diamond shaped polyhedra, as seen in Fig. 2, instead of perfect circles/spheres. This leads us to take more specific consideration of the geometry of the problem.

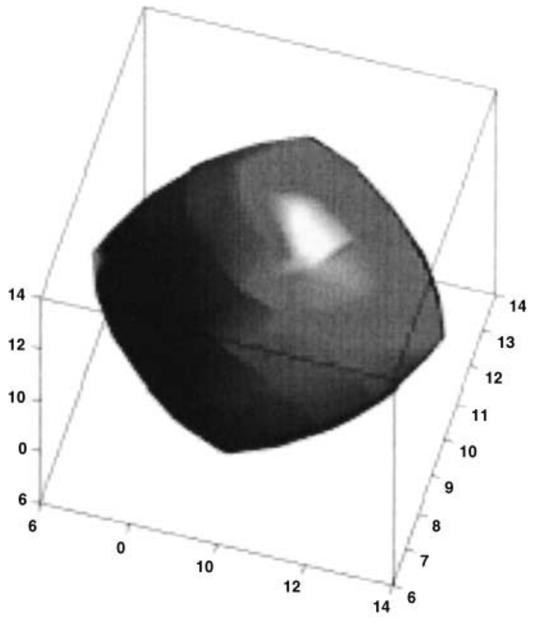


FIG. 2. The isosurface $\{u = 0.1\}$ computed by the solution of the Godunov Hamiltonian.

4. APPROXIMATION BY GEOMETRIC CONSIDERATIONS

Since in our algorithms the two-dimensional problem is only a special case of the three-dimensional case, we will now discuss our algorithms in the three-dimensional setting. To facilitate our discussion, we first introduce some more notation.

Notation 4.1. Let $P \in \Omega \subset \mathbb{R}^2$, and let $u(P)$ be the value of the viscosity solution of (1). We use the following notation for brevity,

$$C_P := \{\bar{x} \in \Omega : |\bar{x} - P| = u(P)\},$$

which is the circle with radius $u(P)$ centered at P. In 3D, we will use S_P to denote the sphere centered at P with radius $u(P)$.

Notation 4.2. Let $E \in \Omega$ be represented by $(i, j, k) \in \Omega_d \in \mathbb{R}^3$. Depending on the situation, we will use $u(E)$ and $u(i, j, k)$ interchangeably to represent the value of u at E.

DEFINITION 4.1 (Sweeping Directions in 3D). We define the following iterations going through the uniform grids $(1 : Nx - 1, 1 : Ny - 1, 1 : Nz - 1)$:

```
(x+, y+, z+) sweeping:
for k=1:Nz-1
  for j=0:Ny-1
    for i=0:Nx-1
      update $u_{i,j,k}$
(z+, y−, x+) sweeping:
for i=Nx-1:1
  for j=Ny-1:1
    for k=Nz-1:1
      update $u_{i,j,k}$
```

Following the above examples, we can define the rest of the six iterations: (y+, x+, z−), (y−, x−, z+), (x−, y+, z+), (z−, y−, x+), (x+, y−, z+), and (z−, y+, x−).

4.1. Motivation and Algorithm

We return briefly to two dimensions for simplicity for this part of the exposition. Let us first consider a single Voronoi generator E^* whose location is not explicitly known at the start of the algorithm. Suppose r_P and r_Q are the distances of two distinct points P and Q to Γ, respectively; i.e., $u(P) = r_P$, $u(Q) = r_Q$. Let $E = (x, y)$ be a point of interest with

$$\|E - E^*\| \geq \max\{u(P), u(Q)\} \tag{6}$$

a priori. Consider C_P and C_Q as in the definition: C_P is the circle of radius r_P centered at P and C_Q of radius r_Q centered at Q. The intersections of C_P and C_Q are the solutions of a quadratic equation. Denote the two intersections by V and W. It is then easy to see that either V or W is E^*. From the hypothesis that E is farther from E^* than is P or Q, we deduce that $u(E) = \max\{\|E - W\|, \|E - V\|\}$.

Thus, by enforcing this sort of upwinding decision in selecting intersections to maintain monotonicity of the solution "from" Γ, we are able to approximate the distance function.

In fact, this is the key property for the success of the aforementioned fast marching and sweeping methods.

Now we can generalize to a full 3D algorithm for distance approximation of a given set of isolated points.

ALGORITHM 1 (Spherical Intersection Solver).

1. Initialize: give the exact distance to u at grid points that are $\sqrt{2}$-neighbors of $\Gamma \subset \Omega_d$. Mark them so they will not be updated. Mark all other grid values as ∞.

2. Iterate through each grid point E with index (i, j, k) in each sweeping direction or according to the fast marching heap sort;

- Select stencils: At each grid point (i, j, k), find its neighbors P_1, P_2, and P_3 in x-, y-, and z-direction, respectively, such that $u(P_1) \leq u(i \pm 1, j, k)$, $u(P_2) \leq u(i, j \pm 1, k)$, and $u(P_3) \leq u(i, j, k \pm 1)$. Let $NBD = \{P_1, P_2, P_3\}$.
  - —Discard from NBD the element(s) on which u is ∞.
  - —If $NBD = \emptyset$. Goto 3.
  - —Otherwise, let $Q \in NBD$ such that $u(Q) = \min\{u(P) : P \in NBD\}$. Discard from NBD those P_l that are too large: $u(P_l) - u(Q) > \|P_l - Q\|$.
- If all P_1, P_2, and P_3 remain in NBD:
  - —If S_{P_1}, S_{P_2}, and S_{P_3} intersect: Let V and W be the two points of their intersection, and set $u(E) = \max\{\|E - V\|, \|E - W\|\}$.
    - $*$ If the monotone requirement $u(E) \geq u(P_i), i = 1, 2, 3$ is satisfied, skip to step 3.
    - $*$ Otherwise the intersections contain false information. Treat as if S_{P_1}, S_{P_2}, and S_{P_3} do not intersect.
  - —If S_{P_1}, S_{P_2}, and S_{P_3} do not intersect: Discard the element from NBD on which u has the greatest value, i.e., discard from NBD the P with $u(P) = \max_{p_l \in NBD} u(P_l)$. Go to the next case.
- If there are only two $P_l^{\prime}s$ remaining in NBD (assuming they are P_2, P_3 without loss of generality):
  - —If the circles C_{P_2} and C_{P_3} lying on the y-z plane intersect, let V and W be the intersections. Set $u(E) = \max\{\|E - V\|, \|E - W\|\}$. Check if the monotonicity requirement $u(E) \geq u(P_2), u(P_3)$ is satisfied. If yes, go to step 3, otherwise, treat as if C_{P_2} and C_{P_3} do not intersect.
  - —If C_{P_2} and C_{P_3} do not intersect: discard from NBD the P with $u(P) = \max\{u(P_1), u(P_2)\}$.
- If there is only one P_l remaining, simply set $u(E) = u(P_l) + h$.

3. Go to step 2.

Note that in the updating procedure, we never update the boundary of Ω_d, which is initially set to ∞. This is due to the fact that the characteristics flow only toward the boundaries; therefore, the upwinding stencils never contain the grid points on the boundaries.

4.2. Relation to "Reversing" Huygens's Principle and the Computation of Envelopes

In the above algorithms, we determine the location of one Voronoi generator in Γ by finding the intersections of two circles in 2D and three spheres in 3D. In the 2D case, we are asking basically "where is the point whose distances to the stencils P_1 and P_2 are as prescribed by $u(P_1)$ and $u(P_2)$?" If we consider the points in a neighborhood of E in the continuous case, we know that E^* has to be the intersection of all the corresponding circles centered at the neighboring points. More precisely, one can consider Γ as the envelope of the circles centered at each point in space with the radius prescribed by the distance function evaluated at that point. This can be viewed as sending circular waves from each point in the neighborhood of E at times prescribed by the distance function and asking for the envelope of the waves at a certain time. It is as if we are running Huygens's principle "backwards." This concept can certainly be extended to general Γ such as curves and surfaces.

4.3. A Hybrid Algorithm

As noted above, solving the Godunov Hamiltonian will give the exact distance to hyperplanes in their Voronoi regions. On the other hand, our methods give the exact distance for some simple point configurations and at least first-order approximation in certain circumstances (please see the section about the error analysis). We would like to create a method that makes use of the advantages of both approaches.

The idea is to adopt a consistency check of the solution obtained from the spherical method. At grid points where the result of the check implies an inconsistency of the distance information, we return to solving Godunov's Hamiltonian (2). Here, the inconsistency of the distance information means either an ambiguity (defined below) or the *disagreement* in the intersection computed from using another set of neighboring grid points as stencil. Figure 3 delineates what *disagreement* means: let S be the $\sqrt{2}$-neighbor with $u(S) = \min\{u(T)$: T is a $\sqrt{2}$ neighbor of $E\}$, and P and Q be the stencils selected from the regular neighbors of E by the spherical solver. If the three circles C_S, C_P, and C_Q do not intersect, we switch to the Godunov method.

DEFINITION 4.2. We say we have ambiguity when any of the following cases happen:

1. C_1 and C_2 do not intersect.
2. $u(E) = \max\{|E - W|, |E - V|\} < \min\{u(P), u(Q)\}$.
3. $u(E) = u(P) = u(Q)$.

ALGORITHM 2 (A 2D Hybrid Solver).

1. Initialize: Set $u_{ij} \equiv \infty$ for $(i, j) \notin \Gamma$. For each $\gamma \in \Gamma$, set $u(\gamma) = 0$. For $z \in \Gamma$ not in Ω_d, compute the exact solution at the vertices of the grid cell in which z lies.

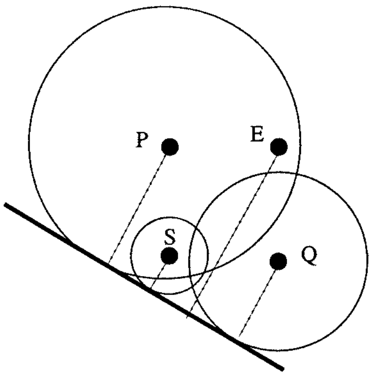


FIG. 3. A consistency check for plane wave.

2. For each sweep direction in {(x+, y+), (y−, x−), (x+, y−), (x−, y+)}, iterate through each grid point according each of the sweep directions or according to the fast marching heap sort.

3. At each grid E with index (i, j)
   - if all the neighbors are infinity, skip
   - if there is at least one noninfinity neighbor in both the x- and y-directions, let P and Q be the smallest one in each direction.

Without loss of generality, assume $u(P) > u(Q)$. Discard P if $u(P) - u(Q) > \|P - Q\|$.
   - if there is only one finite neighbor $\bar{z}$, set $u(E) = u(\bar{z}) + h$.
   - if both P and Q remain, find the points of intersection of circles C_P and C_Q.

   —If ambiguous, solve Eq. (5) for $u(E)$ instead.

   —If not ambiguous, find the smallest neighbor S from the $\sqrt{2}h$ neighbors $\{i \pm 1, j \pm 1\}$, and find the points of intersection of the circle C_S and C_P or C_Q. If there are no common points of intersection, solve Eq. (5) for $u(E)$ instead.

   —Otherwise, set $u(E) = \max\{\|E - W\|, \|E - V\|\}$, where W and V are the intersects.

We emphasize here that if the normals of the line segments in Γ do not have slopes which are integer multiples of $\pi/4$, we need to adopt the $\sqrt{2}$-neighbors of a given grid point as the regular stencils. Otherwise, the sweeping method would simply fail because during each iteration, we cannot compute and propagate the exact distance away from Γ. However, this problem may be resolved if we adopt a "generalization" of the closest point formulation (which is a current research direction and will be mentioned at the end of this paper).

4.4. The Closest Point Formulation

We see that the above algorithm tries to determine the closest point E^* of each grid point E. So if we store the generator of each grid point and propagate this closest point information somewhat "along" the characteristics, the computation of the intersections of spheres at each grid point becomes unnecessary. All we have to do is to maintain the monotonicity of the solution. This formulation is motivated by the work of Mauch [11] and the dynamic surface extension method of Steinhoff and co-workers [19, 21].

DEFINITION 4.3. Given Γ, a closest point function for Γ is $V : \mathbb{R}^3 \to \mathbb{R}^3$ such that

$$V(E) = E^*$$

for E^* predetermined as described in Definition 1.

We thus arrive at the following algorithm:

Let u be the distance function on the grids, and V be the corresponding closest point function.

ALGORITHM 3 (The Closest Point Solver).

1. Initialize: give the exact distance to u and the exact closest points to V at grid points near Γ. Mark them so they will not be updated. Mark all other grid values as ∞.
2. Iterate through each grid point E with index (i, j, k) in each sweeping direction or according to the fast marching heap sort.
3. For each neighbor P_l of E, compute $u_l^{tmp} = \|E - P_l^*\|^2$.

4. If $\|E - P_l^*\|^2 < \min_l u(P_l)$, set $u_l^{tmp} = \infty$. This is to enforce the monotonicity of the solution.

5. Set $u(E) = \min_l u_l^{tmp} = u_\lambda^{tmp}$ and $V(E) = P_\lambda^*$.

4.5. A Generalized Closest Point Formulation

In the spirit of Steinhoff's dynamic surface extension, we can define functions that map each point in $\mathbb{R}^3$ to the space of representations of surfaces. Given a proper initialization of this "generalized" closest function, we can then compute the values of the function on the whole computational domain using the "sweeping" approach mentioned above.

In computational geometry and related fields such as computer graphics and computer vision, surfaces are often triangulated and stored as a set of triangles in 3D. This motivates our next generalization of the closest point algorithm.

Given a triangle T (or line segment S) in $\mathbb{R}^3$ and a point P, the following algorithm explores the fact that we can easily compute the exact distance from P to T (or S) using elementary Euclidean geometry.

DEFINITION 4.4 (Piecewise Linear Surface Element). We define a piecewise linear surface element S to be a set of three ordered points in $\mathbb{R}^3$, and interpret the three points as the three vertices of a triangle. In a case in which two of the three points are identical, we say that $\mathcal{S}$ is a line segment. In the degenerate case where the three points are identical, we say that $\mathcal{S}$ is a point in $\mathbb{R}^3$.

We can further define the distance function of a point P and a surface element $\mathcal{S}$ as the minimum distance between P and the points on the triangle represented by $\mathcal{S}$,

$$\operatorname{dist}(P, \mathcal{S}) := \min_{y \in T}(P, y),$$

where T is the triangle defined by $\mathcal{S}$. Notice that this definition is valid even for the degenerate cases when $\mathcal{S}$ is a line segment or a point.

DEFINITION 4.5 (Generalized Closest Point Function). We say W is a generalized closest point function if

$$W : \mathbb{R}^3 \to \{\text{the space of piecewise linear surface elements}\}.$$

Let Γ be represented by a set of piecewise linear surface elements in $\mathbb{R}^3$. Let u be the distance function on the grids, and W be the corresponding generalized closest point function.

ALGORITHM 4 (The Generalized Closest Point Solver).

1. Initialize: give the exact distance to u and the exact surface elements to W at grid points near Γ. Mark them so they will not be updated. Mark all other grid values as ∞.

2. Iterate through each grid point E with index (i, j, k) in each sweeping direction or according to the fast marching heap sort.

3. For each neighbor P_l of E, compute $u_l^{tmp} = \operatorname{dist}(E, W(P_l))$.

4. If $\operatorname{dist}(E, W(P_l)) < \min_l u(P_l)$, set $u_l^{tmp} = \infty$. This is to enforce the monotonicity of the solution.

5. Set $u(E) = \min_l u_l^{tmp} = u_\lambda^{tmp}$ and $W(E) = W(P_\lambda)$.

We show the results at the end of this paper. See Figs. 11 and 12.

4.6. Analysis of the Algorithms

Our experience shows that a satisfactory approximation can be reached after one set of sweeping iterations. We have not yet been able to prove convergence. However, we do see the limits to grid-based computation of distance functions.

Both the spherical intersection method and the closest point approach described above are in essence trying to determine the closest point of E using the closest points of the neighbors of E. In order to get the exact solution ($u(E)$ or E^*), we need the following property:

$$E^* \in \{P_i^* : P_i \text{ the neighbors of } E\}.$$

However, this is not always possible. In fact, near the Voronoi boundary, we might reach the limits of our methods using the standard 4-point stencil. See Fig. 4.

Consider the following situation: Suppose E^* falls into the first quadrant (I). Let P be the grid such that $\text{dist}(P, E^*) < \text{dist}(E, E^*)$. We form a circle C_E centered at E, passing through E^*, and C_P centered at P, also passing through E^*. Let D_E, D_P be the corresponding disks. Then if $\Upsilon = \Gamma \cap \{D_P \backslash D_E\} \neq \emptyset$, then P^* *has* to be in it, because any point in Υ satisfies

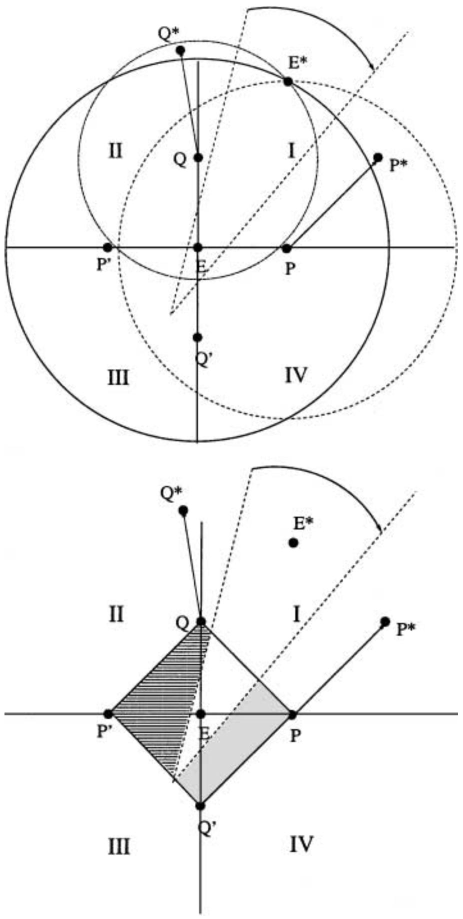


FIG. 4. An example of the limit of grid-based distance approximation.

the property that it is closer to P than is E^*. Similarly, we can construct the same scenario for Q.

As illustrated in Fig. 4, E is the only point in the neighborhood that is in the Voronoi region generated by E^*; i.e., $E^* \not\in \{P^*, Q^*, (P')^*, (Q')^*\}$. Hence, our updating procedure will not compute the exact distance at E in this case.

4.6.1. *Local Error Analysis*

Let $r_E := \|E - E^*\|$ and $\tilde{r}_E := \|E - P^*\|$. We want to know what the maximum error $e := \tilde{r}_E - r_E$ can be, given exact values of u and V at the neighbors of E. From the illustration above, we can easily see that $\max \tilde{r}_E < \max \|E^* - P\| + h$, and

$$\max \|E^* - P\| < \sqrt{r_E^2 + h^2} = r_E + h^2 \frac{1}{2r_E} + \mathcal{O}(h^4).$$

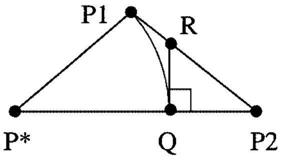
Thus, $0 \le \tilde{r}_E - r_E < h$. So the local error is $\mathcal{O}(h)$.

4.6.2. *The Selection of Stencils*

In the algorithms described above, we have some criteria which eliminate invalid stencils. In the case where the value on a grid point is still infinity, we know automatically that it cannot lie in a valid stencil. In the case where the grid point carries "obviously incorrect" distance information, we may make use of the observation of the constant travel time of the eikonal equations: if the two neighboring grid points carry finite values and the difference of the values of the distance function on them is too large, we have an incompatibility. Since we want to enforce the upwinding stencil selection, we favor the grid point with the smaller value. Therefore, we have the following propositions:

PROPOSITION 4.1. *Let P_1 and P_2 be two neighbors of E, and $u(P_2) - u(P_1) > \|P_2 - P_1\|$. Then $P_1^* \neq P_2^*$.*

Proof. Assume on the contrary that $P_1^* = P_2^* = P^*$ and $u(P_2) > u(P_1)$. Let Q be a point on $\overline{P^*P_2}$ such that $\|Q - P^*\| = u(P_1)$. Let R be a point on $\overline{P_1P_2}$ such that $\Delta\, P_2QR$ is a right triangle with hypotenuse $\overline{RP_2}$.



Then $\|P_2 - P_1\| = |\overline{P_1P_2}| \ge |\overline{RP_2} \ge |\overline{QP_2}| = u(P_2) - u(P_1)$. Thus, we have a contradiction. The proof is complete. ■

This proposition shows that if P_1^* is correct and $u(P_2) - u(P_1) > \|P_2 - P_1\|$, then the information we hold on P_2 is not correct.

5. RESULTS

We first run our algorithms to approximate the distance function of a single Voronoi generator at the center of a $11 \times 11 \times 11$ grid. As shown in Table I, the spherical method

TABLE I
A Comparison of Test Case Results

$11 \times 11 \times 11$ ($h = 0.1$)	max error	L_1 error
Godunov method	0.114	0.091
Spherical method	1.53e-15	2.99e-16
Closest point method	0	0

gives an almost exact result with error close to machine zero. The closest point method gives the exact solution. Since we need to take a square root repeatedly in the first case, we expect to lose some accuracy. However, as demonstrated by the numerical results, the whole process of the spherical method appears to be stable.

We then tested our generalized closest point solver in both 2D and 3D with several different configurations. In the test case, our generalized closest point algorithm is exact to machine precision. We will describe our test cases in more detail in a following section.

In the following sections, we will "visualize" the distance functions by their level sets. In 2D, we plot the contour lines, and in 3D, we plot the isosurfaces.

5.1. A Comparison of Results in 2D

The following two sets of figures, Figs. 5 and 6, are computed in the sweeping fashion. We can clearly see that the hybrid algorithm gives straight contour lines in the Voronoi regions generated by straight line segments. In other regions, the contour lines have a nice circular shape.

5.2. A Comparison of Results

We run the Godunov method and our methods to approximate the distance function to a set of 10 randomly selected isolated points. From Tables III and IV, we can see that the

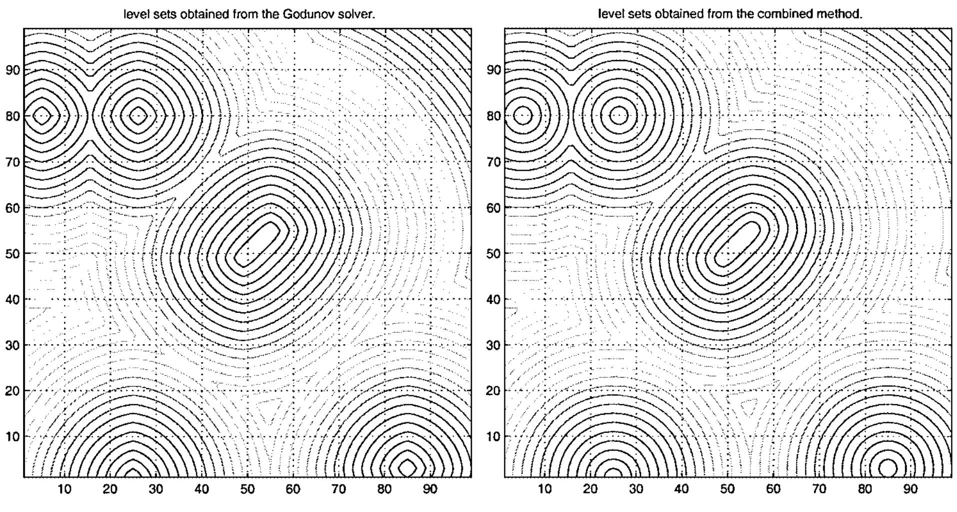


FIG. 5. A comparison of results by Godunov method and spherical intersection method in 2D.

TABLE II
Godunov Method

Grid	max error	Convergence rate	L_1 error	Convergence rate
$10 \times 10 \times 10$ ($h = 0.01$)	0.008555		5.223e-6	
$20 \times 20 \times 20$	0.005723	0.58	3.343e-6	0.64
$40 \times 40 \times 40$	0.003656	0.65	2.059e-6	0.70
$80 \times 80 \times 80$	0.002311	0.66	1.248e-6	0.72

TABLE III
Spherical Method

Grid	max error	Convergence rate	L_1 error	Convergence rate
$10 \times 10 \times 10$ ($h = 0.01$)	0.006869		1.500e-6	
$20 \times 20 \times 20$	0.002855	1.3	0.201e-6	2.9
$40 \times 40 \times 40$	0.001355	1.1	0.041e-6	2.3
$80 \times 80 \times 80$	0.000674	1.0	0.011e-6	1.9

TABLE IV
Closest Point Method

Grid	max error	Convergence rate	L_1 error	Convergence rate
$10 \times 10 \times 10$ ($h = 0.01$)	0		0	
$20 \times 20 \times 20$	0	∞	0	∞
$40 \times 40 \times 40$	0	∞	0	∞
$80 \times 80 \times 80$	0	∞	0	∞

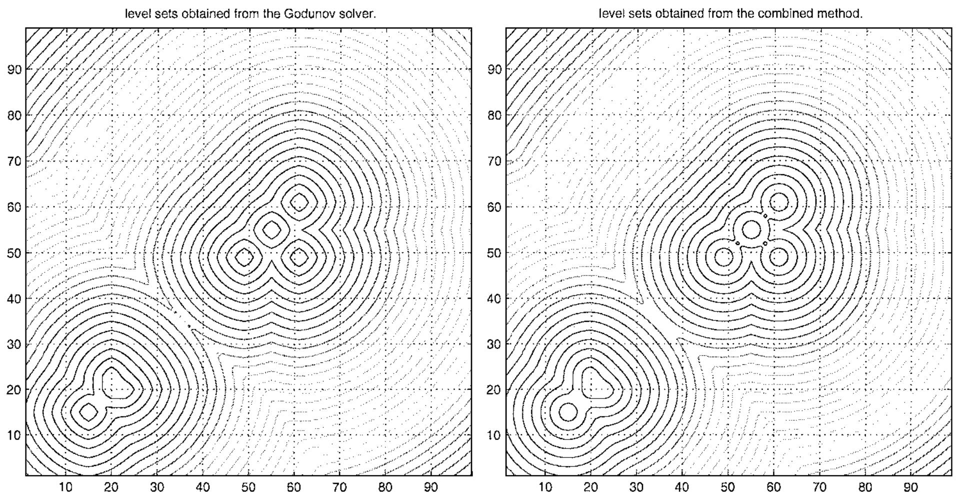


FIG. 6. A comparison of results by Godunov method and spherical intersection method in 2D.

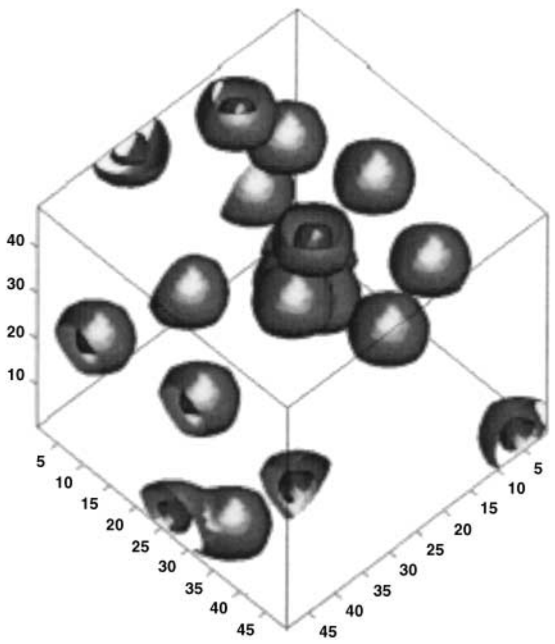


FIG. 7. Isosurfaces computed by the Godunov solver.

spherical method converges numerically with a second-order rate while the closest point solver gives the exact distance function. We see that both methods converge faster than the Godunov method (Table III). There is a loss of accuracy in the solution obtained by using the spherical method. The error comes from two major sources. Within each Voronoi region, the computed solution loses accuracy from solving the quadratic equations. Near the Voronoi boundaries, it is not always true that the grid value we are updating has to be greater than or equal to the values of the stencils. By enforcing monotonicity of our solution near the Voronoi boundaries, we may therefore create a slight perturbation to the true solution. This perturbation propagates outwards from the Voronoi boundaries during later updating iterations. In the closest point algorithm for isolated points, we enforce a slightly relaxed monotonicity (steps 4 and 5) on solution. Unless the situation mentioned in previous analysis occurs, we will obtain the exact distance function.

In Figs. 7–10 we show the comparison of the isourfaces computed from the solutions of the Godunov solver and our methods. We have a $50 \times 50 \times 50$ grid with uniform grid size $h = 0.01$ and 20 randomly selected isolated points. We can clearly see that the spherical intersection method gives much better results in terms of both L_1 error and “visual” examination of the isosurfaces.

5.3. Results of the Generalized Closest Point Sweeping

Figure 11 shows the contour lines of the computed distance function to a set of a closed polygon, two line segments, and two isolated points. Table V shows, in turn, that the generalized closest point algorithm computes the distance to this set of objects to the machine precision after one sweeping iteration. Figure 12 shows the isosurfaces rendered from the computed distance function to a set of two polyhedra, three line segments, and two isolated points.

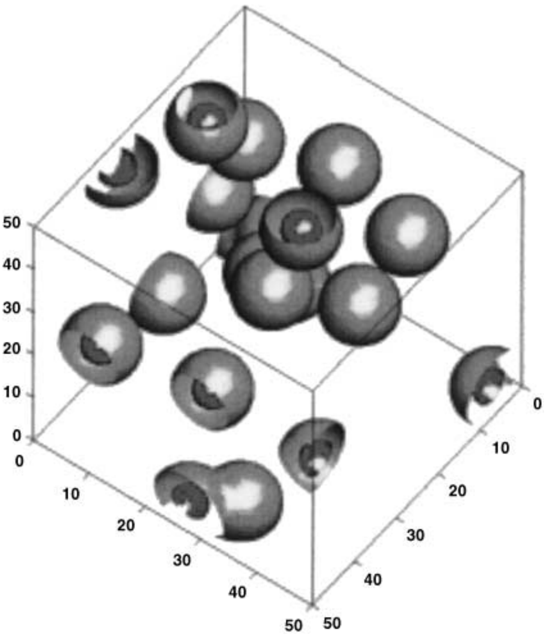


FIG. 8. Isosurfaces computed by the spherical intersection method.

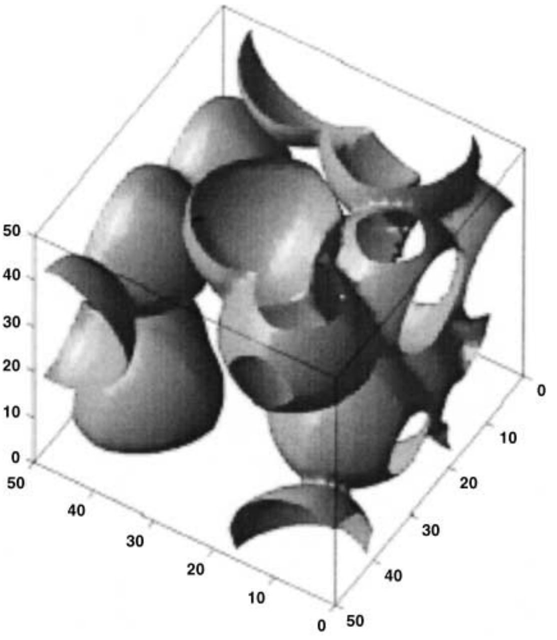


FIG. 9. Isosurfaces computed by the Godunov solver.

TABLE V
The Generalized Closest Point Method

Grid	max error	Convergence rate	L_1 error	Convergence rate
50×50 ($h = 0.02$)	0		0	
100×100	0	∞	0	∞
200×200	0	∞	0	∞

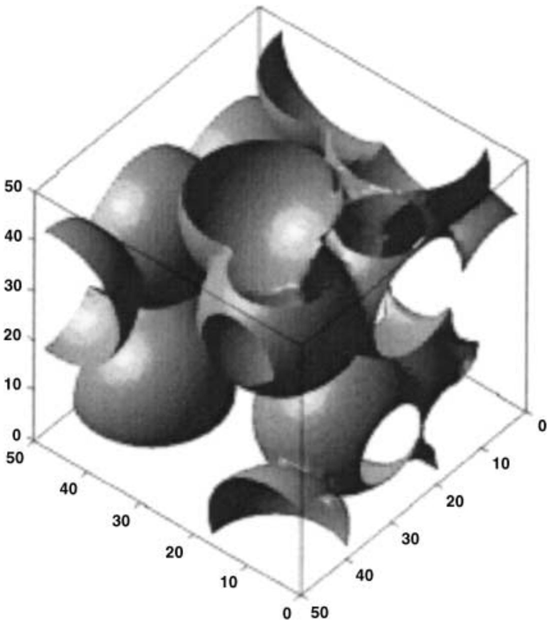


FIG. 10. Isosurfaces computed by the spherical intersection method.

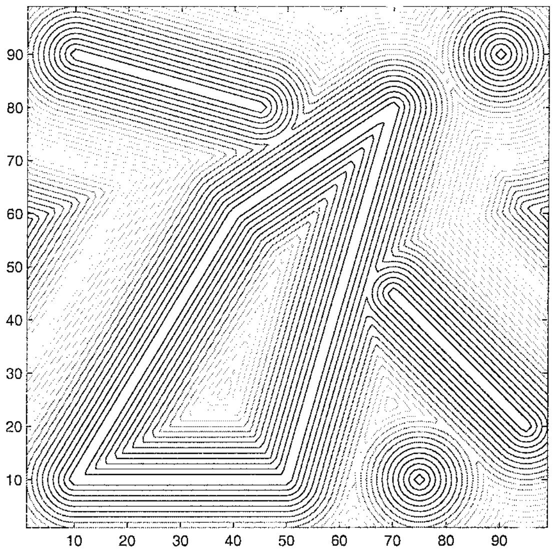


FIG. 11. Contour lines for the distance to a set of piecewise linear objects.

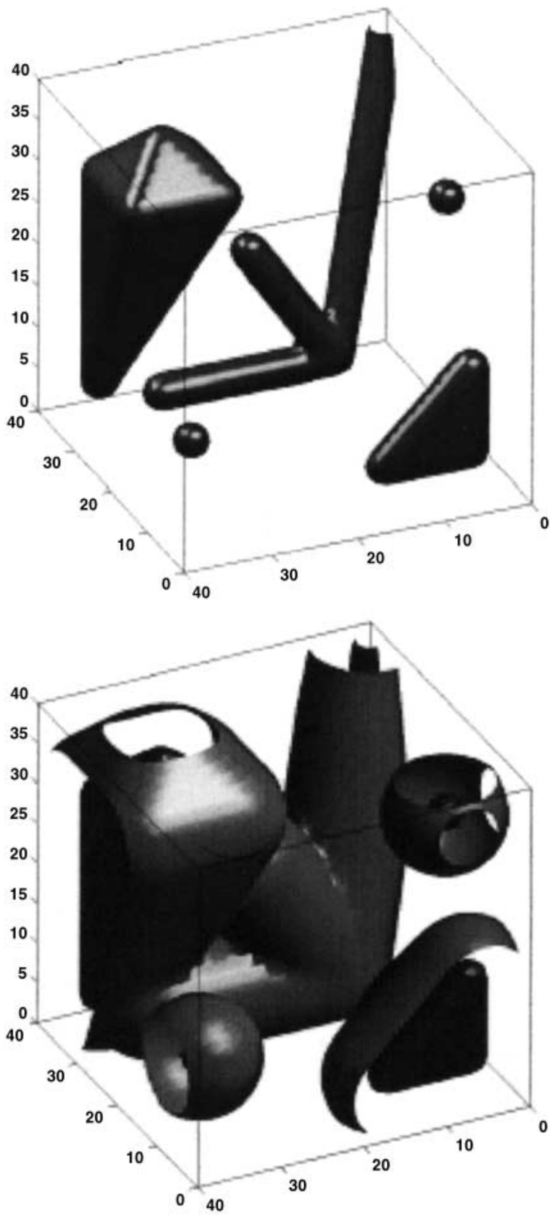


FIG. 12. Isosurfaces of the distance to triangulated surfaces and isolated points.

6. CONCLUSION

In this paper, we have introduced two grid based algorithms for computing the distance function to a given configuration of isolated points, which are then generalized to compute the distance function to piecewise linear objects. They prove to be accurate, fast ($\mathcal{O}(N)$), and easy to implement. The key to the algorithms is the combination of "sweeping," the upwinding-based stencil selection, and smarter geometric interpretation. Putting them all together leads to accurate and viscosity-solution-compliant approximations. The generalized closest point solver provides a quick translator of the triangulated data commonly used in computational geometry to the level set formulation. It also suggests a more general framework for the distance computation to more general surfaces.

7. FUTURE WORK

Our first remark is that our algorithms (both spherical intersection and (generalized) closest point sweeping) can easily be generalized to other grid geometries since we are not approximating partial derivatives on the stencil. This topic will be investigated by the author.

Furthermore, from the adaptation of the closest point formulation, we have recognized the potential of computing the distance function to more general geometrical objects. We are currently seeking a generalized closest point formulation for computing more general surfaces in 3D. As indicated in the previous section, instead of storing only the closest point, we can also store a representation of the surface element;

$$V(E) \to (E^*, \text{ surface element}).$$

The "surface element" can be for example the curvature or a NURB description of the surface. The challenge is to compute the exact distance to a given surface element and to derive the "upwinding" criteria for propagating the surface information throughout the grids. A preliminary result of one method using a combination of Newton's method and sweeping iterations on a smooth parametrized curve segment is shown in Fig. 13.

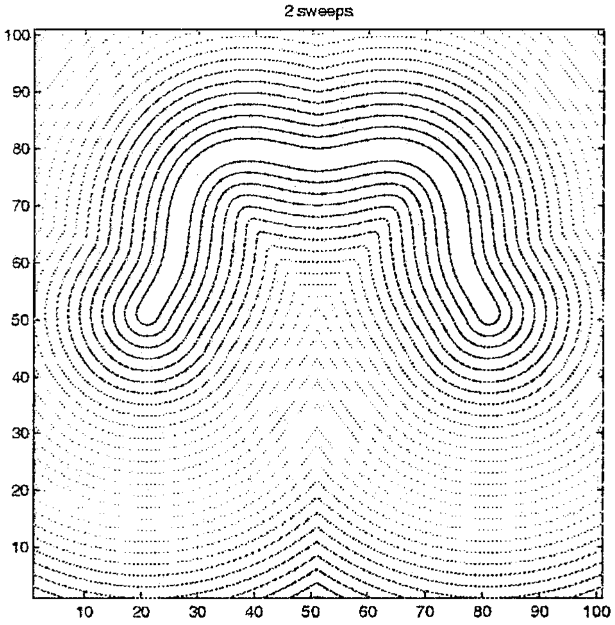


FIG. 13. The level contours of the distance function to a smooth parametrized curve constructed using Newton's method and sweeping.

ACKNOWLEDGMENTS

The author thanks L. T. Cheng and P. Burchard for their very useful conversations on this topic.

REFERENCES

1. A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor, A linear-time algorithm for computing the Voronoi diagram of a convex polygon, *Discrete Comput. Geometry* **4**, 591 (1989).
2. G. Barles, *Solution de Viscosité des Équations de Hamilton–Jacobi* (Springer-Verlag, Berlin/New York, 1994).
3. M. Bertalmio, L. T. Cheng, S. Osher, and G. Sapiro, Variational problems and partial differential equations on implicit surfaces, *J. Comput. Phys.* **174**, 759–780 (2001).
4. M. Boué and P. Dupuis, Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control, *SIAM J. Numer. Anal.* **36**(3), 667, 1999 (electronic).
5. W. Brostow, J. P. Dussault, and B. L. Fox, Construction of Voronoi polyhedra, *J. Comput. Phys.* **29**, 81 (1978).
6. S. Chen, B. Merriman, M. Kang, R. E. Caflisch, C. Ratsch, L.-T. Cheng, M. Gyure, R. P. Fedkiw, and S. Osher, Level set method for thin film epitaxial growth, *J. Comput. Phys.* **167**, 475 (2001).
7. M. G. Crandall and P. L. Lions, Two approximations of solutions of Hamilton–Jacobi equations, *Math. Comp.* **43**, (1984).
8. Per-Erik Danielsson, Euclidean distance mapping, *Comp. Graph. Image Proc.* **14**, 227 (1980).
9. J. Helmsen, E. Puckett, P. Colella, and M. Dorr, Two new methods for simulating photolithography development in 3D, *Proc. SPIE 2726*, 253 (1996).
10. H. Jin, A. Yezzi, and S. Soatto, Stereoscopic shading: Integrating multi-frame shape cues in a variational framework, *IEEE Computer Vision Pattern Recog.*, (2000).
11. S. Mauch, The closest point transform, in Preparation; *see* http://www.its.caltech.edu/sean.
12. A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations Concepts and Applications of Voronoi Diagrams*, (Wiley, New York, 1992).
13. S. Osher, A level set formulation for the solution of the Dirichlet problem for Hamilton–Jacobi equations, *SIAM J. Math. Anal.* **24**(5), 1145 (1993).
14. S. Osher and Ronald P. Fedkiw, Level set methods: An overview and some recent results, *J. Comput. Phys.* **169**, 463–502 (2001).
15. S. Osher and J. Helmsen, A generalized fast algorithm with applications to ion etching, in Preparation.
16. S. Osher and J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
17. S. Osher and C. W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, *SIAM J. Numer. Anal.* **28** (4), 907 (1991).
18. Elisabeth Rouy and Agnés Tourin, A viscosity solutions approach to shape-from-shading, *SIAM J. Numer. Anal.* **29**(3), 867 (1992).
19. S. J. Ruuth, B. Merriman, and S. Osher, A fixed grid method for capturing the motion of self-intersecting interfaces and related PDes, *J. Comput. Phys.* **163**, 1 (2000).
20. J. A. Sethian, Fast marching level set methods for three-dimensional photolithography development, *SPIE 2726*, 261 (1996).
21. J. Steinhoff and M. Fan, Eulerian computation of evolving surfaces, curves and discontinuous fields, Technical Report 37388, (University of Tennessee Space Institute, 1998).
22. John Strain, Fast tree-based redistancing for level set computations, *J. Comput. Phys.* **152**, 664 (1999).
23. M. Sussman, P. Smereka, and S. Osher, A level set method for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
24. M. Tanemura, T. Ogawa, and N. Ogita, A new algorithm for three-dimensional Voronoi tessellation. *J. Comput. Phys.* **51**, 191 (1983).

25. Y. R. Tsai, L. T. Chang, S. Osher, and H. K. Zhao, Fast sweeping algorithms for a class of Hamilton–Jacobi equations, UCLA CAM Report 01-27 (2001), submitted for publication.

26. J. Tsitsiklis, Efficient algorithms for globally optimal trajectories, *IEEE Trans. Automatic Control* **40**(9), 1528 (1995).

27. C. K. Yap, An $O(n \log n)$ algorithm for the Voronol diagram of a set of simple curve segments, *Discrete Comput. Geom.* **2**(4), 365 (1987).

28. H. K. Zhao, S. Osher, M. Merriman, and M. Kang, Implicit and non-parametric shape reconstruction from unorganized points using variational level set method, *Computer Vision and Image Understanding* **80**, (2000).