Journal of Computational Physics **151,** 616–648 (1999)
Article ID jcph.1999.6205, available online at http://www.idealibrary.com on IDEAL®

Tree Methods for Moving Interfaces

John Strain[1]

Department of Mathematics, University of California, 970 Evans Hall, Number 3840, Berkeley, California 94720-3840
E-mail: strain@math.berkeley.edu

Received August 19, 1998; revised January 19, 1999

Fast adaptive numerical methods for solving moving interface problems are presented. The methods combine a level set approach with frequent redistancing and semi-Lagrangian time stepping schemes which are explicit yet unconditionally stable. A quadtree mesh is used to concentrate computational effort on the interface, so the methods move an interface with N degrees of freedom in $O(N \log N)$ work per time step. Efficiency is increased by taking large time steps even for parabolic curvature flows. The methods compute accurate viscosity solutions to a wide variety of difficult moving interface problems involving merging, anisotropy, faceting, and curvature.

Key Words: moving interfaces; level sets; adaptive mesh refinement; fast algorithms; semi-Lagrangian methods; CIR scheme; motion by curvature.

CONTENTS

[1] Research supported by NSF Young Investigator and SCREMS Awards and by Air Force Office of Scientific Research Grant FDF 49620-93-1-0053.

0021-9991/99 $30.00

1. INTRODUCTION

Moving interface problems occur frequently in applications, involve complex topology, merging, faceting, and curvature, and challenge standard numerical methods. We present efficient adaptive numerical methods for solving these problems. Our methods merge and break interfaces automatically via a level set approach with frequent redistancing. Quadtree meshes resolve the interface with almost optimal efficiency: we move an N-element interface in $O(N \log N)$ work per step. Semi-Lagrangian time stepping schemes allow large time steps with unconditional stability. Fast redistancing algorithms maintain a robust numerical approximation with minimal computational effort.

Section 2 of this paper defines moving interface problems and reviews the level set approach. Section 3 discusses semi-Lagrangian time stepping schemes and summarizes their application to level set equations on a uniform mesh. Section 4 presents the properties of quadtree meshes that we use, and Section 5 develops our tree methods for moving interfaces. Section 6 validates these methods with numerical examples including geometric motions which merge faceted interfaces under anisotropic curvature-dependent velocities. Section 7 draws conclusions and discusses future extensions and applications.

2. MOVING INTERFACES AND LEVEL SETS

This section presents standard background material on moving interface problems and the level set approach. Subsection 2.1 defines these problems and describes examples such as passive transport, unit normal velocity, and anisotropic curvature-dependent flow. Subsection 2.2 converts general moving interface problems into level set equations on a fixed domain and reviews their solution by the level set approach.

2.1. *Moving Interfaces*

A general moving interface is the boundary $\Gamma(t) = \partial\Omega(t)$ of a set $\Omega(t) \subset \mathbf{R}^d$ depending on time t. If Ω is sufficiently smooth, then $\Gamma(t)$ has an outward unit normal N and a normal velocity V at each point, which can be calculated from standard geometric formulas found in [23]. *A moving interface problem* is a closed system of equations which specifies V as a functional of Γ, possibly in a highly indirect and nonlocal way. Some representative solutions of the following specific moving interface problems are shown in Fig. 1.

Passive transport. An interface is transported in an ambient flow which is independent of Γ. Thus a velocity field $F(x, t)$ is given on $\mathbf{R}^d$ and $\Gamma(t)$ moves with normal velocity $V = N \cdot F$.

Unit normal velocity. The simplest geometric flow moves $\Gamma(t)$ along its normal with velocity $V = 1$. Nonconvex interfaces produce complex merging and cornering patterns under this flow.

Anisotropic curvature-dependent velocity. A more general geometric motion has normal velocity

$$V(x, t) = R + \epsilon \cos(K\theta + \theta_0) + (R' + \epsilon' \cos(K'\theta + \theta_0'))C, \tag{1}$$

where $\cos\theta = N \cdot e_1$ is the cosine of the angle between the normal vector and the positive x-axis. These velocity fields produce faceted interfaces merging in complex anisotropic patterns and are often used as simplified models in materials science [22].

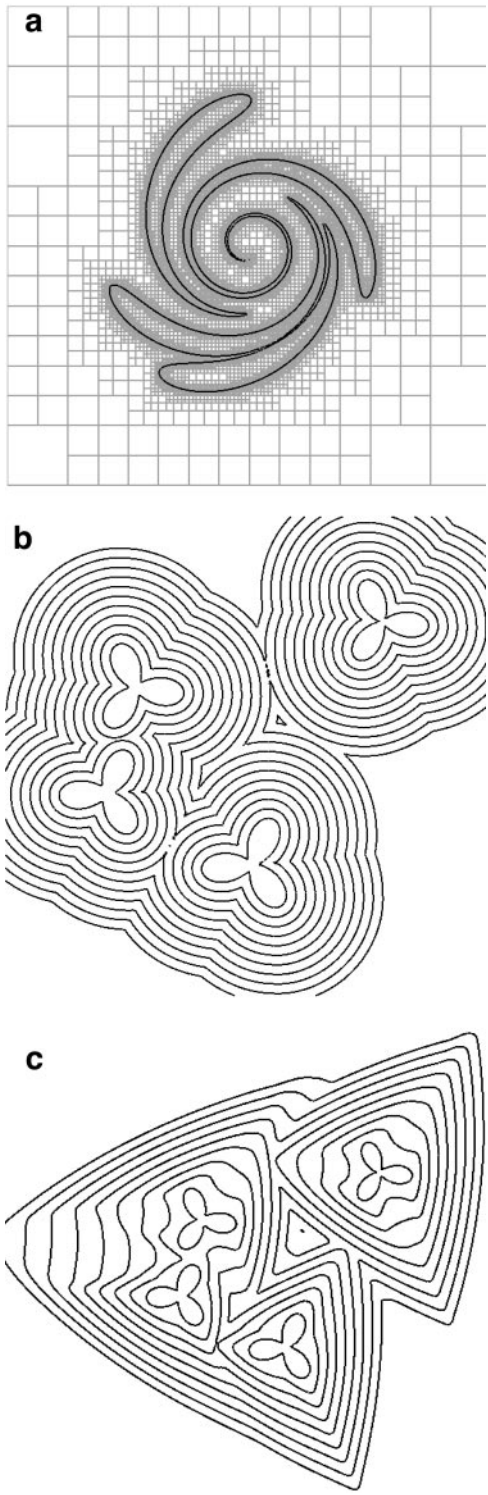

FIG. 1. Sample moving interface problems: (a) initially circular bubbles after passive transport in a shearing flow, (b) merging of complex interfaces with unit normal velocity, and (c) crystalline facets developing under the threefold anisotropic curvature-dependent velocity defined in Eq. (1).

Crystal growth. Many industrial problems involve moving interfaces between different phases of a material. The interface between a growing solid crystalline material and its liquid or gaseous phase, for example, has been modeled by a Stefan-type problem

$$u_t = \nabla u \qquad \text{off } \Gamma(t) \tag{2}$$

$$u = -\epsilon C \qquad \text{on } \Gamma(t), \tag{3}$$

where the temperature field u is unknown and the interface Γ moves with normal velocity V equal to the jump in the normal derivative of u. See [5] for physical background and [13, 17, 11] for samples of the many numerical methods developed for this problem.

2.2. *The Level Set Approach*

The main difficulty in moving interfaces is the correct handling of merging, breaking, and other topological changes. We can overcome this difficulty by reformulating moving interface problems as "level set equations" on a fixed domain, using the *zero set*

$$\Gamma(t) = \{x \in \mathbf{R}^d : \varphi(x, t) = 0\} \tag{4}$$

of an arbitrary function $\varphi : \mathbf{R}^d \times \mathbf{R} \to \mathbf{R}$, such as the signed distance to $\Gamma(t)$:

$$\varphi(x, t) = \pm \min_{y \in \Gamma(t)} \|x - y\|. \tag{5}$$

(For example, Fig. 2 shows a hexagon in the plane and the corresponding signed distance function φ.) We choose $\varphi > 0$ in $\Omega(t)$, so the outward unit normal vector and normal velocity are given by [23]

$$N = \nabla\varphi / \|\nabla\varphi\|, \tag{6}$$

$$V = \varphi_t / \|\nabla\varphi\|. \tag{7}$$

Given an extension of the vector normal velocity VN to a function $F(x, t)$ on $\mathbf{R}^d$, Eq. (7)

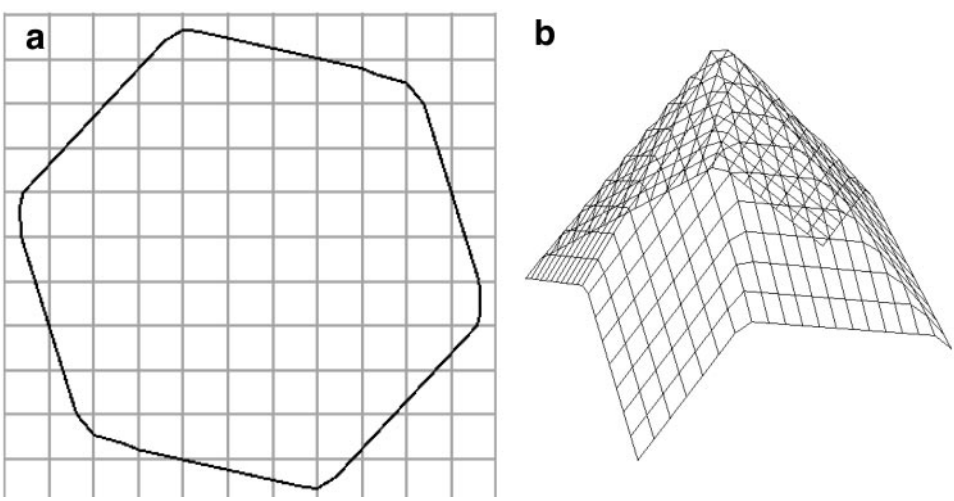

FIG. 2. The correspondence between (a) a hexagonal interface and (b) the signed distance φ to the interface.

implies a partial differential equation—the "level set equation"—which moves Γ by evolving φ:

$$\varphi_t - F \cdot \nabla\varphi = \varphi_t - (F \cdot N)\|\nabla\varphi\| = 0. \tag{8}$$

Equation (8) moves every level set of φ with the extended velocity F, and in particular moves the zero set $\Gamma(t)$ with the correct velocity VN. This approach to moving interfaces embeds the topology in φ rather than $\Gamma(t)$ and automatically handles merging, breaking, and other topological changes. The moving interface problems of Subsection 2.1 can be reformulated as the following level set equations.

Passive transport. For passive transport, F is already defined on $\mathbf{R}^d$ and is a natural extension of VN. The level set equation becomes a linear hyperbolic partial differential equation (PDE)

$$\varphi_t - F(x,t) \cdot \nabla\varphi = 0. \tag{9}$$

Unit normal velocity. With N extended by Eq. (7), motion with unit normal velocity becomes a nonlinear hyperbolic PDE

$$\varphi_t - \|\nabla\varphi\| = 0. \tag{10}$$

Curvature-dependent velocity. The velocity defined by Eq. (1) yields

$$\varphi_t - (R + \epsilon\cos(K\theta + \theta_0))\|\nabla\varphi\| = (R' + \epsilon'\cos(K'\theta + \theta_0'))\nabla \cdot (\nabla\varphi/\|\nabla\varphi\|)\|\nabla\varphi\|. \tag{11}$$

Here $\cos\theta = \varphi_x/\|\nabla\varphi\|$ and we have used the curvature formula $C = -\nabla \cdot N$ from [23]. Equation (11) is a mixed hyperbolic-parabolic PDE which is singular where $\nabla\varphi$ vanishes.

The level set approach moves $\Gamma(t)$ via the level set equation (8). An initial level set function $\varphi(x, 0)$ and an extended velocity field F are built, the level set equation (8) is solved numerically, and the solution $\varphi(x, t)$ is contoured when $\Gamma(t)$ is required. The approach was introduced in [9], and an extensive recent survey is [12]. Its main advantage is the natural treatment of dynamic topology shown in Fig. 3.

There are some potential difficulties with the level set approach. It can be more expensive since it goes up a dimension, particularly if uniform meshes are used. Extending the velocity off $\Gamma(t)$ can be difficult. One must be careful to obtain the correct "viscosity solution" of Eq. (8), by using an appropriate solver for the level set equation [12]. The approach is not naturally *modular*: a new code must be written for each new problem to be solved, since the velocity evaluation is intertwined with the moving interface code by velocity extension.

Our methods combine a level set approach with an adaptive quadtree mesh and are shown experimentally to obtain the correct viscosity solution for passive transport and geometric problems where velocity extension is straightforward. The adaptivity of our methods eliminates the added cost of going up a dimension. A general velocity extension is developed and used to build general modular methods in [20].

3. SEMI-LAGRANGIAN LEVEL SET METHODS

The semi-Lagrangian level set methods introduced in [19] solve level set equations on a uniform mesh with semi-Lagrangian time stepping schemes. The level set equations

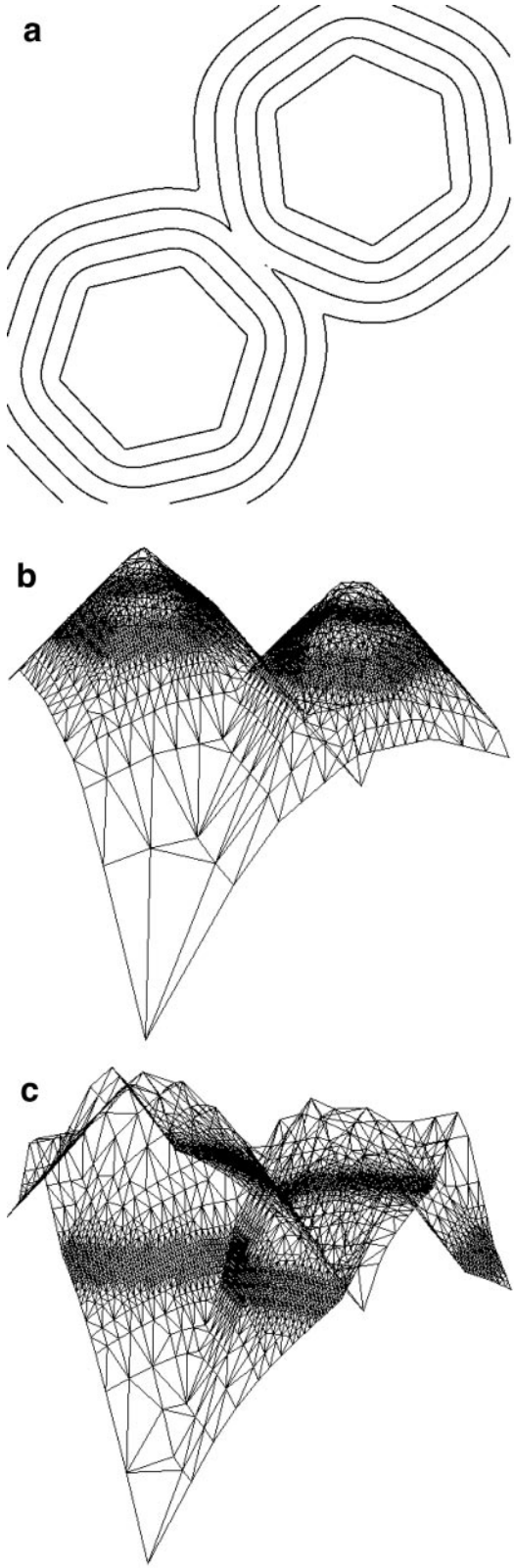

FIG. 3. (a) Two hexagons moving with constant normal velocity grow and merge. The corresponding signed distance function is plotted over a triangulated quadtree (see Section 4) at (b) initial and (c) final times.

handle the dynamic topology of the moving interface, while semi-Lagrangian schemes allow large time steps $k = O(h)$ even for parabolic problems like curvature flows. These methods are robust and accurate, but the uniform mesh spends too much effort far from the interface. We implement semi-Lagrangian level set methods on a quadtree mesh to concentrate computational effort near the interface, attaining accuracy comparable to a uniform mesh method at far less cost. In this section, we review the simplest semi-Lagrangian time stepping scheme, discuss its convergence theory, and summarize the methods of [19].

3.1. *The CIR Scheme*

The linear hyperbolic PDE

$$\varphi_t - F(x,t)\cdot\nabla\varphi = 0 \tag{12}$$

propagates φ values along the characteristic curves $s(t)$ defined by

$$\dot{s}(t) = -F(s(t),t). \tag{13}$$

Thus we can find φ values at any time t by finding the characteristic curve passing through (x,t) and following it backwards to some previous point (x_0,t_0) where the value of φ is known: then $\varphi(x,t) = \varphi(x_0,t_0)$. This observation forms the basis of the "backward characteristic" or "CIR" scheme due to Courant, Isaacson, and Rees [2], which is the simplest semi-Lagrangian scheme. Given φ at time t_n, CIR approximates $\varphi(x,t_{n+1})$ at any point x at time $t_{n+1} = t_n + k$ by evaluating the velocity $F(x,t_n)$, approximating the backward characteristic through x by a straight line

$$x + (t_{n+1} - t)F(x,t_n) \approx s(t), \tag{14}$$

and interpolating φ linearly at time t_n to the point

$$x + kF(x,t_n) \approx s(t_n). \tag{15}$$

Then $\varphi(x,t_{n+1})$ is set equal to the interpolated value.

General semi-Lagrangian time stepping schemes are built along similar lines with higher-order accurate time stepping and interpolation and are widely used in atmospheric science [15, 14].

3.2. *Convergence*

For linear PDEs, the Lax–Richtmyer equivalence theorem [6] guarantees that CIR will converge to the exact solution as $k, h \to 0$ if it is stable and consistent. Stability is unconditionally guaranteed since each new value $\varphi(x,t_{n+1})$ is a single linearly interpolated value of φ at time t_n.

Consistency, however, is conditional. The truncation error of CIR is

$$\tau = O\left(\frac{h^2}{k}\right) + O(k), \tag{16}$$

due to the $O(h^2)$ error in linear interpolation over $O(1/k)$ steps plus the $O(k)$ due to freezing F and approximating the characteristics by straight lines. Thus CIR is consistent

to $O(k)$ if a condition like $k \geq O(h)$ is satisfied, contrary to the usual hyperbolic condition $k \leq Ch$. This condition is extremely convenient, because $k = O(h)$ balances time and space resolution in this first-order accurate scheme.

CIR converges for Lipschitz solutions of nonlinear PDEs but moves shock solutions of conservation laws at the wrong speed because CIR is not in conservation form. Thus semi-Lagrangian schemes such as CIR have been applied mainly to problems in atmospheric science where shocks are absent. Since level set equations have no shocks, CIR is a natural scheme for moving interfaces.

3.3. *Semi-Lagrangian Level Set Methods*

The semi-Lagrangian CIR scheme was applied to level set equations in [19], yielding semi-Lagrangian level set methods on a uniform mesh. Convergence was heuristically discussed and experimentally verified for many moving interface problems involving passive transport, geometry, dynamic topology, faceting, and curvature. Convergence of these methods is straightforward for passive transport and first-order geometry where the level set equation is hyperbolic. For parabolic problems such as curvature flows, the main issue is the Courant–Friedrichs–Lewy (CFL) condition which restricts the time step of most explicit methods by $k \leq O(h^2)$ to ensure information propagates correctly. Semi-Lagrangian level set methods are unconditionally stable and can satisfy the CFL condition by nonlocal velocity evaluation, permitting convergence with large time steps $k = O(h)$ even for parabolic problems. While their convergence theory is still in progress, the combination of experimental evidence with the following heuristics indicates that these methods can converge correctly.

The domain of dependence of the CIR solution $\varphi(x, t_{n+1})$ obviously includes the single interpolation point $s = x + kF(x, t_n)$ and its stencil, but the point s in turn depends on the φ values used to compute the extended velocity $F(x, t_n)$. Thus the CFL condition can be satisfied in principle by computing F nonlocally with arbitrarily large time steps. A specific nonlocal technique which satisfies the CFL condition is to postprocess the velocity field by smoothing or averaging it over a sufficiently large stencil. Accuracy can be maintained by increasing stencil size only logarithmically as $h \to 0$. In practice, a few passes of smoothing produces convergent solutions even though curvature flow velocities give parabolic level set equations, for which explicit schemes usually require $k = O(h^2)$.

Redistancing and velocity extension techniques also implement long-distance information transfer and help satisfy the CFL condition. While these techniques propagate information primarily normal to the interface, their influence is enhanced in regions of high curvature because normal vectors cross near the interface.

4. QUADTREE MESHES

Moving interfaces by solving the level set equation differs from solving general PDEs because we need to resolve the solution φ only near its zero set. Quadtree meshes coarsen rapidly away from $\Gamma(t)$ to resolve the interface with optimal efficiency and eliminate the cost of going up a dimension. In this section, we review standard properties of quadtree meshes. We define, build, and triangulate quadtree meshes in Subsection 4.1, then specialize in Subsection 4.2 to develop some useful properties of quadtree meshes built to resolve a given interface Γ.

4.1. Quadtree Meshes

4.1.1. *Definition*

A *quadtree mesh* covering the cube $[0, 1]^d$ in $\mathbf{R}^d$ is composed of square cells organized into levels, with each cell on level $l+1$ contained in some level-l cell. A quadtree mesh built to resolve a given function φ on $[0, 1]^d$ stores the following information:

- The root cell $C_0 = [0, 1]^d$, which occupies level $l = 0$.
- A maximum depth $L \geq 0$.
- A *cell list* of cells, grouped by level.
- A *vertex list* of cell vertices (corners), without repetitions.
- A *vertex value list* of φ values at cell vertices.
- Other application-dependent data.

Each cell C in the cell list contains:

- Its level l and corner vertex $(i_1, \ldots, i_d)$: the cell covers the box $2^{-l}[i_1, i_1 + 1] \times \cdots \times [i_d, i_d + 1]$.
- The indices in the vertex list of the 2^d cell vertices.
- The index in the cell list of its parent (if there is one).
- The indices in the cell list of its children (if there are any).
- Other application-dependent data.

An example is shown in Fig. 4 and Table I. Given an L-level quadtree, many operations related to searching and sorting can be done efficiently. Finding the tree cell where a point x lies, for example, requires $O(L)$ checks of bits in the binary representation of x.

4.1.2. *Building the Quadtree*

To build a quadtree, start with a root cell at level $l = 0$. Test whether it needs splitting into 2^d children on level $l + 1$. The *splitting criterion* distinguishes one quadtree from another and must be specified to suit the application. If a cell needs splitting, some bookkeeping must be done—creating new vertices, adjusting familial pointers, and so forth—and the values of φ at new vertices must be found. Then the children are tested, split if necessary, and the process repeats recursively. The build terminates when no cell above level L requires splitting.

TABLE I
Stored Information for the Quadtree of Fig. 4

Cell	Children	Parent	Vertices
C_0	C_1, C_2, C_3, C_4	—	V_0, V_1, V_2, V_3
C_1	C_5, C_6, C_7, C_8	C_0	V_0, V_4, V_5, V_8
C_2	—	C_0	V_4, V_1, V_8, V_6
C_3	—	C_0	V_5, V_8, V_2, V_7
C_4	—	C_0	V_8, V_6, V_7, V_3
C_5	—	C_1	V_0, V_9, V_{10}, V_{13}
C_6	—	C_1	V_9, V_4, V_{13}, V_{11}
C_7	—	C_1	V_{10}, V_{13}, V_5, V_8
C_8	—	C_1	$V_{13}, V_{11}, V_{12}, V_8$

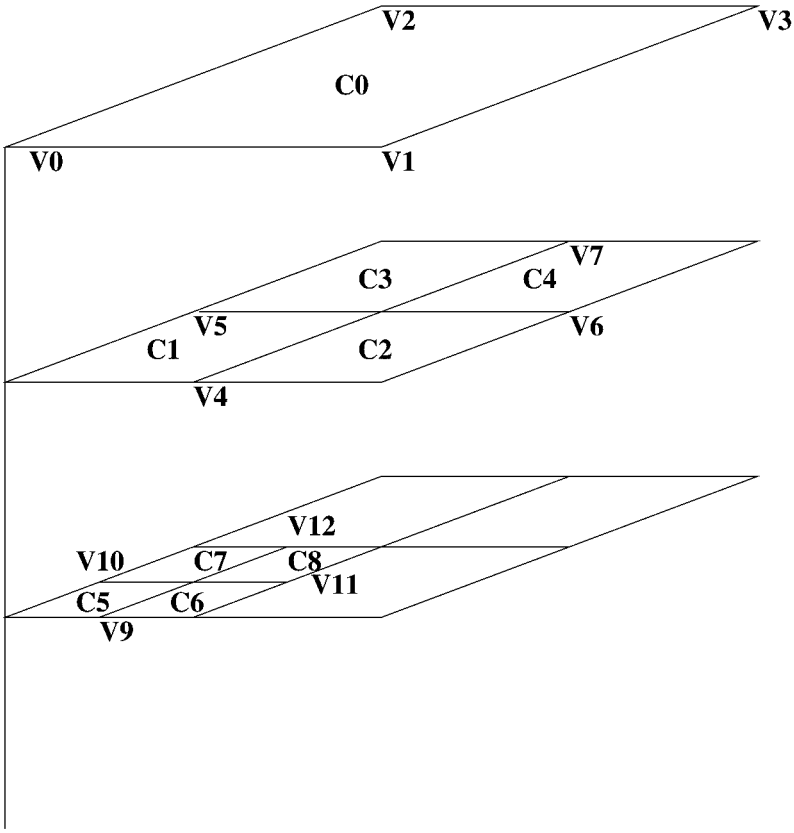

FIG. 4. Levels 0, 1, and 2 of a tree structure with cells C_i and vertices V_i.

4.2. Properties

This paper uses three different quadtrees, each built to resolve some interface Γ with some version of the following splitting criterion:

Split any cell whose edge length exceeds its minimum distance to Γ. (17)

Variants of this criterion determine the quadtree mesh at each time step, initialize the level set function φ and redistance φ at each step. This splitting criterion is one ingredient in the fast redistancing algorithm of [18], which we use in Section 5. The other ingredient is an efficient guaranteed-correct search strategy which uses a quadtree mesh to find nearest points on Γ. Infinite quadtrees built with Criterion (17) are known as Whitney decompositions and used to solve extension problems in harmonic analysis [16].

If φ is the signed distance to Γ, then the values of φ stored at cell vertices make this criterion extremely simple to implement. Figure 5 shows the cells in a quadtree for a simple interface. In general, Criterion (17) builds quadtrees with several useful properties:

- Adjacent cells differ in size by no more than a factor of 2, producing a smooth mesh and simplifying procedures such as neighbor finding and triangulation of the vertices.
- A cell's size is proportional to its distance to Γ.

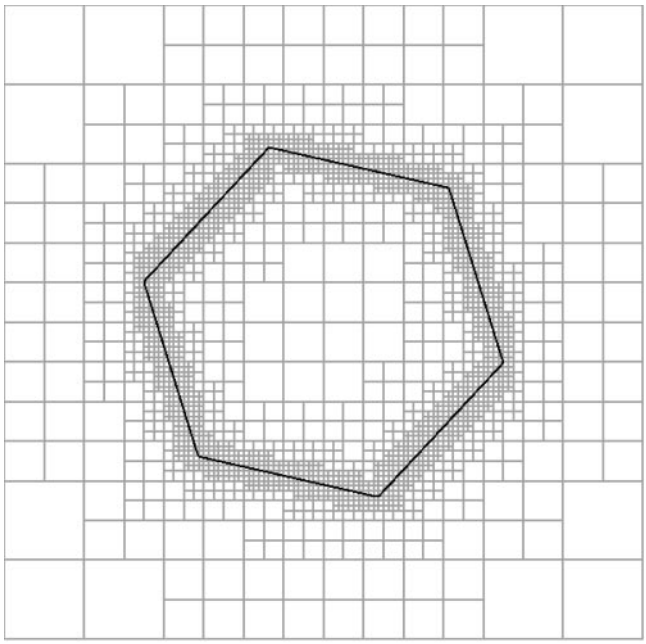

FIG. 5. The eight-level quadtree mesh built around the hexagonal zero set of Fig. 2.

• If φ is the signed distance to Γ at vertices and we extend φ into each cell by d-linear interpolation, then—because cells vary in size—φ will be discontinuous; see Fig. 6. However, the jumps in φ decrease in size in cells close to the interface because of the triangle inequality. Thus the interpolated φ is close to continuous near Γ.

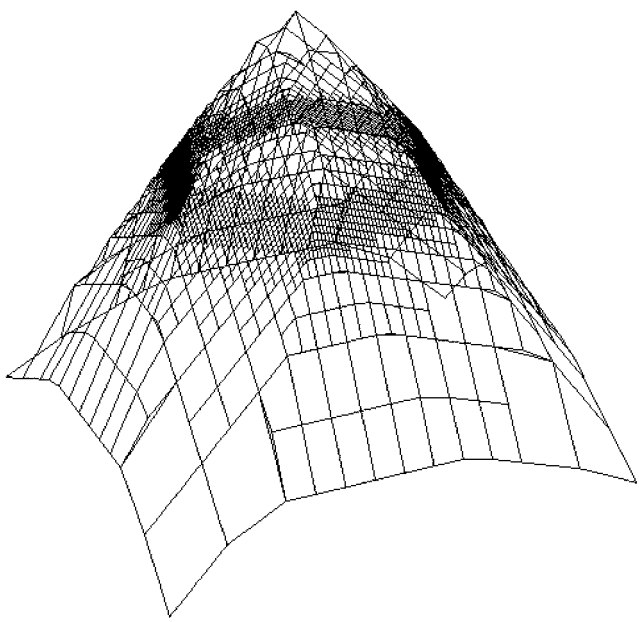

FIG. 6. The piecewise bilinear interpolant φ to the signed distance function on the eight-level quadtree mesh of Fig. 6.

• Cells coarsen very rapidly away from the interface: if there are N childless cells touching Γ, then the entire tree contains only $O(N)$ cells. Hence Γ is resolved accurately at minimal cost.

5. TREE METHODS FOR MOVING INTERFACES

We develop tree methods which move interfaces by combining the following ideas:

○ Topological changes require the solution of the level set equation only locally near the interface, not globally in space.

○ The interface can be accurately resolved at optimal cost by a quadtree mesh.

○ Semi-Lagrangian time stepping schemes such as CIR decouple time steps from CFL conditions, permitting time steps determined by resolution requirements rather than numerical stability.

○ Semi-Lagrangian schemes decouple mesh points into independent computations, permitting adaptive refinement without iteration.

○ With frequent redistancing, the solution φ of the level set equation is close to a signed distance function at all times, giving a natural splitting criterion for building a quadtree mesh and making error estimation unnecessary.

The combination of these ideas yields a family of adaptive methods. We summarize this family, identify the options which parametrize it, and discuss them in detail below.

In Subsection 5.1, we initialize the solution φ of the level set equation: given an initial interface $\Gamma = \Gamma(0)$, we build a quadtree Q_0 and an approximate signed distance function φ_0 on Q_0 which resolves Γ to specified accuracy ϵ in almost optimal time and space.

After initializing, we evolve the interface one step at a time. Optionally φ may be redistanced before the time step, as discussed in Subsection 5.2. Given a quadtree Q_n resolving the zero set Γ_n of $\varphi_n \approx \varphi(t_n)$, and an extended velocity F_n equal to the vector normal velocity VN on Γ_n, we build a quadtree Q_{n+1} to resolve the zero set Γ_{n+1} of the CIR approximation

$$\varphi_{n+1}(x) = \varphi_n(x + kF_n(x)). \tag{18}$$

Computing φ_{n+1} involves four procedures: extension, resolution, interpolation, and application of boundary conditions.

Extension. Extend the vector normal velocity VN off the interface to a global function $F_n(x)$ on the mesh Q_n. This extension problem can be solved in general or tailored to a specific moving interface problem. We discuss some specific techniques for passive transport and geometry in Subsection 5.3: local and global extensions, smoothing, truncation, interpolation, and differentiation on uniform and adaptive meshes. A general extension technique is developed in [20].

Resolution. Apply the splitting criterion of Subsection 5.4: form a quadtree Q_{n+1} resolving the zero set Γ_{n+1} of the CIR approximation φ_{n+1} from Eq. (18) to specified accuracy ϵ.

Interpolation. At off-mesh points $s = x + kF_n(x)$, our interpolation strategy determines stability as well as accuracy and is detailed in Subsection 5.5.

Boundary conditions. Numerical boundary conditions are straightforward and discussed in Subsection 5.6.

5.1. Initialization

A moving interface computation begins with the initial interface $\Gamma_0 = \Gamma(0)$, while the level set equation requires an initial level set function $\varphi_0 = \varphi(0)$ with zero set Γ_0. The signed distance function

$$D(x) = \pm \min_{y \in \Gamma} \|x - y\| \tag{19}$$

is prohibitively expensive to compute directly: If

$$\Gamma = \bigcup_{i=1}^{N} [\gamma_i, \gamma_{i+1}]$$

is a polygonal curve in $\mathbf{R}^2$, then evaluating

$$D(x) = \pm \min_{i=1}^{N} \min_{y \in [\gamma_i, \gamma_{i+1}]} \|x - y\|$$

costs $O(N)$ work per evaluation. We initialize φ_0 efficiently by building a quadtree Q_0 with Criterion (17), setting $\varphi_0 = D$ at the vertices of Q_0 and on cells touching Γ_0, and interpolating φ_0 linearly on cells not touching Γ_0.

As noted in Subsection 4.2, this splitting criterion produces a mesh which coarsens so rapidly away from Γ that if there are N cells touching Γ, then the entire mesh contains only $O(N)$ cells. Thus if Γ has N elements, then direct evaluation of all the quadtree vertex values costs only $O(N^2)$ work, much less than the $O(N^{d+1})$ for evaluating φ on a uniform mesh in d dimensions. Faster $O(N \log N)$ redistancing algorithms are discussed in Subsection 5.2.

5.2. Redistancing

Moving interfaces by solving the level set equation differs from solving a general PDE, because we can ignore all values of φ far from the zero set. In particular, we can replace the solution at any time by an approximate signed distance with the same zero set.

Frequent redistancing improves numerical accuracy. Figure 7 plots φ for a circle growing with unit normal velocity $V = 1$, computed by the method of [19]. The solution φ satisfies a maximum principle, so maxima can never increase. However, this also leads to flattening of the level set function: $\nabla\varphi$ may become small near the interface, causing level sets to broaden into regions or become difficult to contour. Redistancing cures flattening completely and reestablishes a clean intersection between the φ surface and any horizontal plane. Also, redistancing eliminates numerical effects due to artificial boundaries.

Redistancing is equivalent to initialization once Γ is found, and many contouring techniques which find Γ are available. The simplest technique splits each cell into two triangles, finds the exact zero segment of the linear interpolant to φ on each triangle, then joins the segments to form the interface. The choice of cell splitting direction makes this contouring technique anisotropic and helps indicate errors: underresolved computations can signal error by displaying a directional bias.

A fast algorithm which computes an approximate signed distance φ_0 at the quadtree vertices in $O(N \log N)$ work was developed in [18]. It uses an efficient search strategy to compute the minimum distance from all vertices of the quadtree to Γ and runs fast enough to redistance at every time step. Other fast redistancing algorithms apply the eikonal

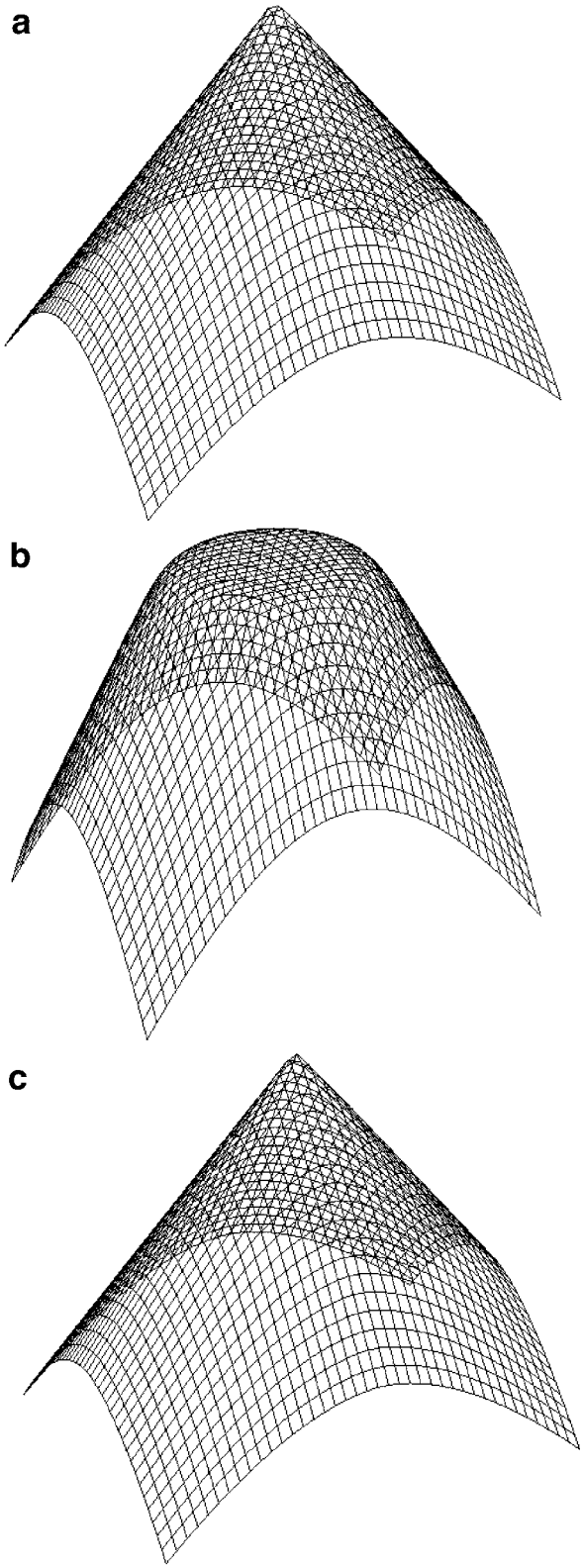

FIG. 7. Flattening of the level set function for a circle moving with constant normal velocity. Initial φ (a), final φ without redistancing (b), and final φ with occasional redistancing (c).

equation [21] and heapsort techniques [1], primarily on a uniform mesh. A polygonal interface made of N line segments has a "Voronoi diagram" which can be computed in $O(N \log N)$ time [26] and solves the redistancing problem exactly. However, the constant in $O(N \log N)$ is large and the algorithm is complex to program. A simpler structure called the compact Voronoi diagram may lead to faster redistancing algorithms [7], though at present no implementation is available.

5.3. Extension

Level set methods require a globally defined velocity F which extends VN smoothly off the interface $\Gamma(t)$. Many ad hoc velocity extensions for specific problems are described in [12], while general extension techniques are developed in [1, 20, 21].

Our tree methods also require velocity extension. The test problems solved in Section 6 have natural velocity extensions: for passive transport F is given, while geometric velocities such as

$$F = (R + \epsilon \cos(K\theta + \theta_0))N + (R' + \epsilon' \cos(K'\theta + \theta_0'))CN \tag{20}$$

can be evaluated by the natural geometric formulas $N = \nabla\varphi/\|\nabla\varphi\|$ and $C = -\nabla \cdot N$. For more general problems, we plan to incorporate the general extension of [20].

Naturally extended geometric velocities produce two numerical difficulties. First, the exact solution φ is not differentiable when facets or corners develop, $\nabla\varphi$ vanishes at extrema so N and C are not defined there, and redistancing on a quadtree introduces discontinuities as well. Our approximate signed distance function is discontinuous when cells change size, though the jumps decrease steadily in size as we approach Γ.

The second difficulty is the CFL condition, which requires small time steps $k = O(h^2)$ in almost all explicit schemes for parabolic level set equations such as curvature flow. The CIR scheme on a uniform mesh converges with a much more efficient time step $k = O(h)$ provided that the CFL condition is satisfied by smoothing the velocity and redistancing φ frequently [19]. Hence convergence for curvature-dependent velocities will require smoothing and frequent redistancing.

We have developed both cell-based and grid-based schemes for evaluating geometric velocities. Cell-based schemes are fast and work well for problems with first-order φ derivatives, while grid-based schemes are slower, more general, and work well for curvature-dependent velocities. We describe these approaches below.

5.3.1. *Cell-Based Velocity Evaluation*

The cell-based approach computes geometric velocities $F_n(x)$ locally at each new tree vertex x in Q_{n+1}. Suppose x lies in a cell C of the old quadtree Q_n. Then we can form the bilinear interpolant B to the vertex values of φ and approximate $\nabla\varphi$ by ∇B on C. Second derivatives can be computed by iterating the interpolation, or by using the biquadratic interpolant Q to the nine φ values at vertices of C and its siblings. Q raises the order of accuracy by one but doubles the cell size and introduces a stability issue: for linear constant-coefficient problems in one space dimension, CIR is unstable with quadratic interpolation.

We can vary this technique by computing the velocity at all vertices of the old quadtree Q_n and interpolating it to the new tree vertices. Smoothing techniques can then be applied because the velocity is computed on the whole quadtree rather than piecemeal and permit more effective solution of parabolic problems.

5.3.2. *Grid-Based Velocity Evaluation*

We can evaluate geometric velocities with an auxiliary grid by the following procedure: build a uniform $2^L \times 2^L$ grid matching the smallest cell in the quadtree. Interpolate φ to the uniform grid by the cell-based bilinear interpolation of Subsection 5.5.1, which is exact at vertices shared by the quadtree and the uniform grid. Apply the standard grid-based techniques of smoothing and differentiating φ, truncating and smoothing F on the grid, as in the geometric velocity evaluation of [19]. Finally, restrict F to the quadtree vertices, which form a subset of the uniform grid. This approach is powerful and general, but costly because of the uniform grid. However, the cost can be reduced by masking off unneeded areas.

5.4. Resolution

At each step, our methods build a quadtree mesh to resolve the CIR approximation

$$\varphi_{n+1}(x) = \varphi_n(x + kF_n(x)) \tag{21}$$

to the level set function $\varphi(x, t_{n+1})$. The quadtree is built recursively from the root cell C_0 by the following splitting criterion:

$$\text{Split every cell where } |\varphi_{n+1}| \text{ is larger than the edge length.} \tag{22}$$

Thus we apply the splitting criterion (17) as if φ_{n+1} were a distance function. Redistancing at every step keeps

$$\varphi_{n+1}(x) = \varphi_n + kF \cdot \nabla\varphi_n + o(k) = \varphi_n + O(k) \tag{23}$$

within $O(k)$ of the signed distance function φ_n. Thus in the limit $k = O(h) \to 0$, Criterion (22) reduces to (17), yielding the properties noted in Subsection 4.2.

5.5. Interpolation

The CIR scheme requires interpolated φ values at the projected points $s = x + kF_n(x)$. Many general interpolation techniques are available, but our choice is restricted by the irregularity of the quadtree Q_n and by two requirements. First, the level set function φ is only Lipschitz continuous in general since faceting may occur. Thus high-order methods which require smooth data should be avoided. Second, stability of the semi-Lagrangian approach in any given norm is guaranteed only for interpolation schemes which do not increase the norm too much. For example, linear interpolation was used in [19] to guarantee unconditional max-norm stability. Similarly, shape-preserving interpolation [10] was used in [24] and monotone advection in [14].

Given these two requirements and a quadtree mesh, two obvious classes of interpolation techniques are available: cell-based and triangulation-based. Both become locally exact by setting φ equal to D near $\Gamma(t)$.

5.5.1. *Cell Interpolation*

Here we use the square cells of the quadtree to interpolate from vertex values of φ. Bilinear interpolation to a point $(x, y) = (x_0 + \alpha h, y_0 + \beta h)$ in a cell C evaluates

$$(1-\alpha)(1-\beta)\varphi_{00} + \alpha(1-\beta)\varphi_{10} + (1-\alpha)\beta\varphi_{01} + \alpha\beta\varphi_{11}, \tag{24}$$

where the vertex values for C are given by $\varphi_{ij} = \varphi(x_0 + ih, y_0 + jh)$. Bilinear interpolation preserves the maximum principle of the CIR scheme and yields local second-order accuracy, with global first-order error $O(k + h)$ after $O(1/k)$ time steps.

Biquadratic cell interpolation requires nine φ values while a cell C has only four vertices. Hence given x in a childless cell, we can ascend one level to interpolate from the nine vertices of C and its siblings. This gains an order of accuracy but doubles the mesh size and sacrifices the maximum principle.

5.5.2. *Triangle Interpolation*

We can also interpolate by triangulating the vertices of the quadtree and building the continuous piecewise-linear interpolant to φ at the vertices. As in [3], we can add one Steiner vertex at the center of each cell and connect the vertices to form a high-quality triangulation in only $O(N)$ work. The center values of φ may be evaluated exactly or interpolated from vertices.

5.5.3. *Exact Interpolation*

A third alternative uses the quadtree to evaluate the signed distance to Γ_n exactly, eliminates interpolation entirely, and is discussed in [20].

5.6. Boundary Conditions

The CIR scheme requires numerical boundary conditions to specify values for $\varphi(s, t_n)$ when s lies outside the domain D covered by the grid.

There are two simple boundary conditions: extension and projection. In extension, we extend φ as a constant or linear function along lines normal to the boundary ∂D then apply our standard interpolation scheme to interpolate the extended values to s. In projection, we arrest s as it leaves the domain and use one-sided interpolation to the point where s crosses ∂D. Our tree methods use projection because it is simple and effective.

6. NUMERICAL RESULTS

We validate our tree methods by computing a variety of interfaces moving under passive transport and geometric motions, with corners, anisotropy, nontrivial topology, and curvature. (Some PDE-type examples with a general velocity extension will be treated in future work [20].) Our methods were implemented in Standard C, compiled with the SunSoft C compiler and the `-fast` optimization flag, and run on one CPU of a 2-CPU 200 MHz Sun Ultra-2 under Solaris 2.6.

6.1. Passive Transport

Passive transport problems where $\Gamma(t)$ moves with a globally defined velocity $F(x, t)$ constitute convenient test cases for moving interface methods, because complex exact solutions can easily be evaluated. Thus we can measure the error and rate of convergence. We carry out convergence studies for three passive transport problems and verify the accuracy, robustness, and conservation properties of tree methods.

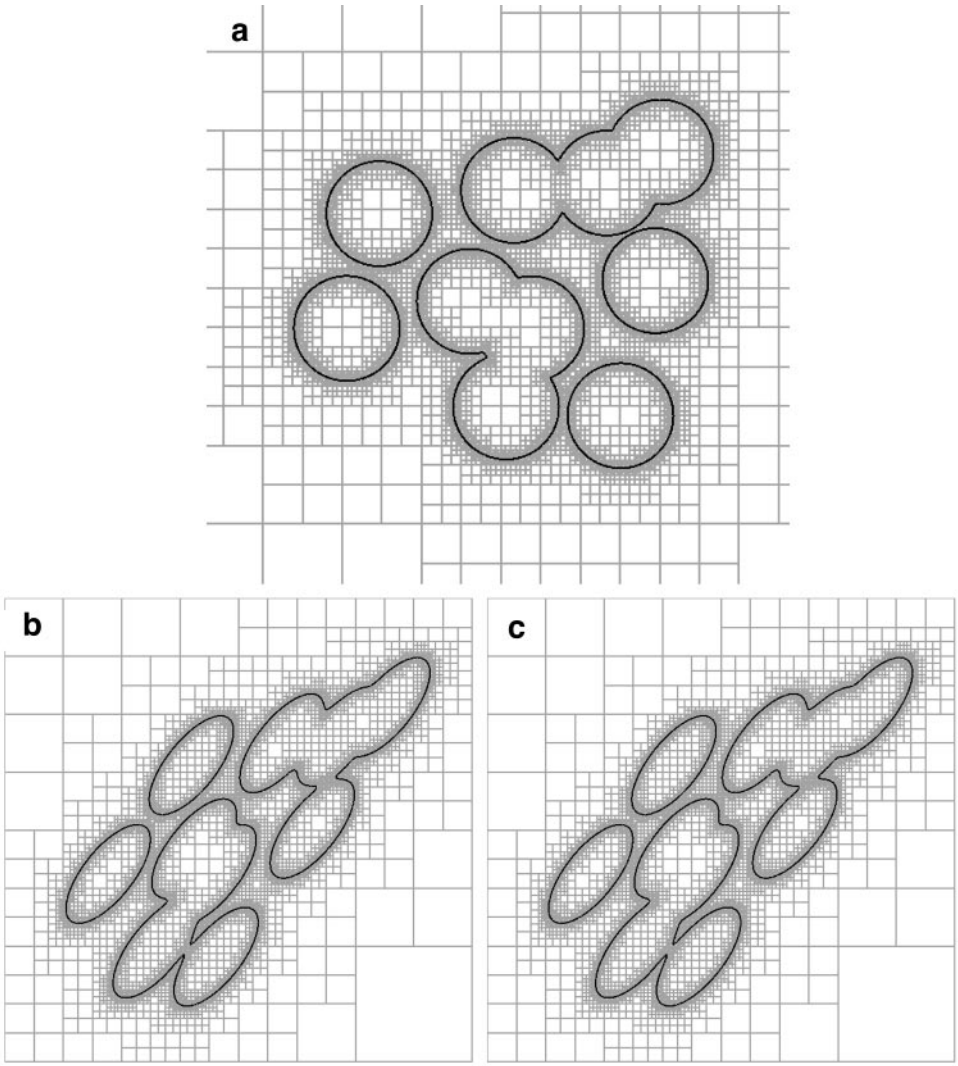

FIG. 8. A collection of bubbles moving with linear shearing velocity.

6.1.1. *Bubbles in a Shear Flow*

We measure the accuracy of our methods on a collection of circular bubbles (Fig. 8) moving with a divergence-free linear shearing velocity

$$F(x, y) = \frac{1}{2}\left(x - 3y + 1, -y - \frac{1}{2}\right). \tag{25}$$

We use $10, 20, \ldots, 320$ time steps on $0 \le t \le 1$ on a quadtree with 5 through 9 levels on $[-5, 5] \times [-5, 5]$. Table II reports the maximum of the exact distance function on the computed contour at time $t = 1$. First-order accuracy is clearly evident along diagonals, where $h = O(k)$. This agrees with the consistency condition of Subsection 3.2. The error decreases considerably when we change from bilinear to biquadratic cell interpolation, indicating that the error is largely due to spatial discretization.

TABLE II

Grid levels	$N_T = 10$	20	40	80	160	320
			Linear interpolation			
5	0.276	0.139	0.631	0.96	0.36	0.567
6	0.0413	0.169	0.253	0.135	0.176	0.457
7	0.031	0.027	0.0986	0.194	0.269	0.209
8	0.0449	0.0148	0.0143	0.054	0.112	0.222
9	0.0479	0.0221	0.00686	0.0102	0.0384	0.0831
			Quadratic interpolation			
5	0.0423	0.12	0.283	0.345	0.338	0.341
6	0.0398	0.0267	0.0351	0.0518	0.0378	0.0352
7	0.0468	0.0227	0.00982	0.0128	0.0208	0.0239
8	0.0486	0.0236	0.0116	0.00565	0.00145	0.00396
9	0.0489	0.0241	0.0118	0.00593	0.00298	0.00118

Note. Maximum error at $t = 1$ in the interface shown in Fig. 8, moving with divergence-free linear shearing velocity $F(x, y) = \frac{1}{2}(x - 3y + 1, -y - \frac{1}{2})$, computed with N_T time steps of linear and quadratic interpolation. The domain is $[-6, 6]^2$.

6.1.2. *Grid Effects on Triangles*

A common difficulty in moving interfaces is sensitive dependence on numerical artifacts such as grid orientation. We check for grid effects in passive transport of a sharply faceted interface by revolving, shrinking, and expanding a triangle with a linear velocity field. In all cases, each facet moves with the appropriate speed independently of its orientation relative to the grid. Figure 9 plots the results with both bilinear and biquadratic cell interpolation on the domain $[-2, 2]^2$ and shows that grid effects are minimal. Each plot demonstrates convergence by superimposing three runs with 40, 80, and 160 time steps on a quadtree with 5, 6, and 7 levels.

6.1.3. *Mass Conservation in a Shear Flow*

We conclude our study of passive transport by measuring mass conservation in a collection of bubbles moving in the divergence-free shearing flow given by

$$F(x, y) = \frac{\max(1 - (1 - x^2 - y^2)_+^4, 0)}{8(x^2 + y^2)}(-y, x). \tag{26}$$

Figure 10 shows the extreme distortion produced by this flow, computed with 160 time steps on $0 \le t \le 100$ and bilinear interpolation on a 9-level quadtree on the domain $[-6, 6]^2$. This mesh resolves $\Gamma(t)$ as accurately as a 512×512 uniform mesh, at far less cost. Despite this distortion, mass is well conserved; the final area inside the computed interface is 12.7701, close to the exact value of $4\pi = 12.5664$.

6.2. Geometry

We validate our methods by computing converged solutions to a variety of geometric moving interface problems including viscosity solutions to corners moving with unit normal velocity, the faceted Wulff limit for anisotropic normal velocity fields, complex topological changes under anisotropic curvature-dependent flows, and nonconvex shapes shrinking to

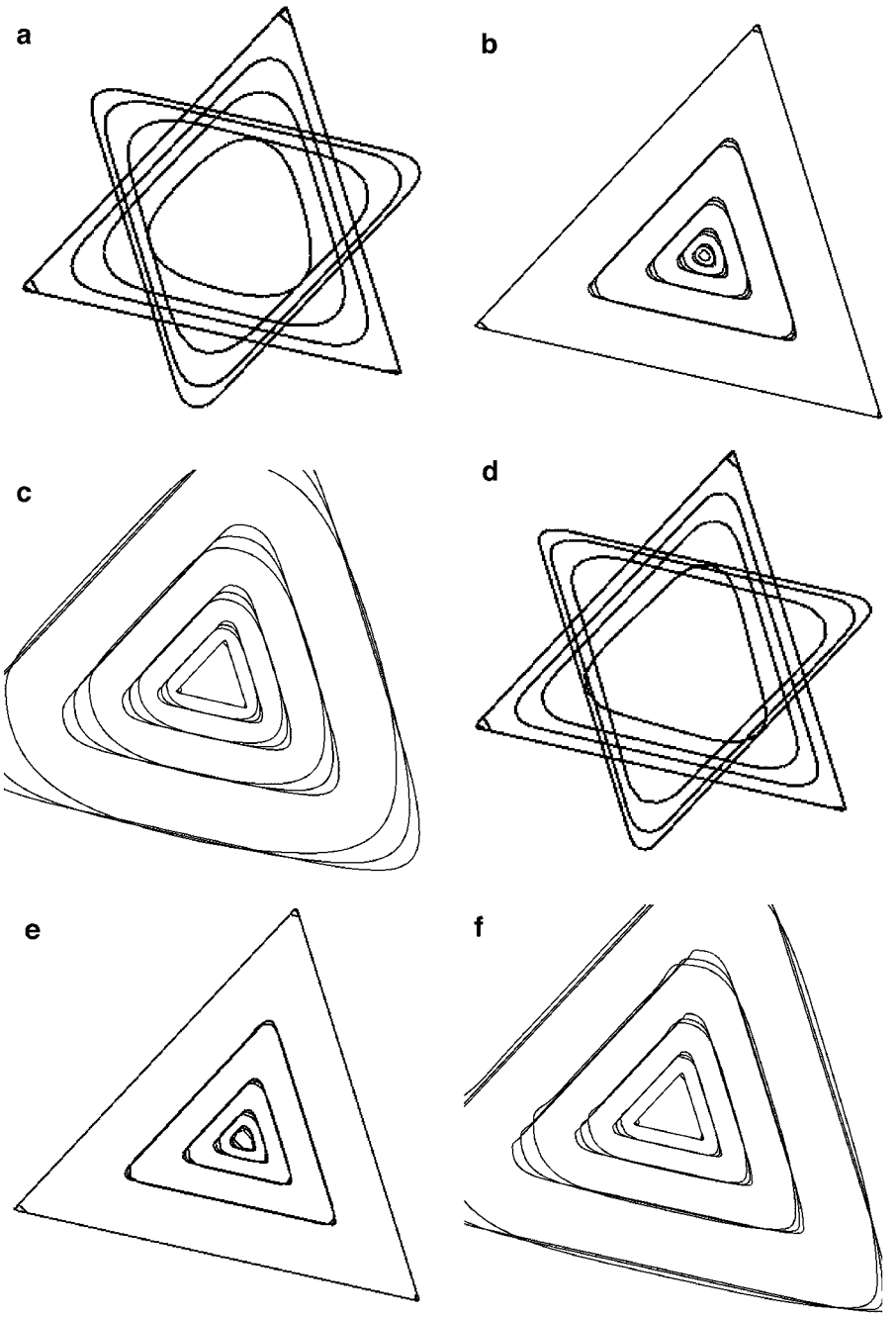

FIG. 9. Tests of grid effects in sharp corners with linear velocity field. (a) A rotating triangle at a half period and a full period, computed with bilinear cell interpolation. (b) A triangle shrinking with $V(x, y) = -\frac{5}{2}(x, y)$ from $t = 0$ to $t = 1$. (c) A triangle expanding with $V(x, y) = 2(x, y)$ from $t = 0$ to $t = 1$. Plots (d) through (f) show the same calculation with biquadratic cell interpolation.

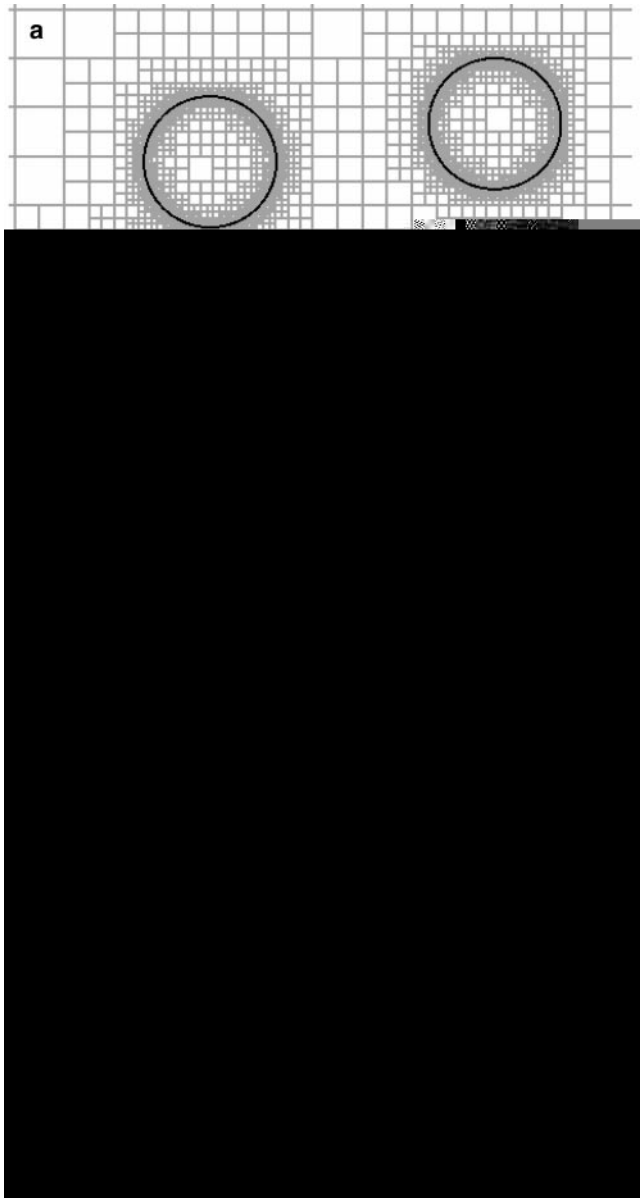

FIG. 10. A collection of circular bubbles passively transported by a divergence-free shearing velocity.

round points under flow by curvature. These are among the most important tests of general moving interface methods.

6.2.1. *Unit Normal Velocity*

We verify first-order accuracy on a unit circle centered at $(1/2\pi, 1/2\pi)$ with unit normal velocity, extended naturally via Eq. (6) with singularities truncated;

$$F = N = \frac{\nabla\varphi}{\max(10^{-8}, \|\nabla\varphi\|)}. \tag{27}$$

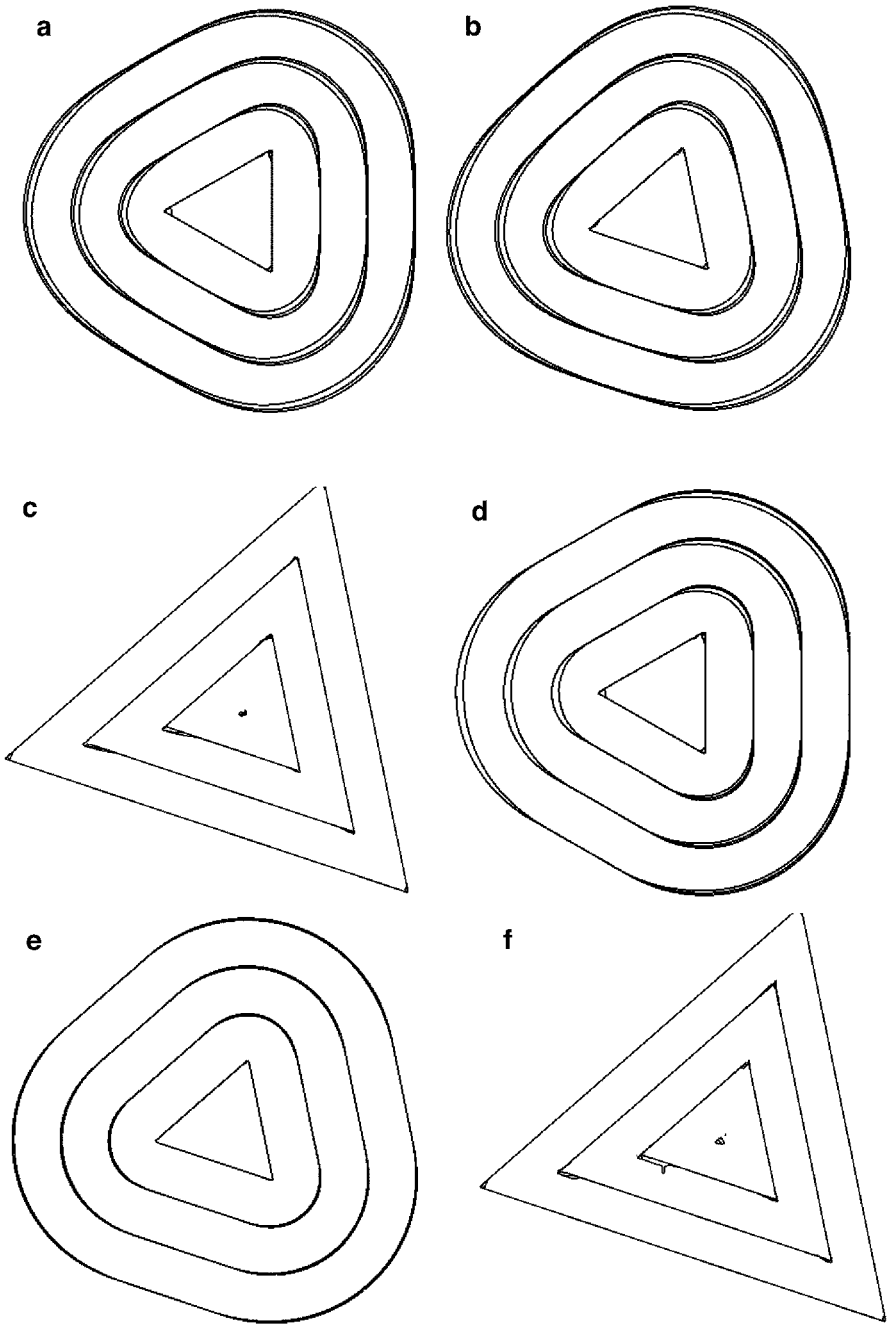

FIG. 11. Viscosity solutions for triangles moving with positive or negative unit normal velocity, computed with bilinear cell interpolation: (a) An expanding triangle at zero angle to the mesh, with round corners. (b) An expanding triangle at angle 0.2 radians to the mesh, with round corners. (c) A shrinking triangle at angle 0.2 radians to the mesh, with sharp corners. Plots (d) through (f) show the same computations with biquadratic cell interpolation. Each plot demonstrates convergence by superimposing three runs with 40, 80, and 160 time steps on tree meshes with 6, 7, and 8 levels.

Table III reports the maximum of the exact distance function on the computed contour at time $t = 1$, with $10, 20, \ldots, 160$ time steps on $0 \le t \le 1$ and quadtrees with 5 through 9 levels on $[-3, 3]^2$. Bilinear and biquadratic interpolation are used for φ interpolation and the cell-based evaluation of N. High accuracy is evident along diagonals, where $h = O(k)$, because the exact interface is a linear function of t.

6.2.2. *Viscosity Solutions with Corners*

Correct computation of "viscosity solutions" for faceted interfaces in geometric problems depends on moving a corner in or out with unit normal velocity [12]. Inward motion should keep corners sharp (the "shock" case), while outward motion should produce rounded corners due to Huygens' principle (the "rarefaction" case). Figure 11 shows a triangle moving with positive and negative unit normal velocity, both aligned with the mesh and at an angle to check for grid effects, and demonstrates that tree methods compute the correct viscosity solution in each case.

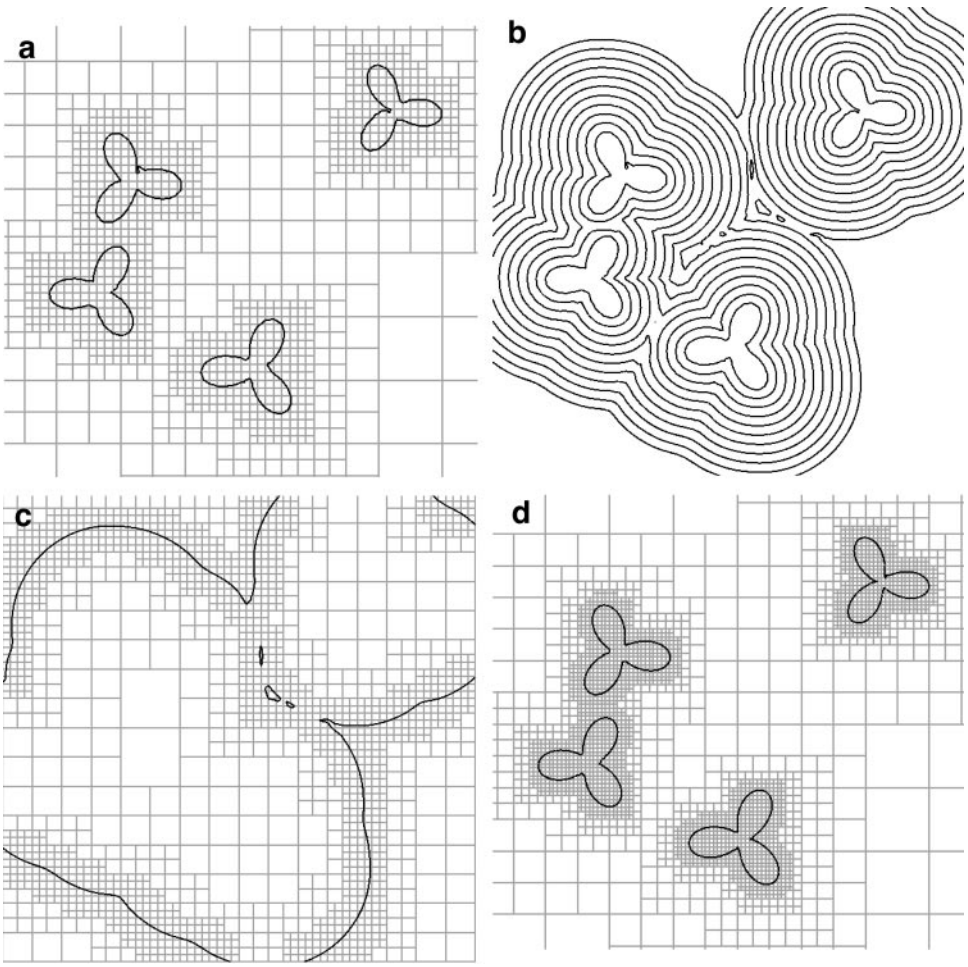

FIG. 12. A collection of randomly located, sized, and oriented trefoils growing and merging under unit normal velocity $V = 1$. Here (a) is the initial interface on a 6-level tree mesh, (b) plots every 8th step of 80 time steps, and (c) shows the final 6-level mesh. Plots (d)–(f) show 7 levels and 160 steps, while (g)–(i) show an accurately converged result with 8 levels and 320 steps.

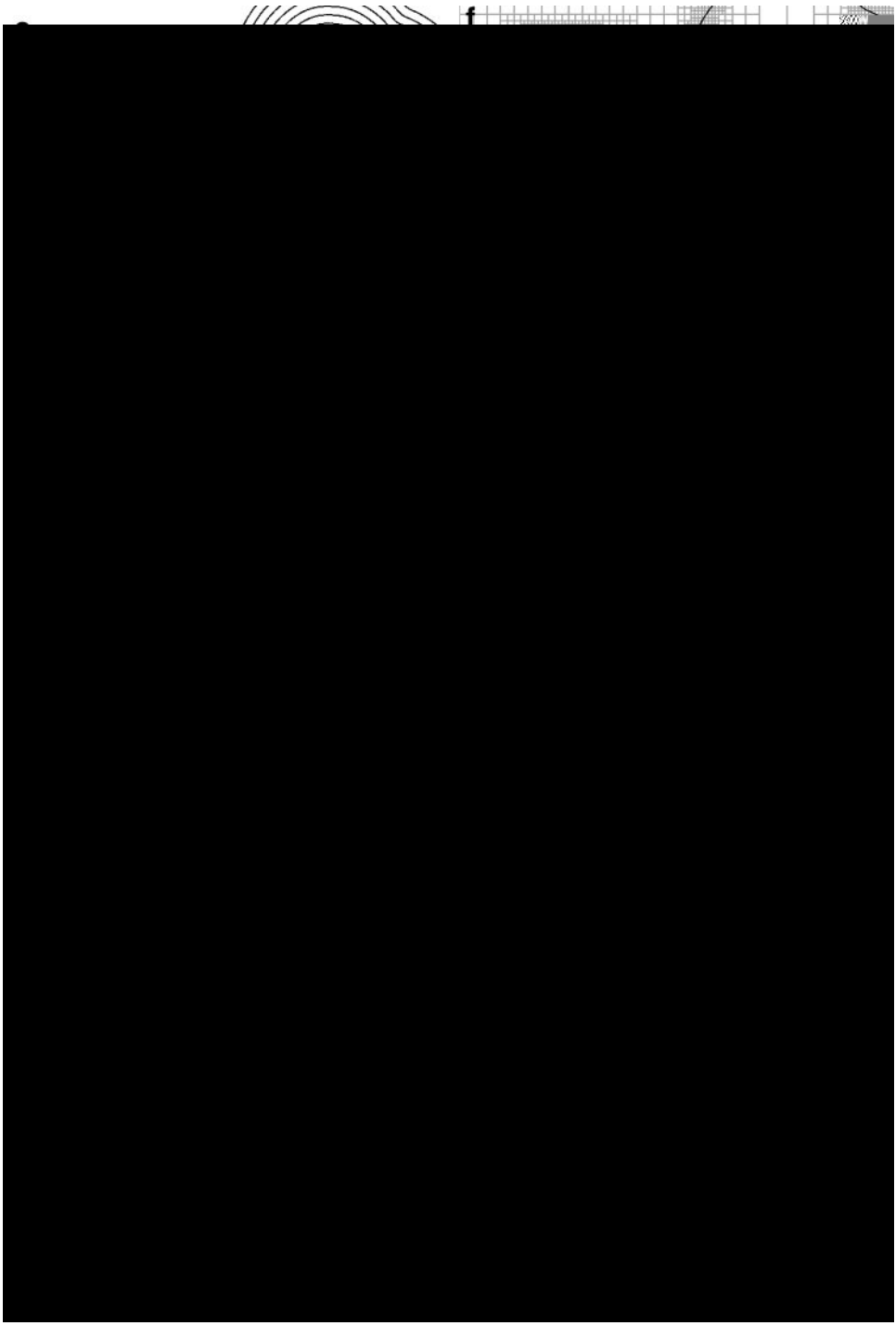

FIG. 12—*Continued*

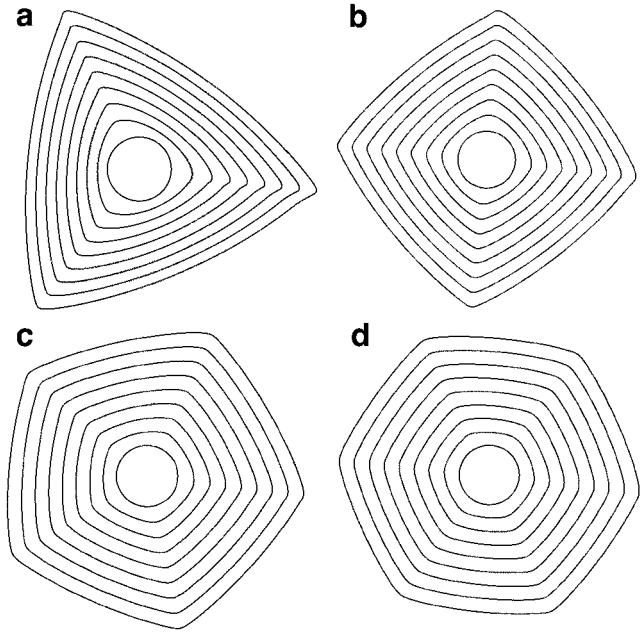

FIG. 13. Wulff shapes growing from circular initial interfaces (with radius $1/2$ and center at $(1/2\pi, 1/2\pi)$). Here we used 160 time steps on $0 \le t \le 1$ and cell-based bilinear interpolation on an 8-level tree mesh covering $[-3, 3]^2$.

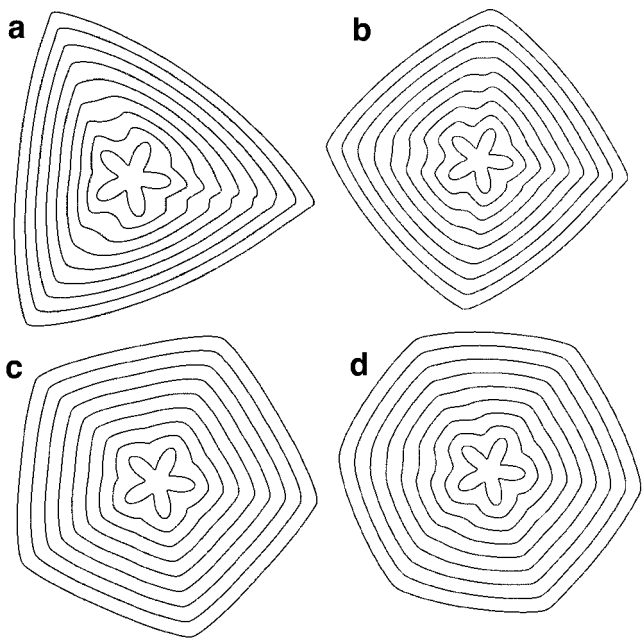

FIG. 14. Wulff shapes developing from nonconvex initial interfaces given by $r = 0.4 + 0.2\cos(5\xi)$ in polar coordinates (r, ξ) centered at $(1/2\pi, 1/2\pi)$. Here we used 160 time steps on $0 \le t \le 1$ and cell-based bilinear interpolation on an 8-level tree mesh covering $[-3, 3]^2$.

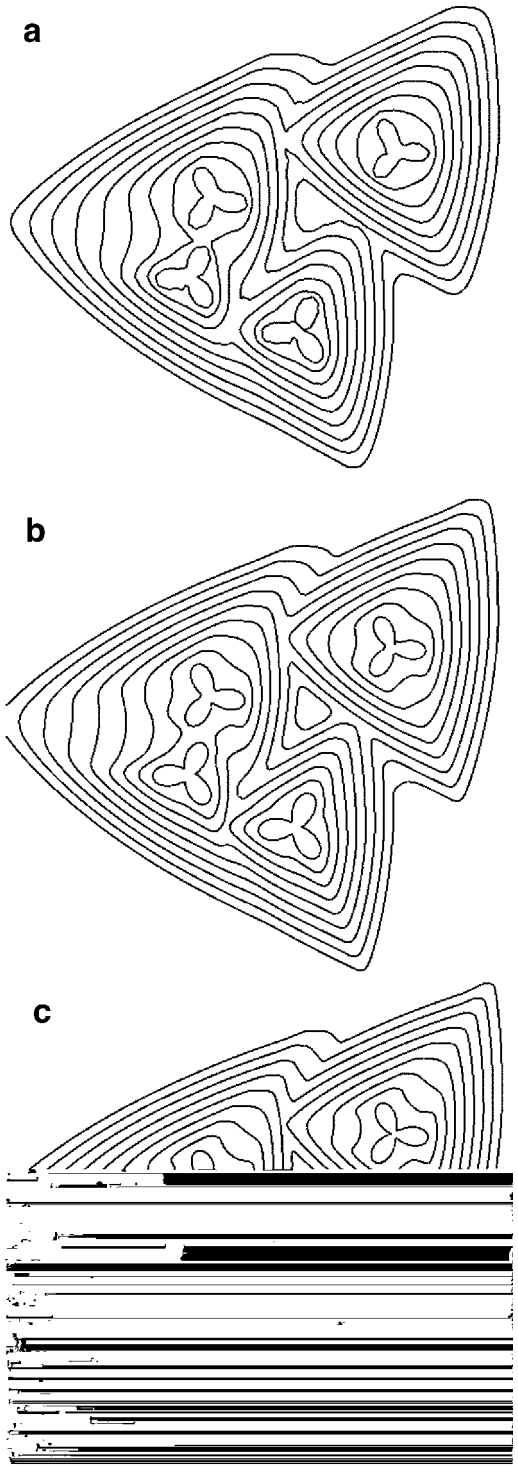

FIG. 15. A collection of randomly located, sized, and oriented trefoils growing and merging under a nonconvex anisotropic normal velocity $V = 2 + \cos(3\theta + 0.3)$. We used biquadratic cell interpolation with (a) 80 time steps on a 6-level tree mesh, (b) 160 steps on a 7-level mesh, and (c) 320 steps on a 8-level mesh, to achieve convergence to graphical accuracy.

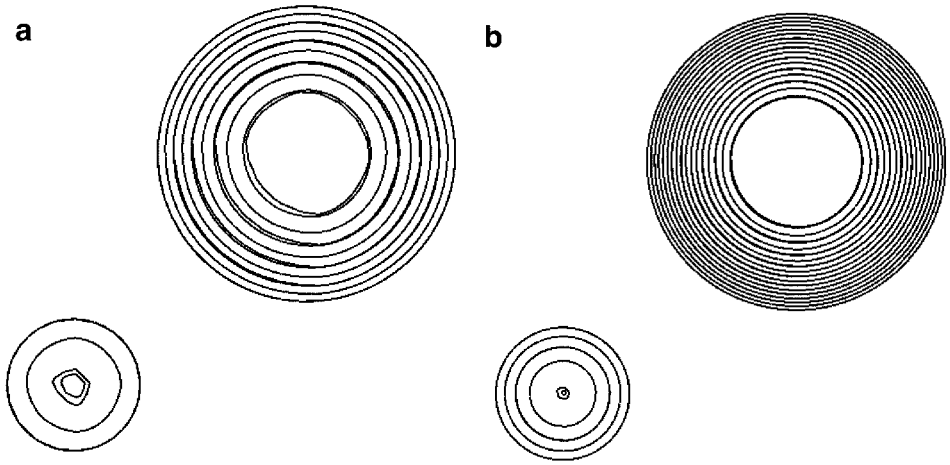

FIG. 16. Convergence of two circles collapsing under curvature flow $V = C$, computed from $t = 0$ to $t = 2$ with (a) 20 time steps on 5-level tree mesh covering $[-4, 5]^2$ with 1 smoothing pass per step, superimposed on 40 steps on 6-level mesh with 2 passes, (b) 80 steps on 7-level mesh with 3 passes, superimposed on 160 steps on 8-level mesh with 4 passes.

Figure 12 shows a complex interface growing and merging with unit normal velocity and exhibits the simplicity of the level set approach to topological complexity. The manifold corners and changes of topology are computed automatically and easily. In particular, outward-moving inward-pointing corners remain correctly sharp, as the viscosity solution theory requires. The final area enclosed by the computed interface is 72.77, 73.15, and 73.29 on the three runs shown, indicating smooth monotone convergence. The initial and final quadtrees are shown to demonstrate the extreme concentration of computational effort near the moving interface. An 8-level mesh resolves the interface as accurately as a 256×256 uniform mesh at far less cost.

TABLE III

Grid levels	$N_T = 10$	20	40	80	160
			Linear interpolation		
5	0.0307	0.0436	0.0505	0.054	0.0557
6	0.00647	0.0153	0.0215	0.0249	0.029
7	0.00135	0.00333	0.00981	0.0133	0.0155
8	0.000506	0.000707	0.00201	0.00675	0.00938
9	0.000123	0.00026	0.000447	0.00155	0.00505
			Quadratic interpolation		
5	0.00176	0.00199	0.00223	0.00234	0.00242
6	0.000377	0.000626	0.000754	0.000819	0.000858
7	0.0000754	0.000128	0.000198	0.000239	0.000263
8	0.0000128	0.0000701	0.0000206	0.0000386	0.000051
9	0.00000401	0.00000402	0.00000256	0.00000562	0.00000978

Note. Maximum of exact distance function at $t = 1$ on a circle of radius $R(t) = 1 + t$ and center $(1/2\pi, 1/2\pi)$, moving with constant normal velocity $V = 1$, computed with N_T time steps of linear and quadratic interpolation.

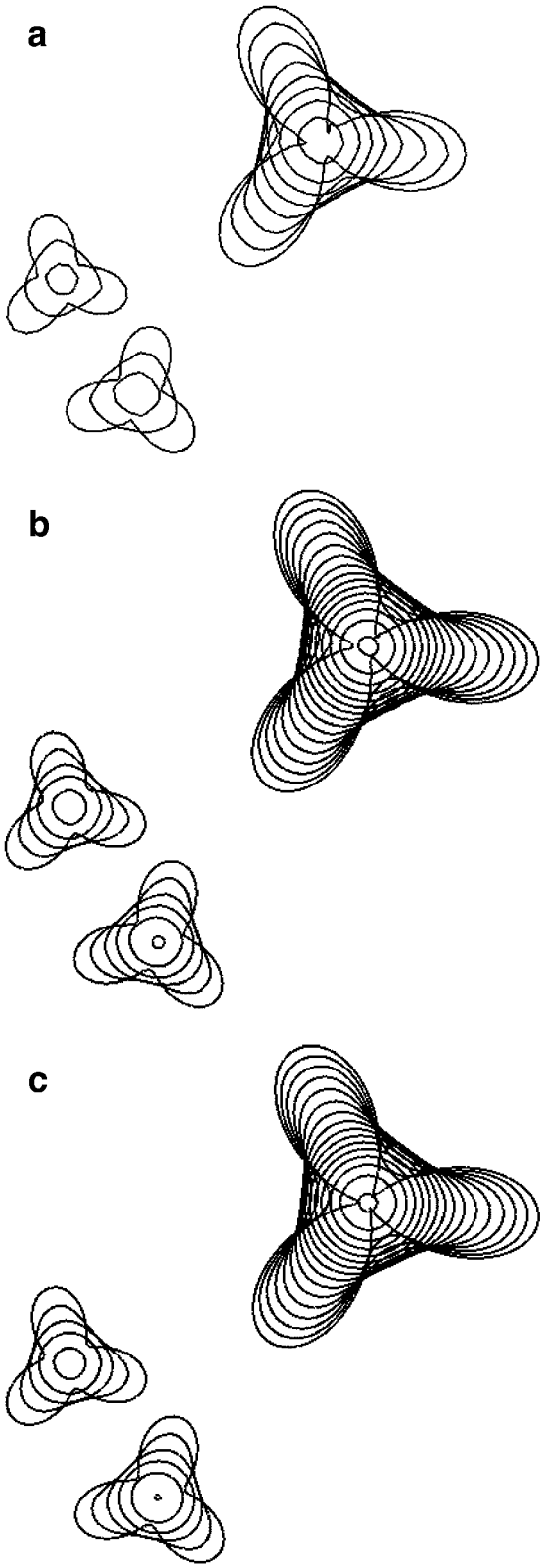

FIG. 17. Convergence of a collection of trefoils to round points under curvature flow $V = C$, computed from $t = 0$ to $t = 1$ with grid-based velocity evaluation using (a) 40 time steps on a 6-level tree mesh covering $[-4, 4]^2$ with one smoothing pass per step, (b) 80 steps on a 7-level mesh with two passes, (c) 160 steps on a 8-level mesh with three passes.

6.2.3. *Anisotropic Normal Velocity and the Wulff Limit*

Anisotropic motion along the normal connects moving interfaces to the theory of Hamilton–Jacobi equations

$$\varphi_t + H(\nabla\varphi) = 0 \tag{28}$$

which encounters difficulties when the Hamiltonian H is nonconvex. For anisotropic normal velocities

$$V = R + \epsilon\cos(k\theta), \qquad \cos\theta = \varphi_x/\|\nabla\varphi\|, \tag{29}$$

the Hamiltonian is nonconvex if

$$R + \epsilon(1-k^2) < 0 < R - |\epsilon|, \tag{30}$$

causing some Hamilton–Jacobi methods to break down.

In Fig. 13, we evolve an initially circular interface under several anisotropic normal velocities producing nonconvex Hamiltonians, with constants chosen to keep $R + \epsilon(1-k^2) = -4$.

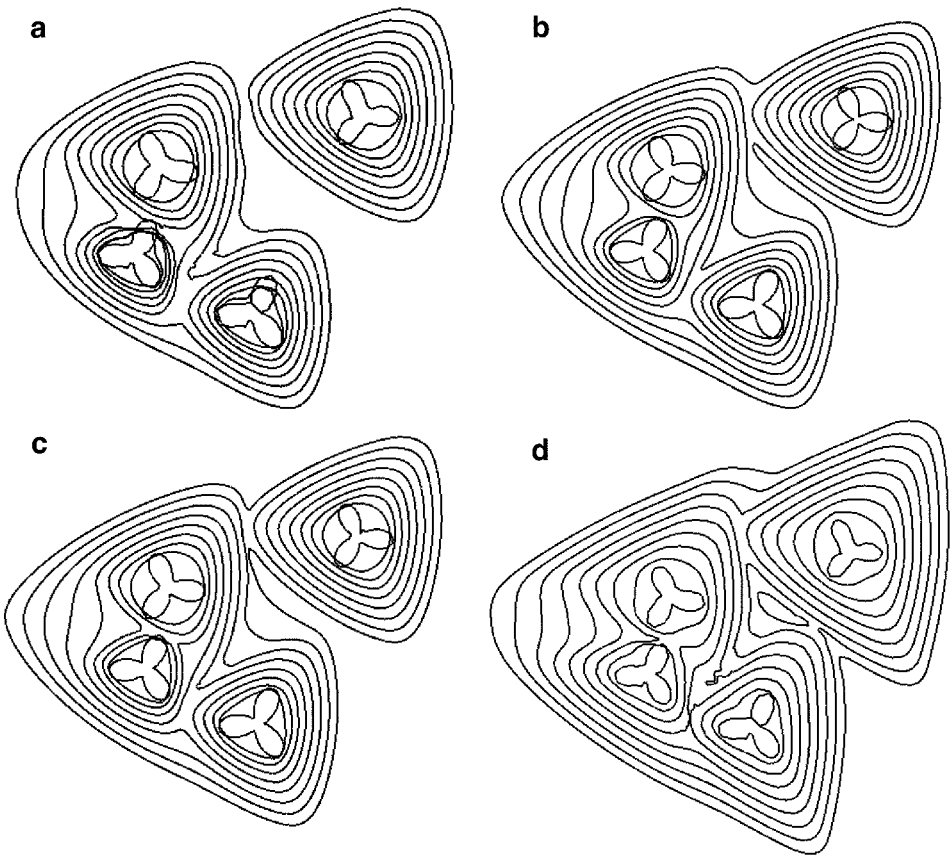

FIG. 18. Nonconvex shapes merging under curvature-dependent anisotropic flow $V = 2 + \cos(3\theta + 0.3) + \epsilon C$. Convergence to the viscosity solution as $\epsilon \to 0$ is demonstrated with $\epsilon = 1$ (a–c), 0.1 (d–f), and 0.01 (g–i). Figure 15 shows the limit case $\epsilon = 0$.

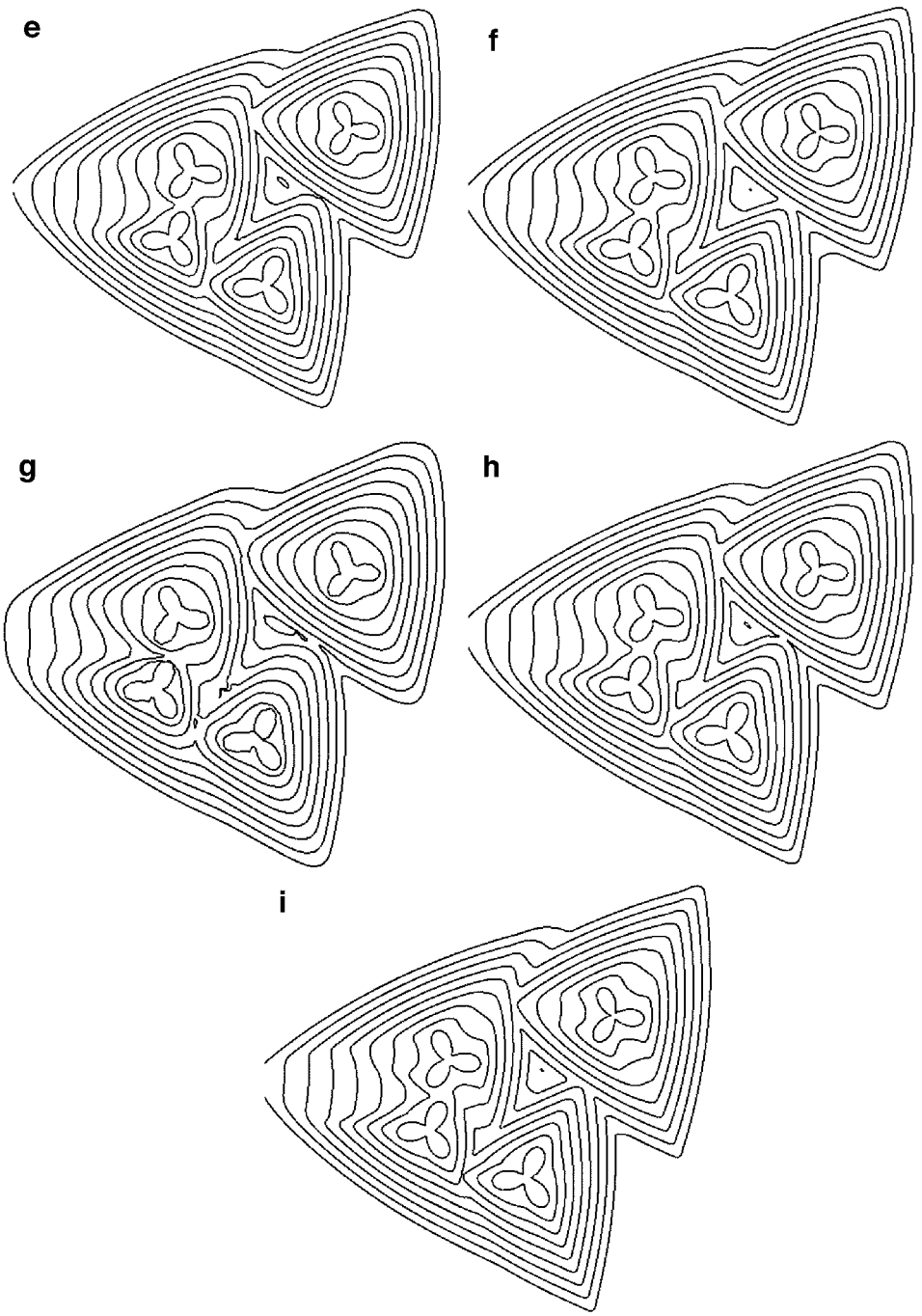

FIG. 18—*Continued*

The interface converges rapidly to the correctly tilted "Wulff shape" [25] corresponding to each given anisotropy, as predicted by rigorous theory [8]. In Fig. 14, we begin with a highly nonconvex initial interface to test our methods even more severely. The asymptotic Wulff shape is still computed accurately.

6.2.4. *Merging under Anisotropy*

Starting from a collection of randomly placed, sized, and oriented trefoil shapes, we move the interface along its normal with a threefold anisotropic speed $V = 2 + \cos(3\theta + 0.3)$, where θ is the angle between the normal vector and the positive x-axis. Figure 15 shows the mechanism which transforms this highly nonconvex initial interface into the asymptotic triangular Wulff shape as $t \to \infty$.

6.2.5. *Circles Shrinking under Curvature*

A classic geometric problem shrinks a plane curve with velocity equal to its curvature and forms a useful test case for curvature-dependent velocity. A circle shrinking with $V = C$ has exact radius $R(t) = \sqrt{R(0)^2 - 2t}$, so with $R(0) = \sqrt{5}$, a circle should shrink to radius 1 at time $t = 2$. A smaller circle with $R(0) = 1$ vanishes completely in time $t = 1/2$. Figure 16 shows convergence to graphical accuracy, computed with 20, 40, 80, 160 time steps on quadtrees with 5 through 8 levels and plotted every 0.2 or 0.1 time units. The final computed area of the large circle is 2.518, 2.849, 3.007, and 3.088, showing a smooth first-order convergence to the exact area π.

For this parabolic problem, we use grid-based velocity evaluation with redistancing every step to satisfy the CFL condition and obtain convergence with large time steps $k = O(h)$. We apply $L - 4$ passes of cosine smoothing on the L-level mesh computation.

6.2.6. *Nonconvex Interfaces under Curvature*

A geometric theorem [4] predicts that any smooth embedded plane curve should collapse to a round point and vanish in finite time under curvature flow $V = C$. We verify that tree methods behave correctly for a collection of randomly placed, sized and oriented nonconvex trefoil shapes, with the converged calculation shown in Fig. 17.

6.2.7. *Merging under Anisotropy Plus Curvature*

Finally, we validate our methods by computing the viscosity limit for a complex interface evolving through merging, fill-in, and faceting. Beginning as in Fig. 15, we move $\Gamma(t)$ with a curvature-smoothed velocity

$$V = 2 + \cos(3\theta + 0.3) + \epsilon C. \tag{31}$$

We illustrate the viscosity limit $\epsilon \to 0$ computationally with $\epsilon = 1$, 0.1, and 0.01. For each value of ϵ, we carry out a numerical convergence study with grid-based velocity evaluation, redistancing and smoothing at each step. Figure 18 shows rapid convergence to the results computed in Fig. 15.

7. CONCLUSION

We have described and validated new adaptive numerical methods for moving interfaces, which combine the level set equation, the semi-Lagrangian CIR time stepping scheme, and quadtree meshes. Our tree methods resolve and move complex interfaces at optimal cost with time steps unconstrained by numerical stability. They form key components of "black-box" methods for moving interfaces, which accept the interface and its velocity at time t

and return the evolved interface one time step later. Such methods simplify the solution of moving interface problems, because the moving interface numerics are independent of the physical problem driving the interfacial motion.

Numerical results show that tree methods converge to correct viscosity solutions even for difficult moving interface problems involving merging, faceting, transport, and anisotropic curvature-dependent geometry. Large time steps can be taken even for parabolic problems, with the aid of frequent redistancing and velocity smoothing.

Planned future developments include

- further investigation of CFL conditions for parabolic problems,
- higher-order accurate time stepping,
- completely modular moving interface methods [20], and
- applications to industrial crystal growth problems, where the moving interface is coupled to complex materials science.

REFERENCES

1. D. A. Adalsteinsson and J. A. Sethian, *The Fast Construction of Extension Velocities in Level Set Methods*, Technical Report PAM-738, UC Berkeley Center for Pure and Applied Mathematics, 1997.
2. R. Courant, E. Isaacson, and M. Rees, On the solution of nonlinear hyperbolic differential equations by finite differences, *Comm. Pure Appl. Math.* **5**, 243 (1952).
3. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications* (Springer-Verlag, Berlin, 1997).
4. M. A. Grayson, The heat equation shrinks embedded plane curves to round points, *J. Differential Geom.* **26**, 285 (1987).
5. J. S. Langer, Instabilities and pattern formation in crystal growth, *Rev. Modern Phys.* **52**, 1 (1980).
6. R. J. LeVeque, *Numerical Methods for Conservation Laws* (Birkhauser-Verlag, Basel, 1990).
7. M. McAllister, D. Kirkpatrick, and J. Snoeyink, A compact piecewise-linear Voronoi diagram for convex sites in the plane, *Discrete Comput. Geom.* **15**, 73 (1996).
8. S. Osher and B. Merriman, The Wulff shape as the asymptotic limit of a growing crystal interface, *Asian J. Math.* **1**, 560 (1997).
9. S. J. Osher and J. A. Sethian, Front propagation with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
10. P. J. Rasch and D. L. Williamson, On shape-preserving interpolation and semi-Lagrangian transport, *SIAM J. Sci. Stat. Comput.* **11**, 656 (1990).
11. A. Schmidt, Computation of three dimensional dendrites with finite elements, *J. Comput. Phys.* **125**, 293 (1996).
12. J. Sethian, *Level Set Methods* (Cambridge Univ. Press, Cambridge, UK, 1996).
13. J. A. Sethian and J. Strain, Crystal growth and dendritic solidification, *J. Comput. Phys.* **98**, 231 (1992).
14. P. K. Smolarkiewicz and J. Pudykiewicz, A class of semi-Lagrangian approximations for fluids, *J. Atmos. Sci.* **49**, 2082 (1992).
15. A. Staniforth and J. Côté, Semi-Lagrangian schemes for atmospheric models—A review, *Monthly Weather Rev.* **119**, 2206 (1991).
16. E. Stein, *Singular Integrals and Differentiability Properties of Functions* (Princeton Univ. Press, Princeton, NJ, 1970).
17. J. Strain, A boundary integral approach to unstable solidification, *J. Comput. Phys.* **85**, 342 (1989).
18. J. Strain, Fast tree-based redistancing for level set computations, *J. Comput. Phys.*, in press.
19. J. Strain, Semi-Lagrangian methods for level set equations, *J. Comput. Phys.*, in press.

20. J. Strain, Modular methods for moving interfaces, *J. Comput. Phys.*, in press.
21. M. Sussman, P. Smereka, and S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
22. J. Taylor, J. W. Cahn, and C. A. Handwerker, Geometric models of crystal growth, *Acta Met. Mat.* **40**, 1443 (1992).
23. C. Truesdell and R. A. Toupin, The classical field theories, in *Handbuch der Physik III/1*, edited by S. Flügge (Springer-Verlag, Berlin, 1960).
24. D. L. Williamson and P. J. Rasch, Two-dimensional semi-Lagrangian transport with shape-preserving interpolation, *Monthly Weather Rev.* **117**, 102 (1989).
25. G. Wulff, Zur Frage der Geschwindigkeit des Wachstums und der Auflösung der Krystallflachen, *Zeit. Krystall. Min.* **34**, 449 (1901).
26. C. K. Yap, An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments, *Discrete Comput. Geom.* **2**, 365 (1987).