Journal of Computational Physics **169,** 503–555 (2001)
doi:10.1006/jcph.2000.6657, available online at http://www.idealibrary.com on IDEAL®

Evolution, Implementation, and Application of Level Set and Fast Marching Methods for Advancing Fronts

J. A. Sethian[1]

Department of Mathematics, University of California, Berkeley, California 94720

Received May 19, 2000; revised September 28, 2000

A variety of numerical techniques are available for tracking moving interfaces. In this review, we concentrate on techniques that result from the link between the partial differential equations that describe moving interfaces and numerical schemes designed for approximating the solutions to hyperbolic conservation laws. This link gives rise to computational techniques for tracking moving interfaces in two and three space dimensions under complex speed laws. We discuss the evolution of these techniques, the fundamental numerical approximations, involved, implementation details, and applications. In particular, we review some work on three aspects of materials sciences: semiconductor process simulations, seismic processing, and optimal structural topology design.

1. Overview and Introduction

A large number of computational problems and physical phenomena involve the motion of interfaces separating two or more regions. These include problems in such areas as fluid mechanics, combustion, materials science, meteorology, and computer vision. In these problems, challenging issues often involve:

- interfaces that change topology, break, and merge as they move;
- formation of sharp corners, cusps, and singularities;
- dependence of the interface motion on delicate geometric quantities such as curvature and normal direction;
- complexities in three dimensions and higher; and
- subtle feedback between the physics and chemistry off the interface and the position and motion of the front itself.

[1] This work was supported in part by the Applied Mathematical Science subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098, and the Office of Naval Research under grant FDN00014-96-1-0381.

0021-9991/01 $35.00

One approach to formulating, modeling, and building computational techniques for some aspects of these problems is provided by level set methods. These techniques work by embedding the propagating interface as the zero level set of a time-dependent, implicit function, and then solving the resulting equations of motion in a fixed-grid Eulerian setting. They have been used with considerable success in a wide collection of settings, including fluid mechanics, crystal growth, combustion, and medical imaging. A general overview of the theory, numerical approximation, and range of applications may be found in [81].

Level set methods, introduced by Osher and Sethian [56], rely in part on the theory of curve and surface evolution given in [69] and on the link between front propagation and hyperbolic conservation laws discussed in [70]. They recast interface motion as a time-dependent Eulerian initial value partial differential equation, and they rely on viscosity solutions to the appropriate differential equations to update the position of the front, using an interface velocity that is derived from the relevant physics both on and off the interface. These viscosity solutions are obtained by exploiting schemes from the numerical solution of hyperbolic conservation laws. Level set methods are specifically designed for problems involving topological change, dependence on curvature, formation of singularities, and the host of other issues that often appear in interface propagation techniques. Over the past few years, various aspects of these techniques have been refined to the point where a general computational approach to arbitrary front propagation problems is available. This general computational approach allows one to track the motion of very complex interfaces, with significant and delicate coupling between the relevant physics and the interface motion.

Level set methods cast interface propagation in terms of a time-dependent initial value problem. More recently, a set of finite difference numerical techniques known as "fast marching methods" were developed by Sethian [75]; they were constructed to solve the Eikonal equation, which is a boundary value partial differential equation. These techniques rely on a marriage between the numerical technology for computing the solution to hyperbolic conservation laws and the causality relationships inherent in finite difference upwind schemes. Fast marching methods are Dijkstra-type methods, in that they are closely connected to Dijkstra's well-known network path algorithms [29]; however, they approximate the solution to the underlying Eikonal equation in a consistent manner. While the Eikonal equation itself describes some front propagation problems, the important link we shall emphasize in this review is that fast marching methods provide a general, efficient, and accurate way to actually implement some aspects of level set methods.

Both sets of techniques, that is, level set methods and fast marching methods, require an adaptive methodology to obtain computational efficiency. In the case of level set methods, this leads to the preferred narrow-band level set method introduced by Adalsteinsson and Sethian in [1]. In the case of fast marching methods [75], adaptivity and speed stem from the causality relationship and the use of heap data structures.

In this review, we discuss some aspects of the evolution and implementation of these techniques. We give pointers to some of the many applications and then focus on three in particular. First, we discuss interface propagation techniques for process simulation in semiconductor manufacturing, focusing on etching and deposition simulations. The goal in these simulations is to follow the profile evolution during the various stages of building a silicon chip. The evolving profile depends on such factors as material-dependent etch and deposition rates, visibility and masking, complex flux laws, and integral equations arising from reemission and redeposition processes. Here, we follow closely the work and text of Adalsteinsson and Sethian, [2–4]. Second, we discuss aspects of fast marching methods

applied to seismic processing, following closely the work and text of Sethian and Popovici [85]. Third, we discuss the application of level set techniques to optimal structural topology design; the goal is to design materials which can carry given loads and minimize the amount of material involved. Here, we follow closely the work and text of Sethian and Wiegmann [88].

I. FORMULATIONS OF MOVING INTERFACES, HYPERBOLIC EQUATIONS, AND CONNECTIONS WITH SHOCK SCHEMES

2. Characterizations of Moving Interfaces

2.1. *Mathematical Formulations*

There are at least three ways to characterize a moving interface, and none of them are new. Interestingly, each comes from its own branch of mathematics. For simplicity, we discuss the issues in two space dimensions, that is, a one-dimensional interface which is a simple closed curve $\Gamma(t)$ moving in two dimensions. Assume that a given velocity field $\mathbf{u} = (u, v)$ transports the interface. All three constructions carry over to three dimensions.

The geometric view. Suppose one parameterizes the interface, that is, $\Gamma(t) = (x(s,t), y(s,t))$. Then one can write (see, for example, [68]) the equations of motion in terms of individual components $\mathbf{x} = (x, y)$ as

$$\begin{aligned} x_t &= u\left(\frac{y_s}{\left(x_s^2 + y_s^2\right)^{1/2}}\right), \\ y_t &= -v\left(\frac{x_s}{\left(x_s^2 + y_s^2\right)^{1/2}}\right). \end{aligned} \tag{1}$$

This is a differential geometry view; the underlying fixed coordinate system has been abandoned, and the motion is characterized by differentiating with respect to the parameterization variable s. Since the front motion is categorized in terms of the speed normal to the interface, the above equation represents motion along that normal vector field.

The set theoretic view. Consider the characteristic function $\chi(x, y, t)$, where χ is one inside the interface Γ and zero otherwise. Then one can write the motion of the characteristic function as

$$\chi_t = \mathbf{u} \cdot \nabla \chi. \tag{2}$$

In this view, all the points inside the set (that is, where the characteristic function is unity) are transported under the velocity field.

The analysis view. Consider the implicit function $\phi : R^2 \times [0, \infty) \to R$, defined so that the zero level set $\phi = 0$ corresponds to the evolving front $\Gamma(t)$. Then the equation for the evolution of this implicit function corresponding to the motion of the interface is given by

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0. \tag{3}$$

2.2. *Discretizations*

Each of these views is perfectly reasonable, and each has spawned its own numerical methodology to discretize the equations of motion. Marker particle methods, also known as

string methods and nodal methods, discretize the geometric view and take a finite number of points to divide up the parameterization space S. Volume-of-fluid methods, also known as cell methods and volume fraction methods, use a fixed underlying grid and discretize the characteristic function, filling each cell with a number that reflects the amount of characteristic function contained in that cell. Level set methods approximate the partial differential equation for the time-dependent implicit function ϕ through a discretization of the evolution operators on a fixed grid.

These discretizations contain keys to both the virtues and the drawbacks of the various approaches.

- The geometric/marker particle view keeps the definition of a front sharp. Special attention is required when marker particles collide, because these collisions can create corners and cusps, as well as changes in topology. These techniques often go by names such as contour surgery, reconnection algorithms, etc.; at their core, they reflect user-based decisions about the level of resolution. In addition, this discrete parameterized characterization of the interface can be intricate for two-dimensional surfaces moving in three dimensions.
- The characteristic/volume-of-fluid approach straight forwardly applies in multiple dimensions, handling topological merger easily, since this results from Boolean operations on sets. It requires some method of differentiating the characteristic function χ; since by definition this object is discontinuous, one must devise an approximation to $\nabla\chi$ to perform the evolution update. This is typically done through algorithms which locally reconstruct the front from the volume or cell fractions and then use this reconstruction to build the appropriate transport terms.
- The implicit/level set approach extends to multiple dimensions and handles topological changes easily. In addition, because the function ϕ is defined everywhere and smooth in many places, calculation of gradients in the transport term, as well as geometric quantities such as normal derivatives and curvature, is straightforward. It requires a way of delineating the actual interface, since its location does not necessarily correspond to the discretization grid points.

2.3. *Implicit Formulations of Interface Motion*

To take this implicit approach, there are three additional issues.

- First, an appropriate theory and strategy must be chosen in order to select the correct weak solution once the underlying smoothness is lost; this is linked to the work on the evolution of curves and surfaces and the link between hyperbolic conservation laws and propagation equations (see Sethian [68–70]).
- Second, the Osher–Sethian level set technique which discretizes the above requires an additional space dimension to carry the embedding, and hence it is computationally inefficient for many problems. This is rectified through the adaptive narrow-band method given by Adalsteinsson and Sethian in [1].
- Third, since both the level set function and the velocity are now defined away from the original interface, appropriate extensions of these values must be constructed. These extension velocities have been explicitly constructed for a variety of specific problems; see, for example, [4, 19, 20, 51, 63, 86, 91, 103]. One general technique for doing so for arbitrary physics and chemistry problems is given by Adalsteinsson and Sethian in [5] through the use of fast marching methods to solve an associated equation which constructs these extensions.

2.4. *Interrelations between Techniques*

It is important to state at the outset that each of the above techniques has evolved to the point where they provide practical, efficient, and accurate methodologies for computing a host of computational problems involving moving interfaces. Marker particle methods have been around for a very long time and have been used in a collection of settings, including, for example, bubble interactions and fluid instabilities (see, for example, Bunner and Tryggvasson [17], Esmaeeli and Tryggvason [30, 31], and Glimm *et al*. [33, 34]). Volume-of-fluid techniques, starting with the initial work of Noh and Woodward [55] (see also [36]), have been used to handle shock interactions and fluid interfaces (see, for example, Puckett [59] and Popinet and Zaleski [58]). Level set techniques have been applied to a large collection of problems; general reviews may be found in [77, 78, 80, 81]; a popular review may be found in [79].[2] In companion articles in this issue, a variety of interface techniques and applications will be discussed in detail.

Finally, we note that the strict delineations between various approaches is not meant to imply that the various techniques have not influenced each other. Modern level set methods often use a temporary marker representation of the front to help build the extension velocities; volume-of-fluid methods use differentiation ideas in level set methods to help construct normal vectors and curvature values; and marker models often use an underlying fixed grid to help with topological changes. Good numerics is ultimately about getting things to work; the slavish and blind devotion to one approach above all others is usually a sign of unfamiliarity with the range of troubles and challenges presented by real applications.

3. Theory and Algorithms for Front Propagation

3.1. *Propagating Fronts, Entropy Conditions, and Weak Solutions*

To build up to the numerical implementation of the level set method introduced in [56], we review some of the background work. One of the main difficulties in solving the front propagation equations is that the solution need not be differentiable, even with arbitrarily smooth boundary data. This nondifferentiability is intimately connected to the notion of appropriate weak solutions. The goal is to construct numerical techniques which naturally account for this nondifferentiability in the construction of accurate and efficient approximation schemes and to admit physically correct nonsmooth solutions.

In [68, 69], the equation for a curve propagating normal to itself with a given speed F and which remains a graph as it moves was studied. Consider the simple speed function $F = 1$ and a front which is an initial periodic cosine curve, as shown in Fig. 1. In Fig. 1a, the front propagating with speed $F = 1$ passes through itself and becomes the double-valued swallowtail solution; this can be seen by noting that for the case $F = 1$, there is an exact solution to the equations of motion (Eqs. 1) given by the geometric view. This a perfectly reasonable view of the solution, but it is one that does not lend itself to the view of the front as a boundary between two regions.

However, suppose the moving curve is regarded as a physical interface separating two regions. From a geometrical argument, the front at time t should consist of only the set of all points located a distance t from the initial curve. Figure 1b shows this alternate

[2] An introductory web page may be found at www.math.berkeley.edu/∼sethian/level_set.html; this website provides a large number of Applets and tutorials to explain the various techniques and applications.

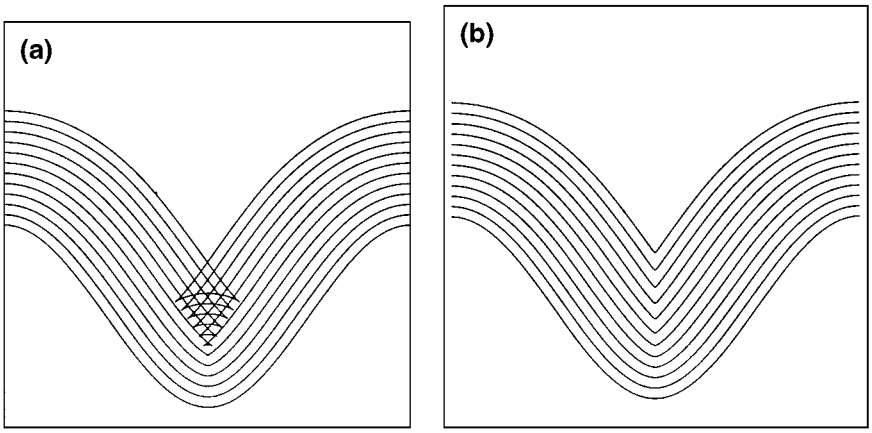

FIG. 1. Cosine curve propagating with unit speed. (a) Swallowtail ($F = 1.0$); (b) entropy solution ($F = 1.0$).

weak solution. Roughly speaking, one wants to remove the "tail" from the "swallowtail" (see [69]). One way to build this solution is through a Huygens principle construction; the solution is developed by imagining wave fronts emanating with unit speed from each point of the boundary data; the envelope of these wave fronts always corresponds to the "first arrivals." This will automatically produce the solution given on the right in Fig. 1. This is the approach taken in [69].

Another way to obtain the solution is through the notion of an entropy condition proposed in [68, 69]; if one imagines the boundary curve as a source for a propagating flame, then the expanding flame satisfies the requirement that once a point in the domain is ignited by the expanding front, it stays burnt. This construction also yields the entropy-satisfying Huygens's construction given in Fig. 1.

3.2. *Curvature-Driven Limits and Viscous Hyperbolic Conservation Laws*

Yet another way of obtaining this nondifferentiable weak solution after the occurrence of the singularity is through the limit of curvature-driven flows. Following the discussions in [69, 70], we consider now a speed function of the from $F = 1 - \epsilon\kappa$, where ϵ is a constant. The modifying effects of the term $\epsilon\kappa$ are profound and in fact pave the way toward constructing accurate numerical schemes that adhere to the correct entropy condition. Following [69], we can write a curvature evolution equation as

$$\kappa_t = \epsilon\kappa_{\alpha\alpha} + \epsilon\kappa^3 - \kappa^2, \tag{4}$$

where the second derivative of the curvature κ is taken with respect to arc length α. This is a reaction–diffusion equation; the drive toward singularities due to the reaction term $(\epsilon\kappa^3 - \kappa^2)$ is balanced by the smoothing effect of the diffusion term $(\epsilon\kappa_{\alpha\alpha})$.

Consider again the cosine front and the speed function $F(\kappa) = 1 - \epsilon\kappa, \epsilon > 0$. As the front moves, the trough is sharpened by the negative reaction term (because $\kappa < 0$ at such points) and smoothed by the positive diffusion term. For $\epsilon > 0$, it can be shown that the moving front stays smooth, as shown in Fig. 2a. However, with $\epsilon = 0$, one has a pure reaction equation $\kappa_t = -\kappa^2$, and the developing corner can be seen in the exact solution $\kappa(s, t) = \kappa(s, 0)/(1 + t\kappa(s, 0))$. This is singular in finite time t if the initial curvature is anywhere negative. The entropy solution to this problem when $F = 1$ is shown in Fig. 2b.

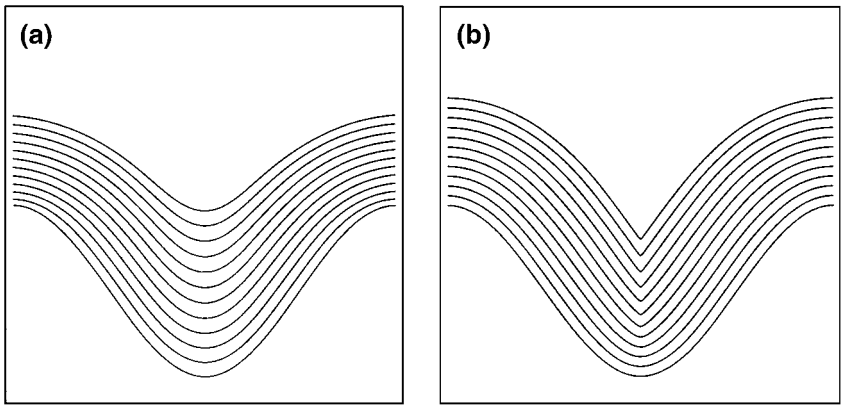

FIG. 2. Entropy solution is the limit of viscous solutions. (a) $F = 1 - 0.25\kappa$; (b) entropy solution ($F = 1.0$).

The limit of the curvature-driven flow as the curvature coefficient ϵ vanishes produces the entropy-limiting solution. This link can be seen more clearly by following the argument given in [70], which we now repeat. Consider the initial front given by the graph of $f(x)$, with f and f' periodic on [0, 1], and suppose that the propagating front remains a graph for all time. Let ψ be the height of the propagating function at time t, and thus $\psi(x, 0) = f(x)$. The tangent at (x, ψ) is $(1, \psi_x)$. The change in height V in a unit time is related to the speed F in the normal direction by

$$\frac{V}{F} = \frac{\left(1 + \psi_x^2\right)^{1/2}}{1}, \tag{5}$$

and thus the equation of motion becomes

$$\psi_t = F\left(1 + \psi_x^2\right)^{1/2}. \tag{6}$$

Use of the speed function $F(\kappa) = 1 - \epsilon\kappa$ and the formula $\kappa = -\psi_{xx}/(1 + \psi_x^2)^{3/2}$ yields

$$\psi_t - \left(1 + \psi_x^2\right)^{1/2} = \epsilon \frac{\psi_{xx}}{1 + \psi_x^2}. \tag{7}$$

This is a partial differential equation with a first-order time and space derivative on the left side and a second-order term on the right. Differentiation of both sides of this equation yields an evolution equation for the slope $u = d\psi/dx$ of the propagating front, namely,

$$u_t + \left[-(1 + u^2)^{1/2}\right]_x = \epsilon \left[\frac{u_x}{1 + u^2}\right]_x. \tag{8}$$

Thus, as shown in [70], the derivative of the curvature-modified equation for the changing height ψ looks like some form of a viscous hyperbolic conservation law, with $G(u) = -(1 + u^2)^{1/2}$ for the propagating slope u. Hyperbolic conservation laws of this form have been studied in considerable detail and our entropy condition is equivalent to the one for propagating shocks in hyperbolic conservation laws.

Finally, we point out that the most mathematically precise way of discussing nonsmooth solutions is through the idea of viscosity solutions introduced by Crandall and Lions [27, 28]; we refer the interested reader to those and associated papers for a complete discussion.

3.3. *Link to Numerical Schemes for Hyperbolic Conservation Laws*

Given this connection, the next step in development of PDE-based interface advancement techniques was to in fact exploit the considerable numerical technology for hyperbolic conservation laws to tackle front propagation itself. In such problems, schemes are specifically designed to construct entropy-satisfying limiting solutions and maintain sharp discontinuities wherever possible; these goals are required to keep fluid variables such as pressure from oscillating, and to make sure that discontinuities are not smeared out. This is equally important in the tracking of interfaces, in which one wants corners to remain sharp, and to intricate development so it can be accurately tracked. Thus, the strategy discussed in [70] was to transfer this technology to front propagation problems, and this view played a role in the level set method introduced by Osher and Sethian in [56].

II. BASIC ALGORITHMS FOR INTERFACE ADVANCEMENT

4. Level Set Methods: Basic Algorithms, Adaptivity, and Constructing Extension Velocities

The above discussion focused on curves which remain graphs. The numerical level set method given in [56] recasts the front in one higher dimension and uses the implicit analytic framework given in Section 2.1 to tackle problems which do not remain graphs; in addition, that work developed multidimensional upwind schemes to approximate the relevant gradients. Here, we briefly review low-order versions of those schemes before turning to issues of adaptivity and construction of extension velocities.

4.1. *Equations of Motion*

Level set methods rely on two central embeddings: the embedding of the interface as the zero level set of a higher dimensional function and the embedding (or extension) of the interface's velocity to this higher dimensional level set function. More precisely, given a moving closed hypersurface $\Gamma(t)$, that is, $\Gamma : [0, \infty) \to R^N$, propagating with a speed F in its normal direction, we wish to produce an Eulerian formulation for the motion of the hypersurface propagating along its normal direction with speed F, where F can be a function of various arguments, including the curvature, normal direction, etc. Let $\pm d$ be the signed distance to the interface. Suppose the propagating interface is embedded as the zero level set of a higher dimensional function ϕ. In other words, let $\phi\ (x, t = 0)$, where $x \in R^N$ is defined by

$$\phi(x, t = 0) = \pm d. \tag{9}$$

If this is done, then an initial value partial differential equation can be obtained for the evolution of ϕ, namely,

$$\phi_t + F|\nabla\phi| = 0 \tag{10}$$

$$\phi(x, t = 0) \quad \text{given.} \tag{11}$$

This is the implicit formulation of front propagation given in [56]. As discussed in [68–70], propagating fronts can develop shocks and rarefactions in the slope, corresponding to

corners and fans in the evolving interface, and numerical techniques designed for hyperbolic conservation laws can be exploited to construct schemes which produce the correct, physically reasonable entropy solution.

There are certain advantages associated with this perspective. First, it is unchanged in higher dimensions, that is, for surfaces propagating in three dimensions and higher. Second, topological changes in the evolving front Γ are handled naturally; the position of the front at time t is given by the zero level set $\phi(x, t) = 0$ of the evolving level set function. This set need not be connected and can break and merge as t advances. Third, terms in the speed function F involving geometric quantities such as the normal vector n and the curvature κ may be easily approximated through the use of derivative operators applied to the level set function, that is,

$$n = \frac{\nabla\phi}{|\nabla\phi|}, \qquad \kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}.$$

Fourth, the upwind finite difference technology for hyperbolic conservation laws may be used to approximate the gradient operators.

4.2. *Approximation Schemes*

Entropy-satisfying upwind viscosity schemes for this initial value formulation were introduced in [56]. One of the simplest first-order schemes is given as

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} - \Delta t[\max(F_{ijk}, 0)\nabla^{+}\phi + \min(F_{ijk}, 0)\nabla^{-}\phi], \tag{12}$$

where

$$\nabla^{+}\phi = \begin{bmatrix} \max\left(D_{ijk}^{-x}\phi, 0\right)^2 + \min\left(D_{ijk}^{+x}\phi, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{-y}\phi, 0\right)^2 + \min\left(D_{ijk}^{+y}\phi, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{-z}\phi, 0\right)^2 + \min\left(D_{ijk}^{+z}\phi, 0\right)^2 \end{bmatrix}^{1/2}$$

and

$$\nabla^{-}\phi = \begin{bmatrix} \max\left(D_{ijk}^{+x}\phi, 0\right)^2 + \min\left(D_{ijk}^{-x}\phi, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{+y}\phi, 0\right)^2 + \min\left(D_{ijk}^{-y}\phi, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{+z}\phi, 0\right)^2 + \min\left(D_{ijk}^{-z}\phi, 0\right)^2 \end{bmatrix}^{1/2}.$$

Here, we have used standard finite difference notation so that, for example,

$$D_{ijk}^{+x} = \frac{(\phi_{i+1,j,k} - \phi_{i,j,k})}{\Delta x}. \tag{13}$$

Higher order schemes are also available; see [56].

The above formulation reveals two central embeddings.

1. First, in the initialization step (Eq. (9)), the signed distance function is used to build a function ϕ which corresponds to the interface at the level set $\phi = 0$. This step is known as "initialization;" when performed at some later point in the calculation beyond $t = 0$, it is referred to as "reinitialization."

2. Second, the construction of the initial value PDE given in Eq. (10) means that the velocity F is now defined for *all* the level sets, not just the zero level set corresponding to the interface itself. We can be more precise by rewriting the level set equation as

$$\phi_t + F^{\text{ext}}|\nabla\phi| = 0, \tag{14}$$

where F^{ext} is some velocity field which, at the zero level set, equals the given speed F. In other words,

$$F^{\text{ext}} = F \quad \text{on } \phi = 0.$$

This new velocity field F^{ext} is known as the "extension velocity."

Both of these issues need to be confronted to efficiently apply level set methods to complex computational problems.

4.3. *Adaptivity: The Narrow-Band Level Set Method*

Equation 12 is an explicit scheme, and hence it can be solved directly. The time step requirement depends on the nature of the speed function F; for an F that depends only on position, the time step behaves like $\frac{\Delta t}{\Delta x} F \leq 1$. In the case when the speed function F depends on curvature terms (for example, $F = -\kappa$), the equation has a parabolic component, and hence the time step requirement resembles that of a nonlinear heat equation; the time step depends roughly on $\frac{\Delta t}{\Delta x^2}$.

In the level set formulation, both the level set function and the speed are embedded into a higher dimension. This then implies computational labor through the entire grid, which is inefficient. A rough operation count for the original level set method assumes N grid points in each space dimension of a three-dimensional problem. For a simple problem of straightforward propagation with speed $F = 1$, assuming that it takes roughly N time steps for the front to propagate through the domain (here, the CFL condition is taken almost equal to unity), this produces an $O(N^4)$ method.

Considerable computational speedup in the level set method comes from the use of the *narrow-band level set method*, introduced by Adalsteinsson and Sethian in [1]. It is clear that performing calculations over the entire computational domain is wasteful. Instead, an efficient modification is to perform work only in a neighborhood (or "narrow band") of the zero level set. This drops the operation count in three dimensions to $O(kN^3)$, where k is the number of cells in the narrow band. This is a significant cost reduction; it also means that extension velocities need only be constructed at points lying in the narrow band, as opposed to all points in the computational domain.

The idea of limiting computation to a narrow band around the zero level set was introduced in Chopp [22] and used in recovering shapes from images in Malladi *et al*. [50]. The idea is straightforward and can be best understood by means of figures, following the discussion in [78].

Figure 3 shows the zero level set corresponding to the front with a dark, heavy line, surrounded by a few neighboring level sets. Figure 4 shows the data structures used to keep track of the narrow band. The entire two-dimensional grid of data is stored in a square array. A one-dimensional object is then used to keep track of the points in the array (dark grid points in Fig. 4 are located in a narrow band around the front of a user-defined width) (see Fig. 4). Only the values of ϕ at such points within the tube are updated. Values of ϕ at grid points on the boundary of the narrow band are frozen. When the front moves near the edge

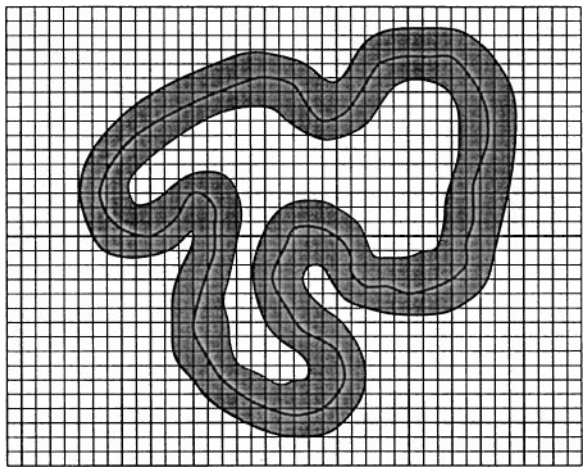

FIG. 3. Grid points in dark area are members of narrow band.

of the tube boundary, the calculation is stopped, and a new tube is built with the zero level set interface boundary at the center. This rebuilding process is known as "reinitialization."

Thus, the narrow-band method consists of the following loop:

- Tag "alive" points in narrow band.
- Build "land mines" to indicate near edge.
- Initialize "Far Away" points outside (inside) narrow band with large positive (negative) values.
- Solve level set equation until land mine hit.
- Rebuild; loop.

In the final step, this "rebuilding" requires some form of reinitialization to rebuild the signed distance function throughout the new narrow band. This is discussed in detail in Section 5.

Use of narrow bands leads to level set front advancement algorithms that are computationally equivalent in terms of complexity to traditional marker methods and cell techniques, while maintaining the advantages of topological merger, accuracy, and easy extension to multidimensions. Typically, the speed associated with the narrow-band method is about ten times faster on a 160×160 grid than the full matrix method. Such a speedup is substantial;

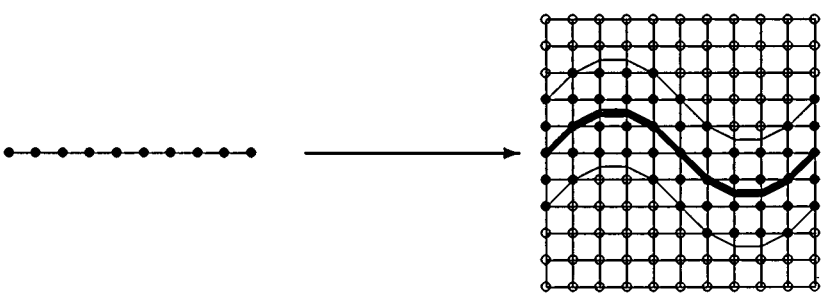

FIG. 4. Pointer array tags interior and boundary band points.

in three-dimensional simulations, it can make the difference between computationally intensive problems and those that can be done with relative ease. Details on the accuracy, typical tube sizes, and number of times a tube must be rebuilt may be found in Adalsteinsson and Sethian [1].

4.4. *Constructing Extension Velocities*

As discussed above, the characterization of an interface as an embedding in an implicitly defined function means that both the front and the velocity of the front are assumed to have meaning away from the actual interface (see Fig. 5). Thus, to be precise, one has

$$\phi_t + F^{\text{ext}}|\nabla\phi| = 0, \tag{15}$$

where F^{ext} is some velocity field which, at the zero level set, equals the given speed F. In other words,

$$F^{\text{ext}} = F \quad \text{on } \phi = 0.$$

There are several reasons why one needs to build these extension velocities.

1. There may be no natural speed function. In some physical problems, the velocity is given only at the front itself. For example, semiconductor manufacturing simulations of the etching and deposition process require determination of the visibility of the interface with respect to the etching/deposition beam (see [2–4], as well as later in this paper). There is no natural velocity off the front, since it is unclear what is meant by the "visibility" of the other level sets. In this case, an extension velocity must be specifically constructed.

2. Subgrid resolution may be required. In some problems, such as etch under very sharp material changes, the speed of the interface changes very rapidly or discontinuously as the front moves through the domain. In such cases, the exact location of the interface determines the speed, and constructing a velocity from the position of the interface itself, rather than from the coarse grid velocities, is desirable.

3. Accurate representation of front velocities may be needed. In some problems, the speed of the interface needs to be calculated from jump conditions or subtle relations involving the solution of an associated partial differential equation on either side of the interface; examples include Stefan problems and problems involving Rankine–Hugoniot speeds. The extension velocity view allows one to construct the correct front velocity and use this to move the front and the neighboring level sets.

4. Maintaining a nice level set representation is important. Under some velocities, such as those which arise in fluid mechanics simulations, the level sets have a tendency to either bunch up or spread out, which is seen when ϕ becomes either very steep or flat. The extension velocity discussed here is designed so that an initial signed distance function is essentially maintained as the front moves. We maintain a signed distance function for an important reason: by keeping a uniform separation for the level sets around the front, calculation of variables such as curvature becomes more accurate.

Suppose one chooses an incorrect velocity extension, one that does not maintain the signed distance function. Unchecked, this can cause the level set function to develop sharp and even discontinuous gradients across the zero level set. This means that calculations of quantities involving derivatives right at the interface, usually where one needs them the most, become highly suspect. Then the only hope is to repair the level set function each

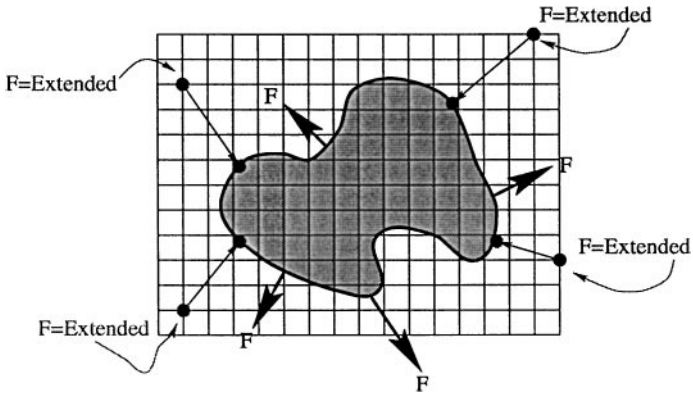

FIG. 5. Constructing extension velocities.

time step so that it is rebuilt as the signed distance function. This has the potential to add considerable error and expense to the algorithm; the process of reinitialization can itself move the location of the zero level set. Instead, we take the approach of building the correct extension velocity in that it maintains the signed distance function as the solution evolves, hence avoiding all reinitialization.

How much freedom does one have in the construction of this extension velocity F^{ext}? Beyond the requirement that it equal the velocity on the front itself, there is considerable freedom. The original level set calculations is [56] were concerned with interface problems with geometric propagation speeds, and hence an extension velocity was naturally built by using the geometry of each given level set. In more nongeometric or local applications, many different extension velocities have been employed. In many fluid simulations, one can choose to directly use the fluid velocity itself to act as F^{ext}. This is what was done by Rhee *et al*. [63] in a series of simulations of turbulent combustion. They built an extension velocity using an underlying elliptic partial differential equation coupled to a source term along the interface. This was also done in the two-phase flow simulations of Chang *et al*. [19] and Sussman *et al.* [91]. In these simulations, some bunching and flattenting of the level set function occurs. This is repaired at every time step through a reinitialization process which rebuilds the signed distance function using an iterative process given in [91].

When there is no choice available for an extension velocity, Malladi *et al*. [51] introduced the idea of extrapolating the velocity from the front. Their idea was to stand at each grid point and use the value of the speed function at the closest point on the front. Another approach is to build a speed function from the front using some other, possibly less physical quantity. Sethian and Strain [86] developed a numerical simulation of dendritic solidification; in this model, the velocity at the interface depended on a jump condition across the interface and hence had no meaning for the other "nonphysical" level sets. A boundary integral expression was developed for the velocity on the interface and evaluated both on and off the front to provide an extension velocity. The crystal growth study of Chen *et al*. [20] worked directly with the partial differential equations (rather than the conversion to a boundary integral) and built an extension velocity by solving an advection equation in each component, again coupled to a reinitialization procedure.

The important point is that the velocity field F^{ext} used to move the level sets neighboring the zero level set need have nothing to do with the velocity suggested by the physics in the

rest of the domain. It need only agree with the velocity F at the zero level set corresponding to the interface.

What are desirable properties of an extension velocity? Here, we follow the discussions in [5, 81]. First, it should match the given velocity on the front itself. Second, it is desirable that it moves the neighboring level sets in such a way that the signed distance function is preserved. Consider for a moment an initial signed distance function $\phi\ (x, t = 0)$, and suppose on builds an extension velocity which satisfies

$$\nabla F^{\text{ext}} \cdot \nabla \phi = 0. \tag{16}$$

It is straightforward to show that under this velocity field, the level set function ϕ remains the signed distance function for all time, assuming that both F and ϕ are smooth. To see that this is so (see [103]), suppose that initially $|\nabla \phi\ (x, t = 0)| = 1$, and one moves under the level set equation $\phi_t + F^{\text{ext}}|\nabla \phi| = 0$; then note that

$$\frac{d|\nabla \phi|^2}{dt} = \frac{d}{dt}(\nabla \phi \cdot \nabla \phi) = 2\nabla \phi \cdot \frac{d}{dt}\nabla \phi = -2\nabla \phi \cdot \nabla F^{\text{ext}}|\nabla \phi| - 2\nabla \phi \cdot \nabla |\nabla \phi| F^{\text{ext}}.$$

The first term on the right is zero because of the way the extension velocity is constructed; the second is sero because $|\nabla \phi(x, t = 0)| = 1$. Thus, the solution satisfies $|\nabla \phi| = 1$; this plus a uniqueness result for this differential equation show that $|\nabla \phi| = 1$ for all time.

Thus, the strategy introduced by Adalsteinsson and Sethian [5] uses a two-tiered system. Given a level set function at time n, namely ϕ_{ij}^n, one first constructs a signed distance function $\bar{\phi}_{ij}^n$ around the zero level set. Simultaneous with this construction, one then constructs the extension velocity F^{ext} satisfying Eq. (16). This velocity is used to update the level set function ϕ^n.

There are several important things to note about this approach:

- This construction finds an extension velocity which is then used to update the level set function. One can, of course, use as high an order method as desired for the level set update. If one wants to perform this update restricted to a narrow band using the narrow-band methodology of [1], one is free to do so. However, this methodology provides a way of doing so at all of the points where one wants to build this extension velocity.
- In this approach, one can choose never to reinitialize the level set function as follows:

1. Consider a level set function ϕ^n at time step $n\Delta t = 0$.

2. Build the extension velocity by simultaneously constructing a temporary signed distance function ϕ^{temp} and an extension velocity such that

$$\nabla \phi^{\text{temp}} \cdot \nabla F^{\text{ext}} = 0,$$

with ϕ^{temp} matching ϕ^n at their zero level sets, and F^{ext} matching the F given on the interface.

3. Then advance the level set function ϕ^n under the computed extension velocity to produce a new ϕ^{n+1} by solving $\phi_t + F^{\text{ext}}|\nabla \phi| = 0$.

This algorithm never reinitializes the evolving level set function, yet moves it under a velocity field that maintains the signed distance function. This avoids a large set of problems that have plagued some implementations of level set methods, namely that reinitialization steps can perturb the position of the front corresponding to the zero level set.

• In this approach, one explicitly finds the zero level set corresponding to the interface to build the extension velocity. This may seem slightly "illegal": one of the appealing features of level set methods is that the front need not be explicitly constructed and that all of the methodology may be executed on the underlying grid. Here, the front is explicitly built; however, one neither moves nor updates that representation. In cases of speed functions that depend on factors such as visibility, this is completely natural. The central virtue of level set methods lies in the update of the level set function on a discrete mesh to embed the motion of the interface itself. This strategy and philosophy are maintained.

Thus, given a front velocity F, this choice of extension velocity allows one to update an interface represented by an initial signed distance function in such a way that the signed distance function is maintained, and the front is never reinitialized. If one chooses to use the adaptive methodologies given in the narrow-band approach, occasional rebuilding of the narrow band may be required, but this is performed only occasionally.

4.5. *Summary*

In summary, two ideas which underpin level set methods are the link between schemes for hyperbolic fronts and propagating interfaces and the implicit formulation which embeds both the interface and the velocity field into one higher dimension, transforming front propagation into an initial value partial differential equation. To efficiently program level set methods, one also needs ways to find the signed distance function, both initially and to rebuild the narrow band. That is, one must quickly and accurately solve

$$|\nabla\phi| = 1, \quad \phi = 0 \quad \text{on } \Gamma.$$

In addition, one must solve the associated equation

$$\nabla\phi^{\text{temp}} \cdot \nabla F^{\text{ext}} = 0,$$

to efficiently and accurately build an extension velocity. Techniques for performing both of these steps result from fast marching methods, which we now discuss.

5. Fast Marching Methods for Reinitialization and Extension Velocities

Fast marching methods are finite difference techniques, more recently extended to unstructured meshes, for solving the Eikonal equation of the form

$$|\nabla T| F(x, y, z) = 1, \quad T = 0 \text{ on } \Gamma.$$

This can be thought of as a front propagation problem for a front initially located at Γ and propagating with speed $F(x, y, z,) > 0$. We note that this is a *boundary value* partial differential equation as opposed to an initial value problem given by level set methods, even though it describes a moving interface. This Eikonal equation describes a large number of physical phenomena, including those from optics, wave transport, seismology, photolithography, and optimal path planning, and fast marching methods have been used to solve these and a host of other problems. Our interest in this article will be confined only to using this Eikonal equation and fast marching method to construct efficient ways of reinitializing level set functions and constructing extension velocities. We refer the reader to [82] and [81] for a large collection of applications based on this technique.

5.1. *Fast Marching Methods*

Consider the upwind finite difference scheme for the Eikonal equation given by

$$\left[\begin{array}{l}\max\left(D_{ijk}^{-x}T, -D_{ijk}^{+x}T, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{-y}T, -D_{ijk}^{+y}T, 0\right)^2 \\ \quad + \max\left(D_{ijk}^{-z}T, -D_{ijk}^{+z}T, 0\right)^2\end{array}\right]^{1/2} = F_{ijk}, \tag{17}$$

which is related to the schemes discussed by Rouy and Tourin [64]. One approach to solving finite difference scheme (see [64]) is through iteration, which leads to an $O(N^4)$ algorithm in three dimensions, where N is the number of points in each direction. Instead, fast marching methods take a different approach.

The fast marching method, introduced in [75], is connected to Huygens's principle. The viscosity solution to the Eikonal equation $|\Delta T(x)| = F(x)$ can be interpreted through Huygens's principle in the following way: circular wavefronts are drawn at each point on the boundary, with the radius proportional to $F(x)$. The envelope of these wavefronts is then used to construct a new set of points, and the process is repeated; in the limit the Eikonal solution is obtained. The fast marching method mimics this construction; a computational grid is used to carry the solution u, and an upwind, viscosity-satisfying finite difference scheme is used to approximate this wavefront.

The order in which the grid values produced through these finite difference approximations are obtained is intimately connected to Dijkstra's method [29], which is a depth-search technique for computing shortest paths on a network. In that technique, the algorithm keeps track of the speed of propagation along the network links, fanning out along the network links to touch all the grid points. The fast marching method exploits a similar idea in the context of a continuous finite difference approximation to the underlying partial differential equation, rather than discrete network links.

In more detail, the fast marching method is as follows; we follow the presentation in [81, 82]. Suppose at some time the Eikonal solution is known at a set of points (denoted *Accepted* points). For every not-yet accepted grid point such that it has an accepted neighbor, a trial solution to the above quadratic Eq. (17) is computed, using the given values for u at accepted points and values of ∞ at all other points. Observe that the smallest of these trial solutions must be correct, since it depends only on accepted values which are themselves smaller. This "causality" relationship can be exploited to efficiently and systematically compute the solution as follows (see Fig. 6):

First, tag points in the initial conditions as *Accepted*. Then tag as *Considered* all points one grid point away and compute values at those points by solving Eq. (17). Finally, tag as *Far* all other grid points. Then the loop is:

1. Begin Loop: Let *Trial* be the *Considered* point with smallest value of T.
2. Tag as *Considered* all neighbors of *Trial* that are not *Accepted*. If the neighbor is in *Far*, remove it from that set and add it to the set *Considered*.
3. Recompute the values of T at all *Considered* neighbors of *Trial* by solving the piecewise quadratic equation according to Eq. (17).
4. Add point *Trial* to *Accepted*; remove from *Considered*.
5. Return to top until the *Considered* set is empty.

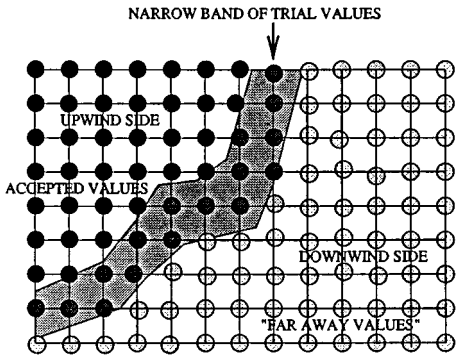

FIG. 6. Upwind construction of *Accepted* values.

The key to an efficient implementation of the above technique lies in a fast way of locating the grid point in the narrow band with the smallest value for T. An efficient scheme for doing so, discussed in detail in [81], can be devised using a min-heap structure, similar to what is done in Dijkstra's method. Given N elements in the heap, this allows one to change any element in the heap and reorder the heap in $O(\log N)$ steps. Thus, consider a mesh with N total points. Then the computational efficiency of the fast marching method for the mesh with N points is $O(N \log N)$; N steps to touch each mesh point with each step requiring $O(\log N)$, since the heap has to be reordered each time the values are changed.

The fast marching method evolved in part from examining the limit of the narrow-band level set method [1] as the band was reduced to one grid cell. Fast marching methods, by taking the perspective of the large body of work on higher order upwind, finite difference approximants from hyperbolic conservation laws, allow for higher order versions on both structured and unstructured meshes. The fast marching method has been extended to higher order finite difference approximations by Sethian in [82], first-order unstructured meshes by Kimmel and Sethian [40], and higher order unstructured meshes by Sethian and Vladimirsky [87]; see also photolithography applications in [76], a comparison of a similar approach with volume-of-fluid techniques in [35], a fast algorithm for image segmentation in [49], and computation of seismic traveltimes by Sethian and Popovici [85]. We also refer the reader to [96] for a different Dijkstra-like algorithm by Tsitsiklis which obtains the viscosity solution through a control-theoretic discretization which hinges on a causality relationship based on the optimality criterion.

Because we strongly suggest using the more accurate fast marching method introduced in [81, 82], we include it here for completeness. Following that discussion, we consider now the switch functions defined by

$$\text{switch}_{ijk}^{-x} = \begin{bmatrix} 1 & \text{if } T_{i-2,j,k} \quad \text{and} \quad T_{i-1,j,k} \text{ are known} \quad \text{and} \quad T_{i-2,j,k} \le T_{i-1,j,k} \\ 0 & \text{otherwise} \end{bmatrix},$$

$$\text{switch}_{ijk}^{+x} = \begin{bmatrix} 1 & \text{if } T_{i+2,j,k} \quad \text{and} \quad T_{i+1,j,k} \text{ are known} \quad \text{and} \quad T_{i+2,j,k} \le T_{i+1,j,k} \\ 0 & \text{otherwise} \end{bmatrix}.$$

(The expressions are similar in y and z.)

We can then use these operators in the fast marching method, namely,

$$\left[\begin{array}{l}\max\left[\left[D_{ijk}^{-x}T+\text{switch}_{ijk}^{-x}\frac{\Delta x}{2}D_{ijk}^{-x-x}T\right]-\left[D_{ijk}^{+x}T-\text{switch}_{ijk}^{+x}\frac{\Delta x}{2}D_{ijk}^{+x+x}T\right],0\right]^2\\ \quad+\max\left[\left[D_{ijk}^{-y}T+\text{switch}_{ijk}^{-y}\frac{\Delta y}{2}D_{ijk}^{-y-y}T\right]-\left[D_{ijk}^{+y}T-\text{switch}_{ijk}^{+y}\frac{\Delta y}{2}D_{ijk}^{+y+y}T\right],0\right]^2\\ \quad+\max\left[\left[D_{ijk}^{-z}T+\text{switch}_{ijk}^{-z}\frac{\Delta z}{2}D_{ijk}^{-z-z}T\right]-\left[D_{ijk}^{+z}T-\text{switch}_{ijk}^{+z}\frac{\Delta z}{2}D_{ijk}^{+z+z}T\right],0\right]^2\end{array}\right]^{1/2}=\frac{1}{F_{ijk}}. \tag{18}$$

This scheme attempts to use a second-order one-sided upwind stencil whenever points are available, but it reverts to a first-order scheme in the other cases. We note in addition that characteristics flow into the shocks, not out of them. The above method provides higher accuracy in regions of smoothness; the ultimate accuracy depends on the relationship of causality to shock lines in the solution. Numerical tests published in [81, 82] indicate a second-order method for a collection of test cases. For details and discussion, see [81, 82].

5.2. *Using Fast Marching Methods for Reinitialization and Extension Velocities*

We can now use the techniques given by Adalsteinsson and Sethian [5] which exploit fast marching methods to both reinitialize level set functions and construct extension velocities. Recall the step:

- Build the extension velocity by simultaneously constructing a temporary signed distance function ϕ^{temp} and an extension velocity such that

$$\nabla\phi^{\text{temp}}\cdot\nabla F^{\text{ext}}=0,$$

with ϕ^{temp} matching ϕ^n at their zero level sets, and F^{ext} matching the F given on the interface.

This can be done as follows. First, use the fast marching method to compute the signed distance ϕ^{temp} by solving the Eikonal equation

$$|\nabla T|=1$$

on either side of the interface, with the boundary condition that $T=0$ on the zero level set of ϕ. The solution T will then be the temporary signed distance function ϕ^{temp}. The fast marching method is run separately for grid points outside and inside the front (note that whether a grid point is inside or outside is immediately apparent from the sign of the level set function ϕ^n). The most accurate way to build values to initialize the fast marching heap is by actually finding the front using an accurate version of a contour plotter and then using this to build the nearby values; programmed correctly, this is both fast and accurate.

In this approach, we explicitly find the zero level set corresponding to the interface in order to reinitialize the front (and, as we shall see below, to build the extension velocity as well). This may seem slightly "illegal": one of the appealing features of level set methods is that the front need not be explicitly constructed and that all of the methodology may be executed on the underlying grid. Here, we choose to explicitly build the front. However, we neither move nor update that representation. In cases of speed functions that depend on

factors such as visibility, this is completely natural. The central virtue of level set methods lies in the update of the level set function on a discrete mesh to embed the motion of the interface itself. This strategy and philosophy are maintained.

Finding the zero level set is quite straightforward. As mentioned above, in two dimensions a contour plotter can be built. In three dimensions, any algorithm which discretizes a particular level set of an implicitly defined function can be used. We typically use a variant of the "marching cube" method discussed in [45], which builds a triangulated representation of the front. This can then be used to start the fast marching method for both constructing the signed distance function and for building the extension velocity, which we now discuss.

Once ϕ^{temp} is found, the next step is to extend a speed function which is given along an interface to grid points around the front. This construction should extend the speed in a continuous manner, and avoid, if possible, the introduction of any discontinuities in the speed close to the front.

Recall that we want to construct a speed function F^{ext} that satisfies the equation

$$\nabla F^{\text{ext}} \cdot \nabla \phi^{\text{temp}} = 0. \tag{19}$$

The idea is to march outward using the fast marching method, simultaneously attaching to each grid point both the distance from the front and the extended speed value. We first compute the signed distance ϕ^{temp} to the front using the fast marching method as described in the previous section. As the fast marching method constructs the signed distance at each grid point, one simultaneously updates the speed value F^{ext} according to Eq. (19). In the gradient stencil, we use only neighboring points close to the front to maintain the upwind ordering of the point construction. As an example of a first-order technique, assume that $(i+1, j)$ and $(i, j-1)$ are the points that are used in updating the distance; if v is the new extension value, it then has to satisify an upwind version of Eq. (19), namely,

$$\left(\frac{\phi^{\text{temp}}_{i+1,j} - \phi^{\text{temp}}_{i,j}}{h}, \frac{\phi^{\text{temp}}_{i,j} - \phi^{\text{temp}}_{i,j-1}}{h} \right) \cdot \left(\frac{F_{i+1,j} - v}{h}, \frac{v - F_{i,j-1}}{h} \right) = 0.$$

Since $(i+1, j)$ and $(i, j-1)$ are known, F is defined at those points, and this equation can be solved with respect to v to produce

$$v = \frac{F_{i+1,j}\left(\phi^{\text{temp}}_{i,j} - \phi^{\text{temp}}_{i+1,j}\right) + F_{i,j-1}\left(\phi^{\text{temp}}_{i,j} - \phi^{\text{temp}}_{i,j-1}\right)}{\left(\phi^{\text{temp}}_{i,j} - \phi^{\text{temp}}_{i+1,j}\right) + \left(\phi^{\text{temp}}_{i,j} - \phi^{\text{temp}}_{i,j-1}\right)}.$$

Similar expressions exist at other mesh points. Complete details on the use of fast marching methods to construct extension velocities may be found in [5].

These two steps allow one to efficiently reinitialize and build extension velocities; higher order fast marching methods provide more accurate versions of these constructions.

6. Extensions and Implementations

6.1. *Extensions*

There have been many algorithmic extensions to these basic ideas, considerably extending the range and applicability of these techniques. To mention only a few, these include variational level set methods to handle multiple differing interface types by Zhao *et al.*

[103] (see also [74]), multiple junctions by Merriman *et al*. [52], level set methods for unstructured meshes by Barth and Sethian, including terms for curvature flow [11], adaptive mesh refinement schemes by Milne [53], higher order fast marching methods [82], fast marching methods for manifolds by Kimmel and Sethian [40] as well as certain types of non-Eikonal static Hamilton–Jacobi equations by Sethian and Vladimirsky [87], level set flows in arbitrary co-dimension by Ambrosio and Sonar [7], hybrid methods, including coupled level set/volume-of-fluid techniques by Bourlioux [13], parallel versions [71], and extensions to motion under the intrinsic Laplacian of curvature by Chopp and Sethian in [25] and Chopp *et al*. in [26]. We refer the reader to these papers and the review in [81], as well as companion articles in this issue of the Journal. This paper is by no means meant to represent the large and rapidly growing body of work in these areas.

6.2. *Implementations*

There are a large number of ways to implement the details of these techniques. These include various high order schemes, iterative ways of performing reinitializations, variants on the narrow-band method, and alternative ways of building extension velocities. In this section, we would like to offer some comments which address some issues and implementation details.

6.2.1. Sources of error. There are several sources of error when level set methods are used to propagate fronts. These include:

- **Errors due to poor choices of extension velocities.** This can lead to distortion in the neighboring level sets, which can require reinitialization procedures to return the level set function to the signed distance function. If the extension velocity methodology described earlier is used, this will ensure, at least formally, that the signed distance function is maintained.
- **Error due to over use of reinitialization.** Reinitialization has a tendency to move the location of the interface. While higher order methods can help, including those that attempt to either redistribute mass or solve an associated constraint problem, our experience is that the best approach is to limit reinitialization. This is one of the reasons that the size of the narrow band in the narrow-band method is chosen large enough to limit reinitialization, rather than being restricted to a one-cell wide band which would force continuous reinitialization.
- **Error due to approximations in the gradient.** First order is usually not sufficient; the numerical diffusion causes sufficient error, and higher order schemes are recommended.
- **Time-stepping errors.** We typically use a Heun's method that is second order in time.

6.2.2. Operation counts. Next, we revisit the issue of operation counts. Consider a computational domain in three space dimensions with N points in each grid direction. An adaptive narrow-band method focuses all the computational labor onto a thin band around the zero level set, thus reducing the labor to $O(N^3k)$, where k is the width of this narrow band, providing the optimal technique for implementing level set methods. In contrast, the fast marching method is an optimal "adaptive" technique, which drops the computational labor involved in solving the boundary value formulation to $O(N^3 \log N)$. At first glance, the computational efficiency of fast marching methods may not be evident on the basis of these operation counts. However, two additional advantages provide the large computational savings. First, because the narrow-band level set method is solving a time-dependent problem, there is a constraint on the time step. The CFL number is based

on the speed F and controls the number of steps required to evolve a front. In contrast, the fast marching method has no such restrictions. The speed F of the front is irrelevant to the efficiency of the method. Second, the number of elements in the heap depends on the length of the front; in most cases, this length is small enough that, for all practical purposes, the sort is very fast and essentially $O(1)$. It is important to note that fast marching methods are methods for computing the solution to the Eikonal equation in all of space, not just in a neighborhood of the interface.

6.2.3. Separation of labor. One good programming design goal is to provide an environment in which the underlying physics and mathematical models that drive moving interfaces may be essentially decoupled from the numerical issues involved in characterizing and advancing these interfaces. While realistic interface problems typically involve significant and intricate feedback mechanisms between the interface the underlying physics, from a programming point of view the two steps can be effectively separated. Our approach is that the two key components, namely, (1) the update of the interface given a specific velocity field from the physics and (2) the construction of that velocity field from information determined by the interface, may be split apart, so that each views the other as a "black box."

Thus, one divides the physical problem into two fundamental components:

1. The user-supplied driver routines, which make calls to the interface routine.
2. The interface advancement routine, which has two functions.

- It can be queried to produce geometric data about the front, such as location, nodes along the front, local curvature, etc.
- Given a user-supplied velocity field along the interface, it can be used to advance the interface position.

By splitting codes in this manner, and building the general routines discussed earlier, robust software can be built and reused.

6.2.4. Flow of codes. Finally, we break down code flow for interface problems. We imagine the problem, somewhat abstractly, as follows:

- We are given an initial interface Γ, which may consist of several pieces.
- Given the position of the interface at any time, we are able to solve a set of partial differential equations on either side of the interface, using information about the interface location itself, as well as the value of certain quantities on the interface, to obtain the speed F on the interface.

A flow chart for the implementation is shown in Fig. 7.

III. THREE APPLICATIONS

The range of applications of level set and fast marching methods is vast, and we refer to only a few for bibliographic reference. These include work on semiconductor manufacturing [2–4, 35, 76, 84], geometry and minimal surfaces [9, 22–24, 72], combustion and detonation [10, 32, 63, 105, 106], fluids and surface-tension-driven flows [15, 19, 44, 54, 89–91, 102–104, 106], shape recognition and segmentation [16, 18, 46–48, 51, 67], crystal growth [20, 86], liquid bridges [21], groundwater flow [37], constructing geodesics [38, 40, 41], robotic navigation and path planning [39], inverse problems [65], grid generation [73], and seismology [85].

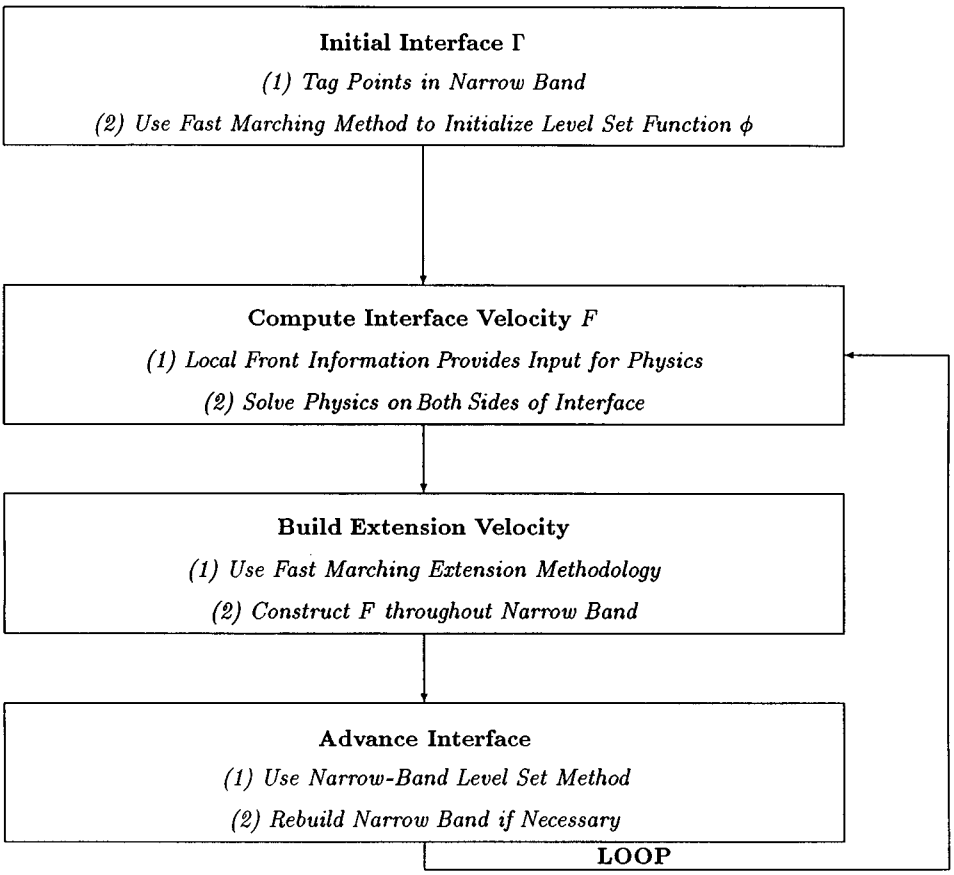

FIG. 7. Flow chart for implementing narrow-band level set methods.

In Fig. 8, we give a perspective on how some of these topics are related. There are many other contributors to the evolution of these ideas; the chart is meant to give one perspective on how the theory, algorithms, and applications have evolved. The text and bibliography of [81] give a somewhat more complete sense of the literature and the range of work underway.

In the next sections, we discuss three applications in detail. The first, semiconductor processing, is chosen because it requires much of the above methodology to obtain the accuracy, efficiency, and robustness required in semiconductor manufacturing, and because the results have been so closely matched with experiment. The second, seismic processing, is chosen because of the need for the great speed provided by fast marching methods. The third, optimal design of materials, is chosen because of the requirement of delicate elliptic solvers, and because of the more unusual nature of the application.

7. Interface Schemes for Semiconductor Processing

The first major application we consider is the application of these front propagation techniques to tracking interfaces in the microfabrication of electronic components. The goal is to follow the changing surface topography of a wafer as it is etched, layered, and shaped during the manufacturing process. These simulations rest on many of the previously

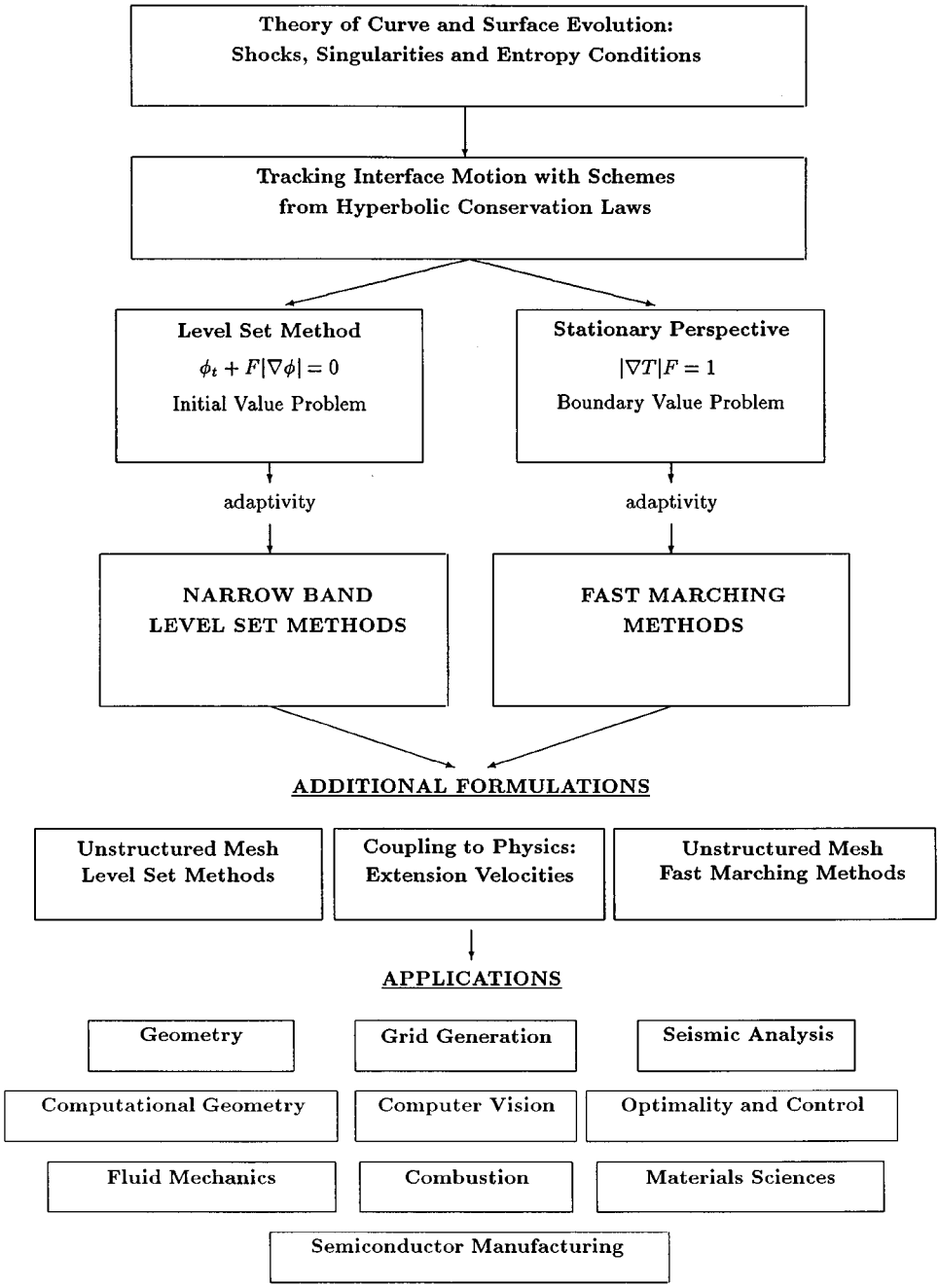

FIG. 8. Algorithms and applications for interface propagation.

discussed techniques, including narrow-band level set methods, fast marching methods for the Eikonal equation, and construction of extension velocities. In addition, they require attention to such issues as masking, discontinuous speed functions, visibility determinations, algorithms for subtle speed laws depending on second derivatives of curvature, and fast integral equation solvers. In this section, we follow closely the text in [2–4, 81].

7.1. *Physical Effects and Background*

The goal of numerical simulations in microfabrication is to model the process by which silicon devices are manufactured. Here, we briefly summarize some of the physical processes. First, a single crystal ingot of silicon is extracted from molten pure silicon. This silicon ingot is then sliced into several hundred thin wafers, each of which is then polished to a smooth finish. A thin layer of crystalline silicon is then oxidized, a light-sensitive "photoresist" is applied, and the wafer is then covered with a pattern mask that shields part of the photoresist. This pattern mask contains the layout of the circuit itself. Under exposure to a light or an electron beam, the exposed photoresist polymerizes and hardens, leaving an unexposed material that is then etched away in a dry etch process, revealing a bare silicon dioxide layer. Ionized impurity atoms such as boron, phosphorus, and argon are then implanted into the pattern of the exposed silicon wafer, and silicon dioxide is deposited at reduced pressure in a plasma discharge from gas mixtures at a low temperature. Finally, thin films such as aluminum are deposited by processes such as plasma sputtering, and contacts to the electrical components and component interconnections are established. The result is a device that carries the desired electrical properties.

These processes produce considerable changes in the surface profile as it undergoes various effects of etching and deposition. This problem is known as the "surface topography problem" in microfabrication and is controlled by many physical factors, including the visibility of the etching and deposition source from each point of the evolving profile, surface diffusion along the front, complex flux laws that produce faceting, shocks, and rarefactions, material-dependent discontinuous etch rates, and masking profiles.

The underlying physics and chemistry that contribute to the motion of the interfce profile are very much areas of active research. Nonetheless, once empirical models are formulated, the problem ultimately becomes the familiar one of tracking an interface moving under a speed function F. Simulations and text in this chapter are taken in part from Adalsteinsson and Sethian [2–4]; complete details may be found therein (see [84] for a review).

The underlying physical effects involved in etching, deposition, and lithography are quite complex. The effects may be summarized briefly as follows:

- *Deposition:* Particles are deposited on the surface, which causes buildup in the profile. The particles may either isotropically condense from the surroundings (known as chemical or "wet" deposition) or be deposited from a source. In the latter case, particles leave the source and deposit on the surface; the main advantage of this approach is increased control over the directionality of surface deposition. The rate of deposition, which controls the growth of the layer, may depend on source masking, visibility effects between the source and surface point, angle-dependent flux distribution of source particles, and the angle of incidence of the particles relative to the surface normal direction. In addition, particles might not stick, but in fact be reemitted back into the domain. This process is known as "reemission" and the "sticking coefficient" between zero and one is the fraction of particles that stick. A sticking coefficient of unity means that all particles stick. Conversely, a low sticking coefficient means that particles may bounce many times before they eventually become fixed to the surface.
- *Etching:* Particles remove material from the evolving profile boundary. The material may be isotropically removed, known as chemical or "wet" etching, or chipped away through reactive ion etching, also known as "ion milling." Similar to deposition, the main advantage of reactive ion etching is enhanced directionality, which becomes increasingly important as

device sizes decrease substantially and etching must proceed in vertical directions without affecting adjacent features. The total etch rate consists of an ion-assisted rate and a purely chemical etch rate due to etching by neutral radicals, which may still have a directional component. As in the above, the total etch rate due to wet and directional milling effects can depend on source masking, visibility effects between the source and surface point, angle-dependent flux distribution of source particles, and the angle of incidence of the particles relative to the surface normal direction. In addition, because of chemical reactions that take place on the surface, etching can cause surface particles to be ejected; this process is known as "redeposition." The newly ejected particles are then deposited elsewhere on the front, depending on their angle and distribution.

- *Lithography:* The underlying material is treated by an electromagnetic wave that alters the resist property of the material. The aerial image is found, which then determines the amount of crosslinking at each point in the material. This produces the etch/resist rate at each point of the material. A profile is then etched into the material, where the speed of the profile in its normal direction at any point is given by the underlying etch rate.

We now formalize the above. Define the coordinate system with the x and y axes lying in the plane, and with z being the vertical axis. Consider a periodic initial profile $h(x, y)$, where h is the height of the surface above the x–y plane, as well as a source Z given as a surface above the profile; we write $Z(x, y)$ as the height of the source at (x, y). Define the source ray as the ray leaving the source and aimed toward the surface profile. Let ψ be the angle variation in the source ray away from the negative z axis; ψ runs from 0 to π, though it is physically unreasonable to have $\pi/2 < \phi < \pi$. Let γ be the angle between the projection of the source ray in the x–y plane and the positive x axis. Let n be the normal vector at a point x on the surface profile and θ be the angle between the normal and the source ray.

In Fig. 9, these variables are indicated. Masks, which force flux rates to be zero, are indicated by heavy dark patches on the initial profile. At each point of the profile, a visibility

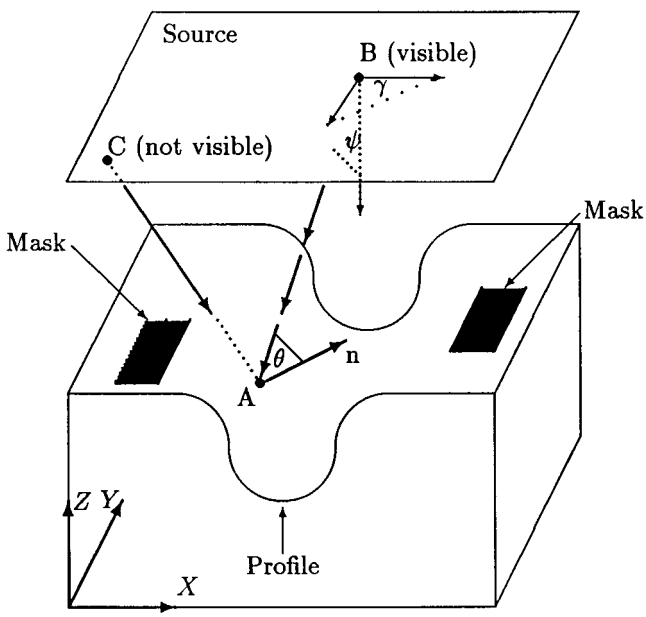

FIG. 9. Variables and setup.

indicator function $M_\Upsilon(x, x')$ is assigned; this indicates whether the point x on the initial profile can be seen by the source point x'.

7.2. *Equations of Motion for Etching and Deposition*

The goal is to write the effects of deposition and etching on the speed F at a point x on the front.

7.2.1. Etching. We consider two separate types of etching:

- $F^{\text{Etching}}_{\text{Isotropic}}$: *Isotropic etching* is uniform etching, also known as chemical or wet etching.
- $F^{\text{Etching}}_{\text{Direct}}$: *Direct etching* is etching from an external source; this can be either a collection of point sources or an external stream coming from a particular direction. Visibility effects are included, and the flux strength can depend on both the solid angle from the emitting source and the angle between the profile normal and the incoming source direction. Etching can include highly sensitive dependence on angle such as in ion milling.

7.2.2. Deposition. We consider four separate types of deposition:

- $F^{\text{Deposition}}_{\text{Isotropic}}$: *Isotropic deposition* is uniform deposition, also known as chemical or wet deposition.
- $F^{\text{Deposition}}_{\text{Direct}}$: *Direct deposition* involves deposition from an external source; this can be either a collection of point sources or an external stream coming from a particular direction. Visibility effects are included and the flux strength can depend on both the solid angle from the emitting source and the angle between the profile normal and the incoming source.
- $F^{\text{Deposition}}_{\text{Redeposition}}$: *Redeposition* involves particles that are expelled during the etching process. These particles then attach themselves to the profile at other locations. The strength and distribution of the redeposition flux function can depend on factors such as the local angle. A redeposition coefficient, $\beta_{\text{Redeposition}}$, which can range from zero to unity, represents the fraction of redeposition that results from the etching process. A value of $\beta_{\text{Redeposition}} = 1$ means that nothing is redeposited and everything sticks.
- $F^{\text{Deposition}}_{\text{Reemission}}$: In *reemission deposition*, particles that are deposited by direct deposition might not stick and may be reemitted into the domain. The amount of particles reemitted depends on a sticking coefficient $\beta_{\text{Reemission}}$. If $\beta_{\text{Reemission}} = 1$, nothing is reemitted.

In Fig. 9, we generalize all of these effects as the "source." The plane source is shown in the figure may consist of locations which emit either unidirectional or point-source contributions.

7.3. *Assembling the Terms*

We may, somewhat abstractly, assemble the above terms into the single expression:

$$F = F^{\text{Etching}}_{\text{Isotropic}} + F^{\text{Etching}}_{\text{Direct}} + F^{\text{Deposition}}_{\text{Isotropic}} + F^{\text{Deposition}}_{\text{Direct}} + F^{\text{Deposition}}_{\text{Redeposition}} + F^{\text{Deposition}}_{\text{Reemission}}. \quad (20)$$

The two isotropic terms are evaluated at a point x by simply evaluating the strengths at that point. The two direct terms are evaluated at a point x on the profile by first computing the visibility to each point of the source and then evaluating the flux function. These terms require computing an integral over the entire source. To compute the fifth term at a point x, we must consider the contributions of every point on the profile to check for redeposition

particles arising from the etching process; thus this term requires computing an integral over the profile itself. The sixth term, $F^{\text{Deposition}}_{\text{Reemission}}$, is more problematic. Since every point on the front can act as a deposition source of reemitted particles that do not stick, the total flux deposition function comes from evaluating an integral equation along the entire profile.

In more detail, let Ω be the set of points on the evolving profile at time t, and let *Source* be the external source. Given two points x and x', let $\Upsilon(x, x')$ be one if the points are visible from one another and zero otherwise. Let r be the distance from x to x', let $\mathbf{n}$ be the unit normal vector at the point x, and finally, let $\boldsymbol{\alpha}$ be the unit vector at the point x' on the source pointing toward the point x on the profile. Then we may refine the above terms as:

$$\begin{aligned} F = {} & \text{Flux}^{\text{Etching}}_{\text{Isotropic}} + \int_{\text{Source}} \text{Flux}^{\text{Etching}}_{\text{Direct}}(r, \psi, \gamma, \theta, x)\Upsilon(x, x')(\mathbf{n}\cdot\boldsymbol{\alpha})\, dx' \\ & + \text{Flux}^{\text{Deposition}}_{\text{Isotropic}} + \int_{\text{Source}} \text{Flux}^{\text{Deposition}}_{\text{Direct}}(r, \psi, \gamma, \theta, x)\Upsilon(x, x')(\mathbf{n}\cdot\boldsymbol{\alpha})\, dx' \\ & + \int_{\Omega} (1 - \beta_{\text{Redeposition}})\text{Flux}^{\text{Deposition}}_{\text{Redeposition}}(r, \psi, \gamma, \theta, x)\Upsilon(x, x')(\mathbf{n}\cdot\boldsymbol{\alpha})\, dx' \\ & + \int_{\Omega} (1 - \beta_{\text{Reemission}})\text{Flux}^{\text{Deposition}}_{\text{Reemission}}(r, \psi, \gamma, \theta, x)\Upsilon(x, x')(\mathbf{n}\cdot\boldsymbol{\alpha})\, dx'. \end{aligned} \tag{21}$$

7.4. *Evaluating the Terms*

The integrals are performed in a straightforward manner. The front is located is located by constructing the zero level set of ϕ; it is represented in two dimensions by a collection of line segments and in three dimensions by a collection of voxel elements; see [2, 3]. The centroid of each element is taken as the control point, and the individual flux terms are evaluated at each control point. In the case of the two isotropic terms, the flux is immediately found. In the case of the two integrals over sources, the source is suitably discretized and the contributions summed. In the fifth term, corresponding to redeposition, the integral over the entire profile is calculated by computing the visibility to all other control points, and the corresponding redeposition term is produced by the effect of direct deposition. Thus, the fifth term requires N^2 evaluations, where N is the number of control points which approximate the front.

7.4.1. Evaluation of the reemission term. The sixth and last term is somewhat more time consuming to evaluate, since it requires evaluation of the flux $\text{Flux}^{\text{Deposition}}_{\text{Reemission}}$ from each point of the interface, each of which depends on the contribution from all other points. Thus, this is an integral equation which must be solved to produce the total deposition flux at any point. When discretized, it produces a full, nonsymmetric matrix which must be solved at every time step to compute the relevant flux. In [4], a recurrence relationship is developed which allows a quick way of solving this discrete integral equation. This approach constructs an iterative solution to the integral equation, based on a series expansion of the interaction matrix. Fortunately, the iterative solution reduces to a simple matrix–vector multiplication and an error bound can be established to predict the number of iterations (which can be thought of as terms in the expansion) to compute the solution to the desired degree of accuracy.

This problem is a good example of the necessity of constructing extension velocities. There is no readably available and physical definition of the velocity off the interface with which to move the neighboring level sets. Consequently, the extension velocity methodology

described earlier can be used to construct extension velocities in the narrow-band level set method.

7.4.2. Visibility. To evaluate these terms above, we need to compute the visibility, that is, to find out if a point on the front is illuminated by another point on the front (or, in some cases, by the source itself). This visibility issue is common to a host of other problems, including scene rendering in computer graphics, ray tracing, and optimal placement of transmitters. This is a time-consuming component of any calculation; programmed directly, it requires $O(N^3)$ evaluations, where there are N points on the front. This is because each point must determine whether it can see each of N other points, and there are N intermediate points which might block the visibility.

Fortunately, a very fast way of determining the visibility is offered by a combination of level set methods and fast marching methods (see [2–4]). In the first step, we determine the signed distance function away from the interface using the fast marching method as discussed above. Armed with this, we may easily determine if two points on the front see each other by checking the sign of this signed distance function along the line segment connecting the two points; if this function changes sign, then the two points cannot see each other. This search may be done in a binary fashion, rendering a rapid way of determining visibility. For details, see [2–4].

7.4.3. Surface diffusion. An additional physical effect comes from surface diffusion, which relates to the motion of metal boundaries. It can be shown that this is connected to motion by the intrinsic Laplacian of curvature (see [25] for details). The basic physical idea behind this motion is that it evolves to balance out the interface so that the final state has constant curvature. It can be shown that the enclosed volume is constant under this motion, and this can serve to develop fast methods. In two dimensions, it reduces to motion by the second derivative of curvature. Thus, we need to add an additional term of the form

$$F = 1 + \epsilon\kappa_{\alpha\alpha}, \tag{22}$$

where α is an arc-length parameterization. The problem is delicate because Eq. (22) leads to a level set equation which is a time-dependent fourth-order partial differential equation, and the presence of the fourth derivative requiress an exceedingly small time step for stability in an explicit scheme; the linear fourth-order heat equation has a stability time step requirement of the form $O(\frac{\Delta t}{\Delta h^4})$. We make use of the methodology given by Chopp and Sethian in [25]. Approximations and fast methods for solving this sort of flow may be found in [26].

7.5. *Results*

7.5.1. Photolithography development. We begin the three-dimensional simulations with a problem in photolithography. Once the electromagnetic and optical simulations are performed, the problem of photolithography development reduces to that of following an initially plane interface propagating downward in three dimensions. The speed in the normal direction is given as a supplied rate function at each point. The speed $F = F(x, y, z)$ depends only on position; however, it may change extremely rapidly. The goal in lithography development is to track this evolving front. To develop realistic structures in three-dimensional development profiles, a grid of size $300 \times 300 \times 100$ is not unreasonable. The higher order fast marching method is of considerable value in the development step. As an example, a rate function calculated using the three-dimensional exposure and post-exposure

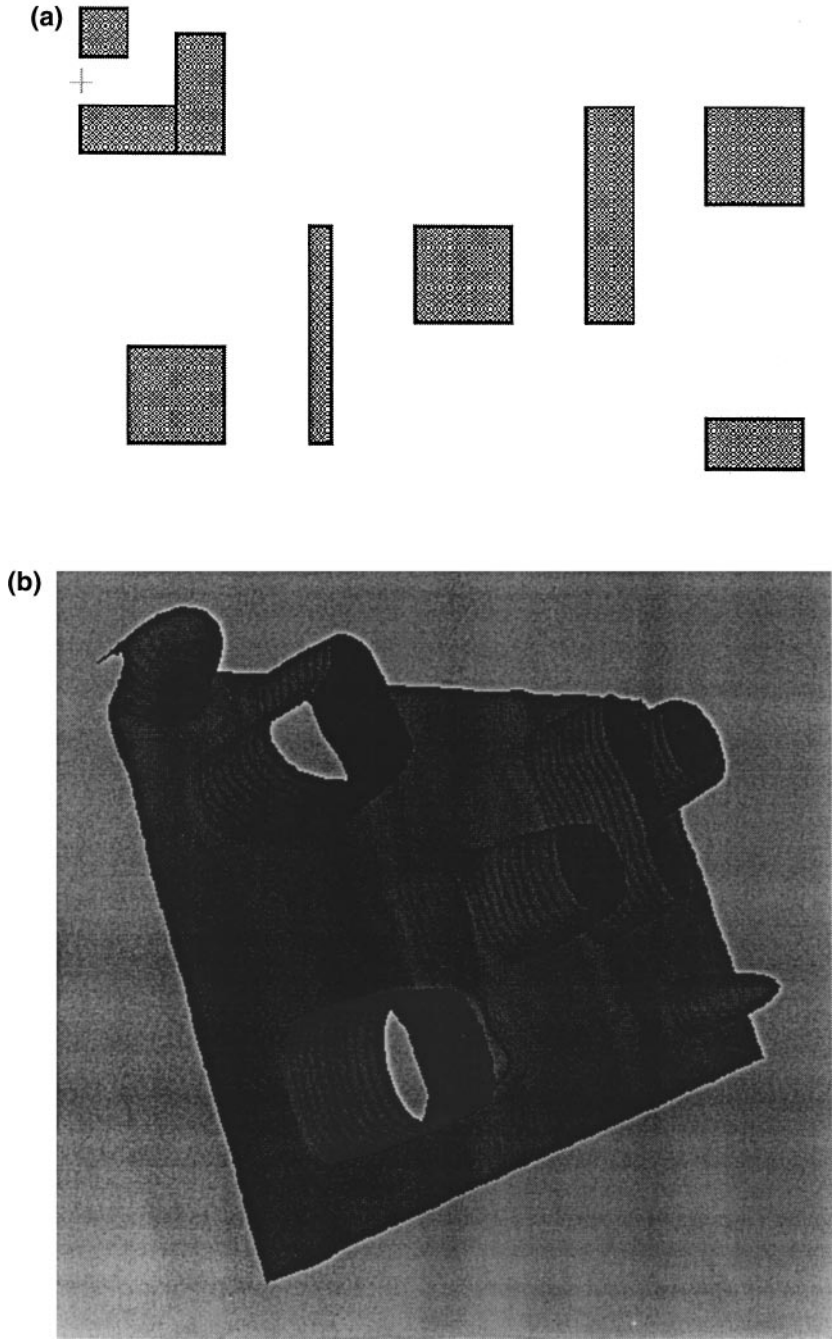

FIG. 10. Lithographic development using fast marching method. (a) Masking pattern; (b) lithographic development; view from below.

bake modules of TMA's Depict 4.0 [93] has been coupled to the fast marching method. Figure 10a shows the top view of a mask placed on the board. The dark areas correspond to areas that are exposed to light. The presence of standing waves, caused by the reflectivity of the surface, can easily be seen. In Fig. 10b a view of the developed profile is shown from

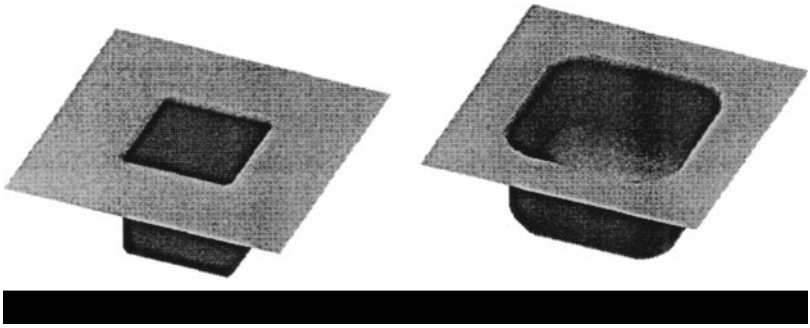

FIG. 11. Isotropic etching into a hole.

underneath; the etching of the holes and the presence of standing waves can be seen easily. For further results, see [76].

7.5.2. Etching and deposition. Next, we show a straightforward calculation of isotropic etching into a hole, taken from [3]. In Fig. 11 we show a square hole from which a material is being isotropically etched, corresponding to a simple speed function of $F = -1$. As expected, the sides of the cavity are cleanly etched away, leaving smoothed, rounded walls.

We follow with a calculation of source deposition from a plate located above the hole. The effects of visibility and shading are included. Along the entire plate, deposition material is emitted uniformly in each direction. In Fig. 12, we show three three-dimensional time plots of the evolving profile. The trench begins to pinch off due to the effects of visibility, and a bulb-shaped profile evolves.

We end the basic calculations with the modeling (Fig. 13) of the effect of nonconvex sputter etch/ion milling of a saddle surface. The nonconvex speed law $F = (1 + 4\sin^2(\theta))\cos\theta$ causes faceting of sharp corners and rounded polishing; for details of this effect, see [3]. Here, we use schemes for nonconvex Hamiltonians given in [57] for the level set update.

7.5.3. Complex simulations. Next, we include an example of three-dimensional effects of redeposition. The initial shape is a double-L, and we consider a combination of two cosine flux deposition sources. That is, the initial flux at each point is given by

$$\text{Flux}(x) = \cos^5(\theta_1)\cos(\theta_2) + \cos(\theta_1)\cos(\theta_2); \tag{23}$$

in addition, the second deposition term is given a sticking coefficient of 0.1, and thus we also consider the effects of redeposition. Here θ_1 is the angle that the vector v from x to y makes with the normal at x, and θ_2 is the angle that the vector v makes with the vertical. The

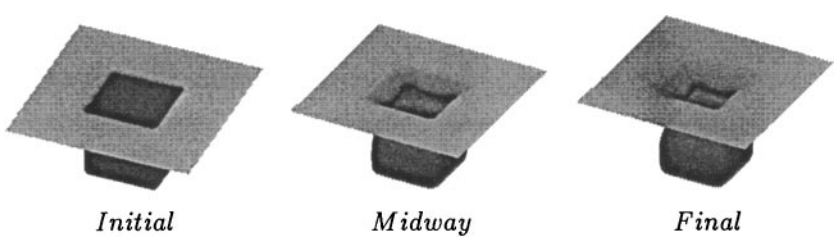

FIG. 12. Source deposition into a hole.

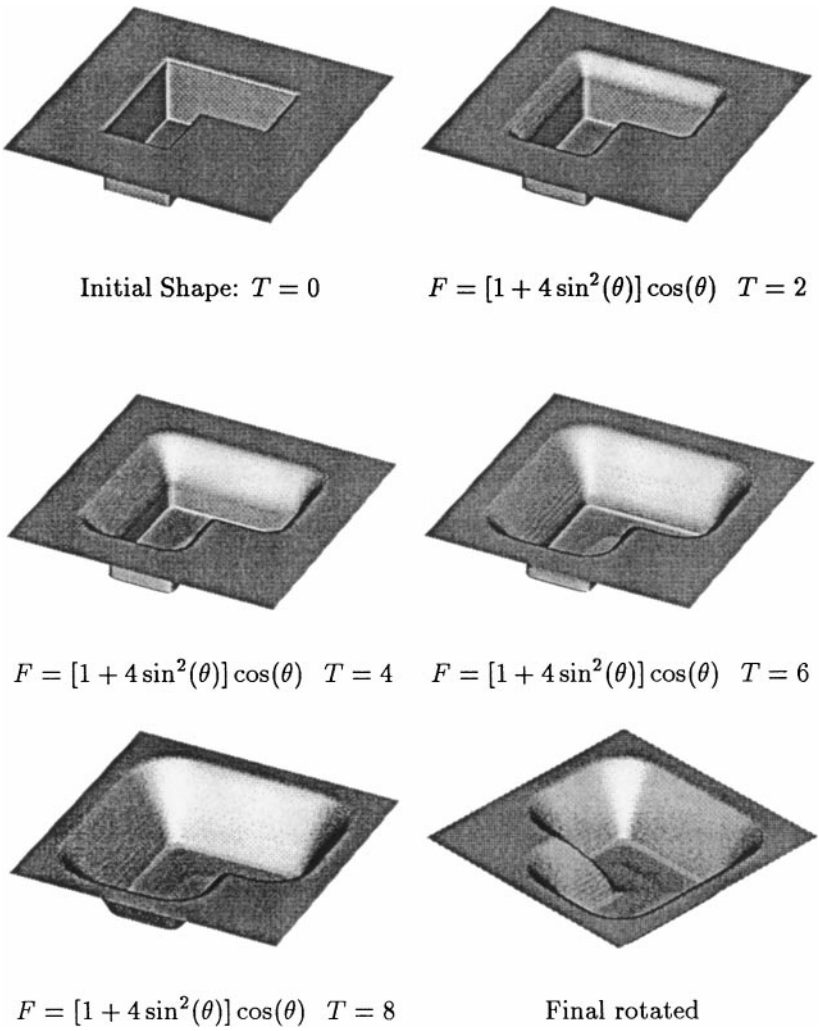

FIG. 13. Downward saddle under sputter etch.

results are shown after some time evolution in Fig. 14b; a two-dimensional cross-sectional cut is shown in Fig. 14c. For more simulations, see [4].

7.5.4. Timings. The computational labor required in these calculations depends on the grid resolution required to represent the front and the complexity of the physical effects under consideration. Tables I and II give rough timings for various sizes and physical complexities for a Sun Ultra. The lithography timings were computed using the fast marching method given in [75].

7.6. *Validation with Experimental Results*

We end with a collection of applications of the level set–fast marching methodology which compare simulations with experiment to analyze various aspects of surface thin film physics. The simulations in this section are performed using either TERRAIN[3] (a commercial version

[3] We thank Juan Rey, Brian Li, and Jiangwei Li for providing these results.

TABLE I
Two-Dimensional Timings

Test	50 by 50			100 by 100		
	Run time	Steps	Time/step	Run time	Steps	Time/step
Lithography (fast marching)	6.9 ms	NA	NA	26 ms	NA	NA
Isotropic (narrow band)	82 ms	24	34 ms	0.4 s	49	8 ms
Unidirectional (with visibility)	0.4 s	17	23 ms	2.3 s	34	70 ms
Etching and redeposition	1.7 s	25	68 ms	14 s	51	0.3 s
Deposition and redeposition (iterative model)	1.1 s	17	65 ms	12 s	39	0.3 s

TABLE II
Three-Dimensional Timings

Test	40 by 40 by 40			80 by 80 by 80		
	Run time	Steps	Time/step	Run time	Steps	Time/step
Lithography (fast marching)	0.16 s	NA	NA	2.1 s	NA	NA
Isotropic (narrow band)	1.3 s	8	0.16 s	13.6 s	24	0.6 s
Unidirectional (with visibility)	16.7 s	24	0.7 s	270 s	47	5.7 s
Etching and redeposition	224 s	12	19 s	260 m	25	10 m
Deposition and redeposition (iterative model)	265 s	11	24 s	290 m	23	12.6 m

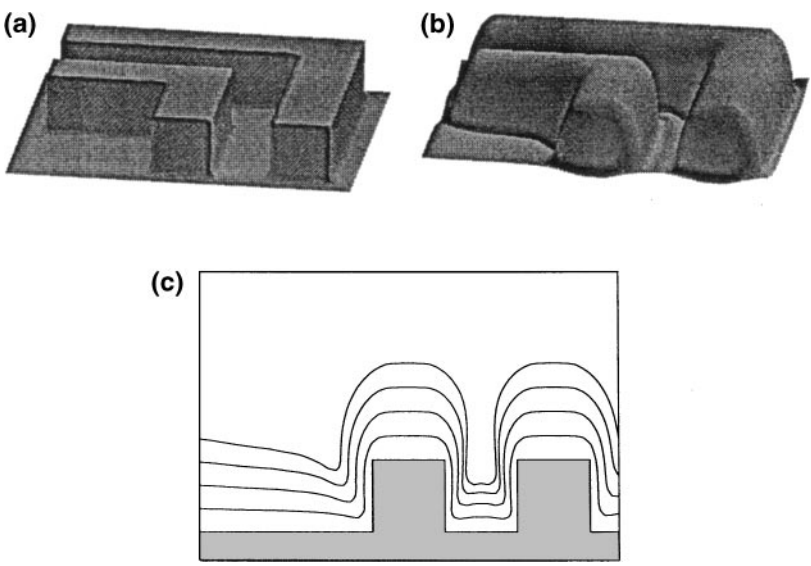

FIG. 14. Three-dimensional evolution under cosine source distribution with sticking coefficient 0.1. (a) Initial position; (b) time evolution; (c) two-dimensional cross section.

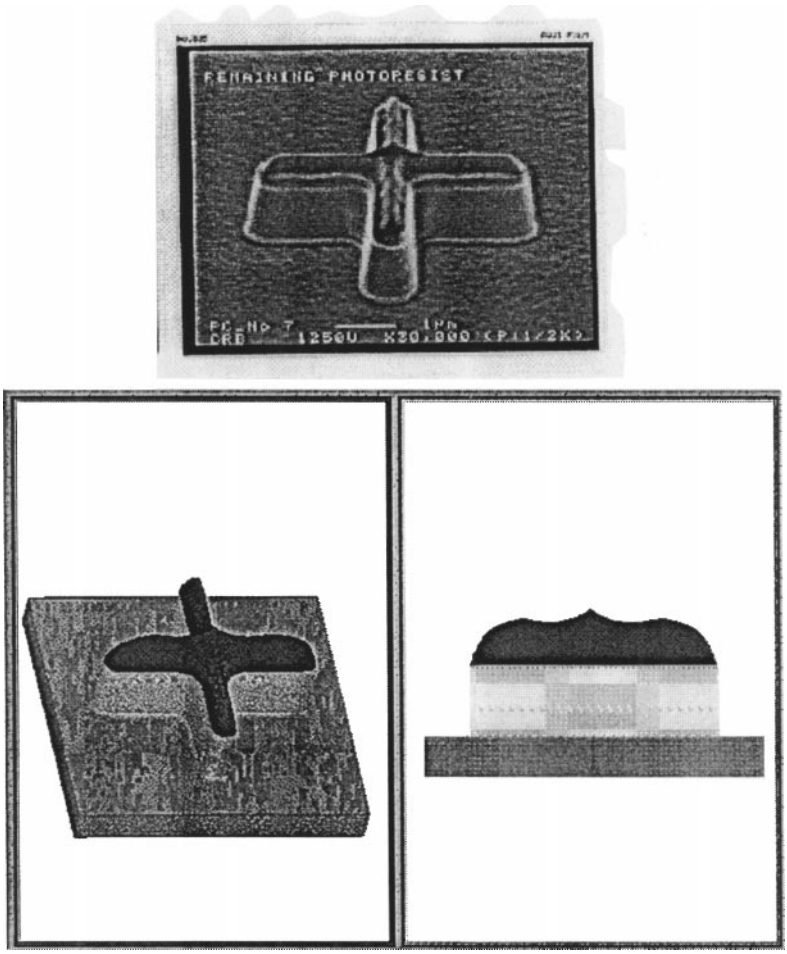

FIG. 15. Ion milling: experiment (top) vs. simulation (bottom).

of these techniques built by Technology Modeling Associates and specifically designed for process simulation; for further details about this code and its capabilities, see [94]) or a research version of the code built by SAIT of Samsung Corporation.[4]

7.6.1. Ion milling. We begin with a comparison with experiment of an ion-milling process. The goal here is verify our ability to handle critical etching angles that are not maximal in the normal direction. In these cases, as discussed earlier, the maximum of the yield function occurs away from 90 degrees, and this causes faceting in the evolving shape. Typically, the yield function is measured experimentally and then used as an empirical model fit in the numerical simulation. We show simulations from the Terrain code matched against experimental data. Figure 15 shows an experiment on the top and a simulation at the bottom. We note that both the simulation and the experiment show the crossing nonconvex curves on top of the structures, the sharp points, and the sloping sides.

7.6.2. Complex effects: measuring the effects of various terms. Next, using the SAIT code, we study of combination of various physical effects.

[4] We thank J. Huh, J. Shin, and H. Lee for use of their results.

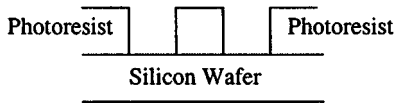

FIG. 16. Photoresist layer.

In this problem (see Fig. 16), the silicon layer is etched by a combination of uniform chemical etching and ion-assisted directional etching. Relative rates depend on process parameters: power, pressure, gas fractions, etc. The photoresist is eroded only by ion milling, and hence it is highly directional, similar to the experiment shown is Section 7.6.1.

In the first experiment, we compare relative rates of uniform and directional etching in two separate cases (see Fig. 17). In Case 1, the directionality of the etching is strong. Thus, the small total etch rate results in large amount of physial erosion of the mask. However, in Case 2, the pressure in taken as 2.5 times larger than that in Case 1, and chemical etching is chosen to be larger than in Case 1. This enchances the total etch rate, as seen in the results, produces a large amount of uniform etching, and results in considerable undercut and sidewall erosion.

In the second set of experiments, we continue with two more cases (see Fig. 18.) Here, we study the angular distribution of incident ion flux. In these studies, the gas fraction of oxygen in Case 4 is 2 times larger than that of Case 3. This results in a broader angular distribution of ion flux in Case 4, and hence the amount of sidewall erosion is bigger in Case 4 than it is in Case 3.

7.6.3. Plasma-enhanced chemical vapor deposition. Next, we show comparison with experiment of two plasma-enhanced chemical vapor deposition (PECVD) simulations using TERRAIN. We show a series of experiments. First, two smaller structure calculations are used to verify the ability to match experiment. Figures 19 shows these results. Figures 20 and 21 show more simulations for more complex structures.

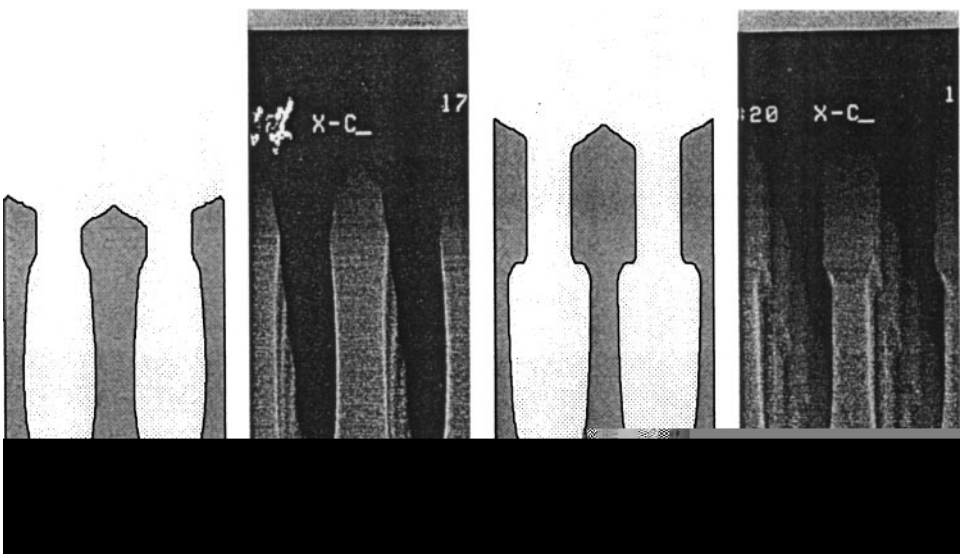

FIG. 17. Relative rates of uniform and directional etching.

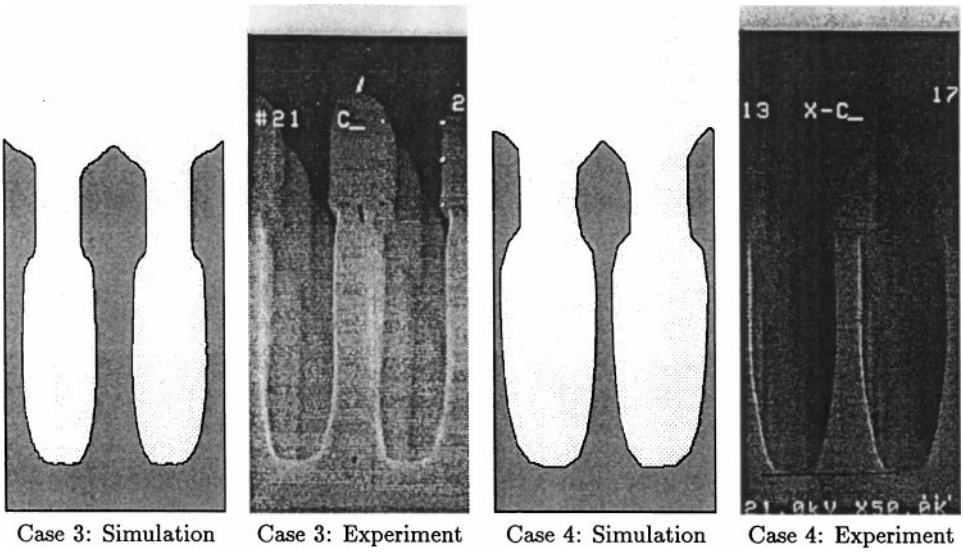

FIG. 18. Angular distribution of incident ion flux.

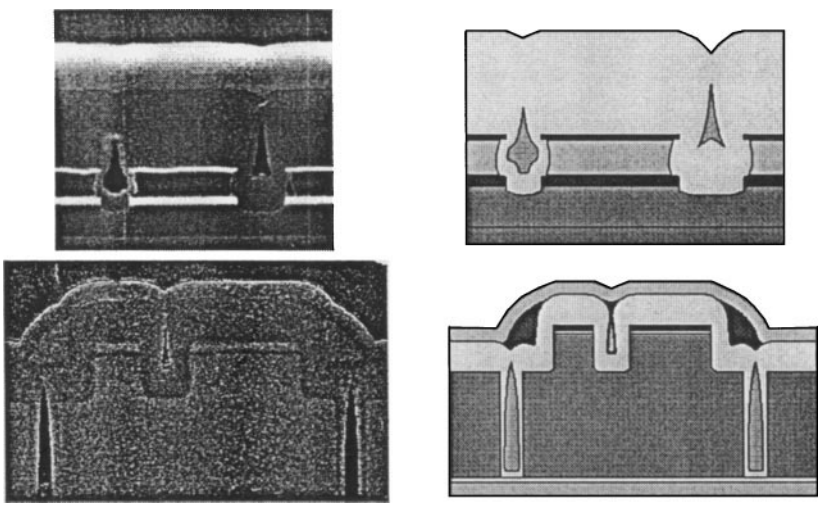

FIG. 19. PECVD on a small-scale structure: experiment (left) vs. simulation (right).

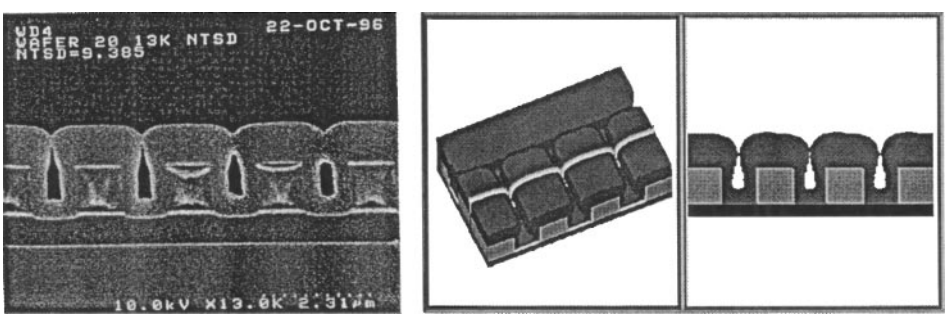

FIG. 20. PECVD: experiment (left) vs. simulation (right).

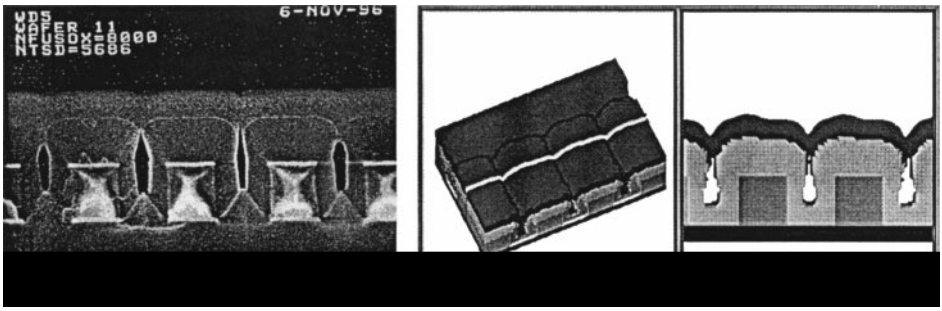

FIG. 21. PECVD: experiment (left) vs. simulation (right).

7.6.4. SRAM simulations. Finally, we show SRAM comparisons between experiment and simulations for both small structures (Fig. 22) and large structures (Fig. 23) using TERRAIN. Each figure shows the original layout together with the actual pattern printed through photolithography, followed by the sequential processing steps.

8. Seismic Traveltimes

Next, we explore applications of fast marching methods to problems involving the imaging of geophysical data sets. In [85], Sethian and Popovici used the fast marching method to rapidly construct first arrival times in seismic analysis and then coupled this work to prestack migration. Here, we follow closely that work and text. For further details, see [85].

Three-dimensional (3D) prestack migration of surface seismic data is a tool for imaging the earth's subsurface when complex geological structures and velocity fields are present, The most commonly used imaging techniques applied to 3D prestack surveys are methods based on the Kirchhoff integral, because of its flexibility in imaging irregularly sampled data and its relative computational efficiency. To perform this Kirchhoff migration, one approximately solves the wave equation with a boundary integral method. The reflectivity at every point of the earth's interior is computed by summing the recorded data on multidimensional surfaces; the shapes of the summation surfaces and the summation weights are computed from the Green's functions of the single scattering wave-propagation experiment (see [60, 66]).

8.1. *Background Equations*

In some more detail, the essence of 3D prestack migration is expressed by the integral equation

$$\text{Image}(\mathbf{x}) = \int \int_{\mathbf{x_s}} \int_{\mathbf{x_r}} G(\mathbf{x_s}, \mathbf{x}, \omega) G(\mathbf{x}, \mathbf{x_r}, \omega) \text{Data}(\mathbf{x_s}, \mathbf{x_r}, \omega)\, d\mathbf{x_r}\, d\mathbf{x_s}\, d\omega,$$

where $\mathbf{x}$ is the image output location, $\mathbf{x_s}$ and $\mathbf{x_r}$ are the data source and receiver coordinates, and ω is angular frequency. The Green's functions $G(\mathbf{x_s}, \mathbf{x}, \omega)$ and $G(\mathbf{x}, \mathbf{x_r}, \omega)$ parameterize propagation from source to image point and from image point to receiver, respectively. In most implementations, the calculation is often done instead in the time domain and can be expressed as the summation

$$\text{Image}(\mathbf{x}) = \sum_{\mathbf{x_s}} \sum_{\mathbf{x_r}} A_s A_r \text{Input}(\mathbf{x_s}, \mathbf{x_r}, t_s + t_r),$$

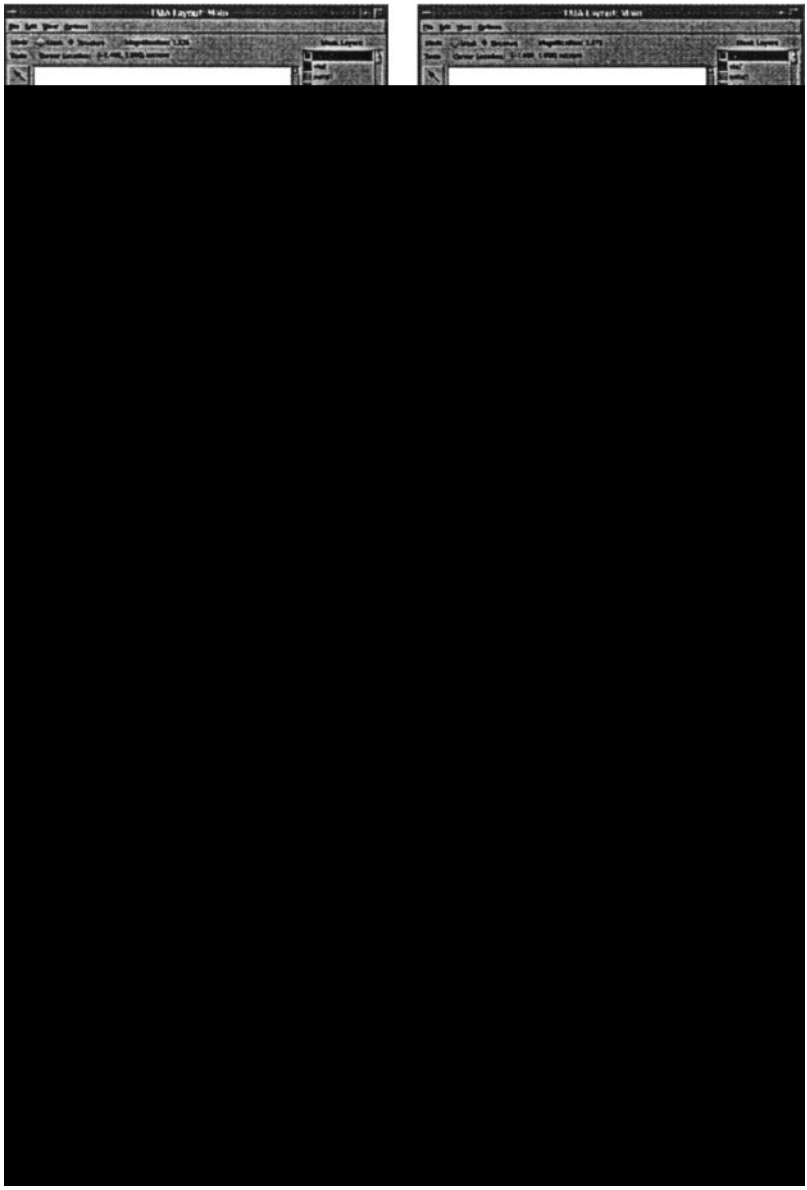

FIG. 22. SRAM simulation.

where Input is a filtered version of the input data, and the Green's functions are parameterized by the amplitude terms A_s and A_r and travel times t_s and t_r.

For 3D prestack Kirchhoff depth migration, the Green's functions are represented by five-dimensional (5D) tables; these tables are functions of the source and receiver surface locations (x, y) and of the reflector position (x, y, z) in the earth's interior. This Green's function parameterization is usually based on the assumption of acoustic propagation. This Kirchhoff prestack migration process consists of two stages. First, travel-time tables are computed and stored. Second, the migrated image is formed by convolving the prestack data with migration operator derived from the travel-time tables. Both phases present challenges

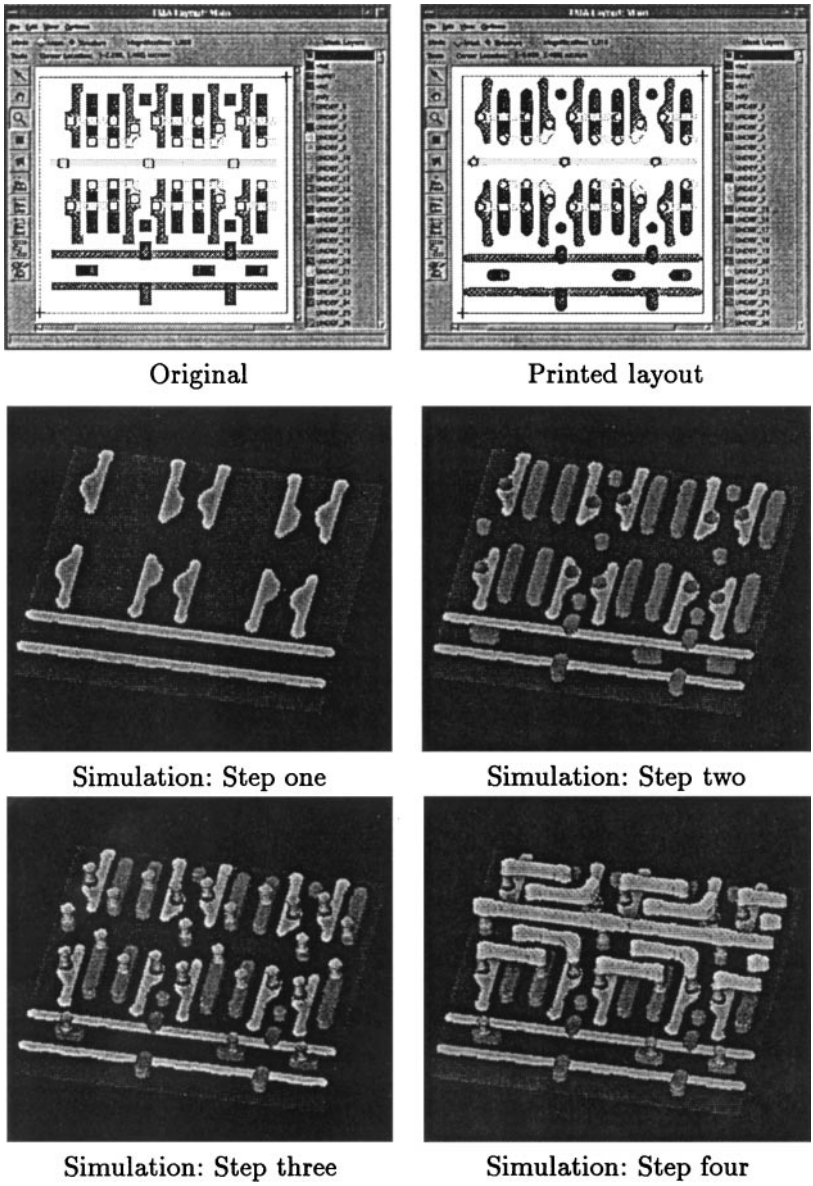

FIG. 23. SRAM simulation.

from the perspectives of the geophysical accuracy and of the computer implementation (see Fig. 24).

The key element of 3D prestack Kirchhoff depth migration is the calculation of travel-time tables used to parameterize the asymptotic Green's functions. An efficient travel-time calculation method is required to generate the 5D travel-time tables needed for 3D Kirchhoff migration.[5] Also, since depth migration problems are generally applied in areas of complex velocity structure, the travel-time calculation method must be robust. Computing 3D Green's

[5] The Green's function can be reconstructed from travel-time tables that describe travel times from all surface points (x, y) to all subsurface locations (x, y, z); thus the tables are five dimensional.

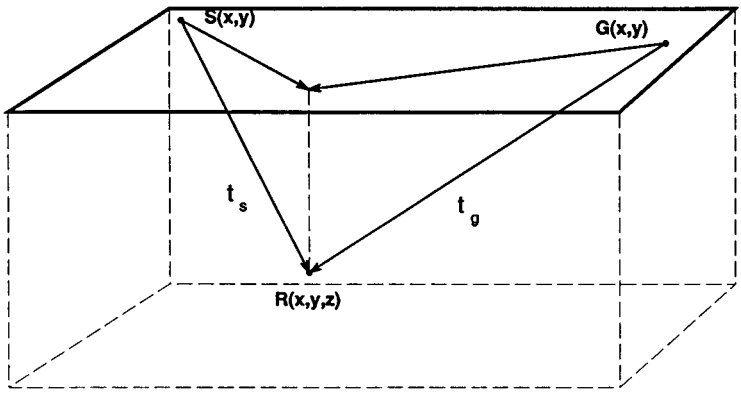

FIG. 24. In 3D seismic surveying, seismic waves are generated by surface sources (shots) S, and the reflected waves are recorded at surface receivers (geophones) G. The Green's function describes the energy of the wavefield backscattered from the reflector point at all possible source and receiver combinations.

function tables over a 100×100 square kilometer area (about 430 marine blocks), with sources positioned every 200 meters, requires 1 terabyte of travel-time volumes. Thus, speed is an important issue.

Designing efficient and accurate travel-time computation methods has a long history; the past ten years have seen considerable new advancements, particularly those aimed at a finite difference approach (see Vidale ([97])). Prior to this work, travel times were typically computed using ray tracing. While these ray-tracing methods offer a high degree of accuracy, they also pose interpolation problems in shadow areas and areas where multiple caustics develop. The use of finite difference travel times ameliorates these interpolation problems in shadow zones, at the price of foregoing detection of most energetic arrivals in exchange for the first arrival.

A broad spectrum of travel-time computation methods was developed in the early 1990s. Vidale extended his finite difference travel time to three dimensions [98], while van Trier and Symes [95] introduced a two-dimensional explicit finite difference method with a vectorizable inner kernel that ran efficiently on vector computer platforms. At its core, the problem of computing first arrival times requires solution of the Eikonal equation, with the goal of accurately and robustly dealing with the formation of cusps and corners, topological changes in the solution, and singularities. Fast marching methods, both first- and second-order versions, provide viable approaches.

8.2. *Computing Fast Marching Method Travel Times through a Salt Structure*

We begin by showing the results of using the fast marching method to compute three-dimensional travel times through a salt structure.[6] We start with the techniques applied to a 3D SEG/EAGE salt dome velocity model [8]. The salt dome model was designed to contain major complex features that are characteristic of complicated Gulf of Mexico salt structures. It includes a northwesterly plunging stock, a secondary reactivation crest southward of the stock, a low-relief eastern flank, a faulted southern flank with a toe thrust, a rounded overhang on the west flank, five sands that are gas charged (at least one contains

[6] All seismic calculations were performed using the implementation of the fast marching method developed by 3DGeo Corporation.

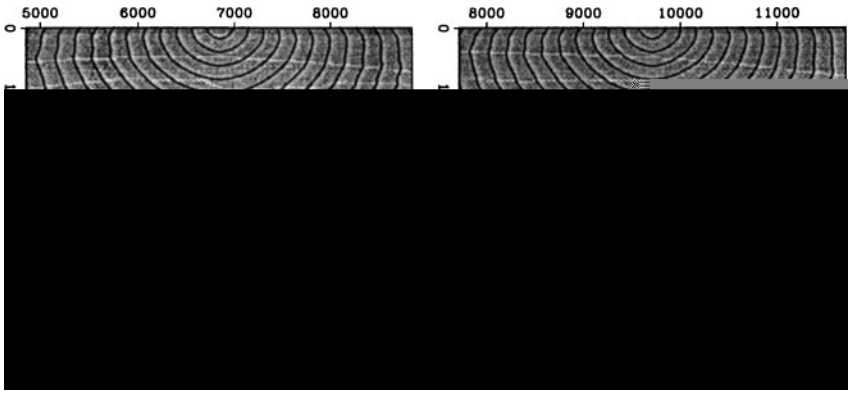

FIG. 25. Travel-time slices through SEG/EAGE salt model.

both a gas–oil contact and an oil–water contact), and a shale sheath that is modeled to be geopressured. The seafloor map exhibits a counter-regional fault scarp, a bathymetric rise associated with the sill crest and a shelf break at the southeast end of the model. The overall model size is $13.5 \times 3.5 \times 4.2$ km^3 on a 20-m grid.

The SEG/EAGE salt model has a complicated salt-to-sediments interface which creates complex wave propagation problems. We show contour travel times superimposed on the velocity model which are representative for the wave propagation patterns encountered while solving the Eikonal equation in the SEG/EAGE salt model. Figure 25 shows a travel-time slice through the SEG/EAGE salt model with a point source at the surface. The grid is a $100 \times 100 \times 100$ mesh, with mesh size equal to 40 m per cell side. Figure 25b shows the formation of headwaves which travel along the salt–sediment interface. Figure 26 shows the result of a horizontal travel-time slice through the travel-time cube at a depth of 1380 m and accurately captures the formation of cusps.

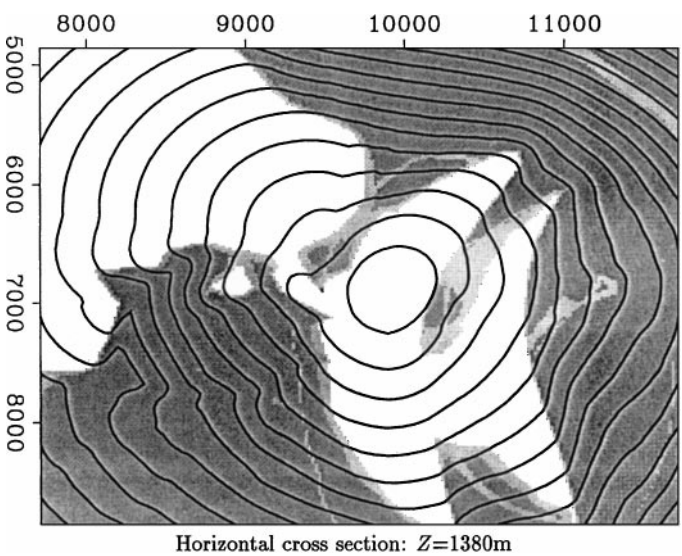

FIG. 26. Horizontal slice through SEG/EAGE salt model.

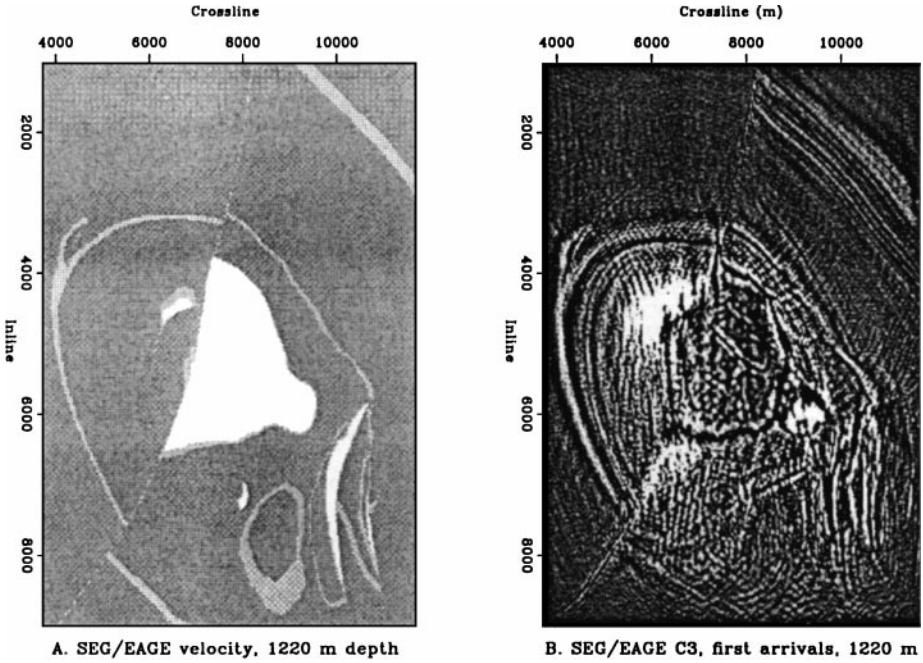

FIG. 27. Velocity model and migrated image.

8.3. *Migration Using the Fast Marching Method*

Figures 27 and 28 show slices through the three-dimensional velocity and corresponding structural images obtained from migration on prestack data obtained from a given data set. On the left, Fig. 27 shows a depth slice through a velocity cube at a depth of 1220 m; on

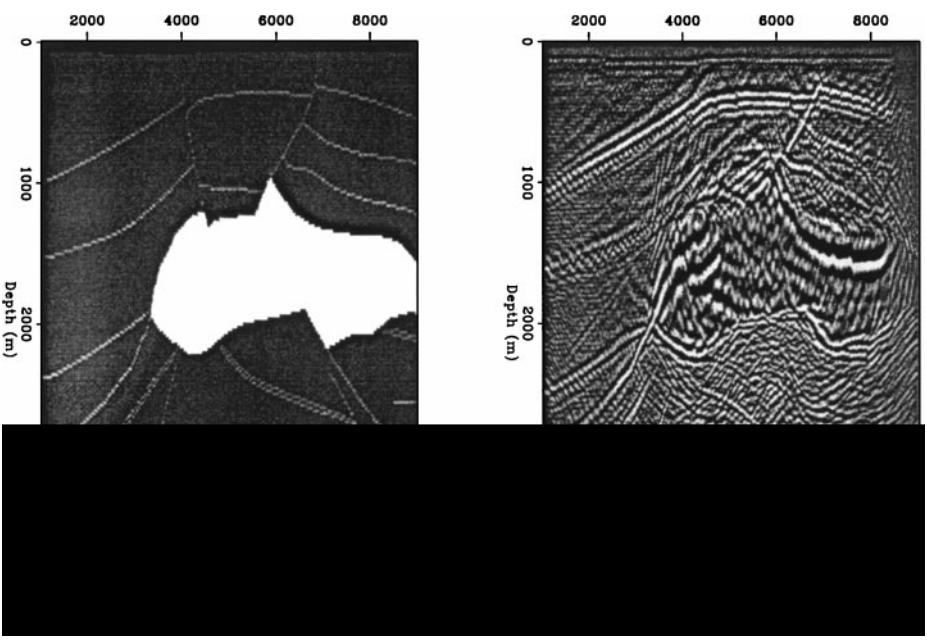

FIG. 28. Velocity model and migrated image.

the right, the corresponding migrated image slice is shown. The salt–sediment interface and the semicircular fault cutting through the salt body are imaged with high resolution. Figure 28 compares the velocity model on the left with the corresponding migrated line on the right for a different slice. The sediment images are imaged at the correct locations, together with the salt body borders. The areas with lesser quality are under the salt, most probably because of the multiple reflected arrivals at this spot from the water bottom and intra-salt reflections, and also close to the left side of the top of the salt, most probably also resulting from the use of first arrivals in the fast marching method.

Next, two-dimensional travel times were used to image the Marmousi data, which is a synthetic data set based on a real geologic model from the Cuanza basin in Angola [14]. The geologic model of the basin consists of a deltaic sediment interval deposited upon a saliferous evaporitic series. The sediments are affected by normal growth faults caused by the salt creep. Under the salt there is a folded carbonate sedimentation series forming a structural hydrocarbon trap. The challenge is to image the hydrocarbon trap. The complex velocity model, with strong lateral velocity variations, is shown in Fig. 29 on the left. On the right, the figure shows the migrated images using three-dimensional travel-time tables computed with the fast marching method, operating in a two-dimensional mode. The typical challenges in imaging this data set are (1) imaging correctly and without artifacts the position of the faults, (2) imaging the V-shaped termination of the layers, which are zones that concentrate rays and produce distorted images, (3) imaging correctly the top of the first anticlinal structure and the bottom of the two salt intrusions, and (4) imaging the sediments in the second, deep anticlinal structures. The images shown correctly image the faults, even in the high-velocity layers, and avoid artifacts at the bottom of the V-shaped fault and layer terminations. The fact that the fast marching method produces first arrivals, which may not correspond to the most energetic arrivals, may explain why the second, deeper

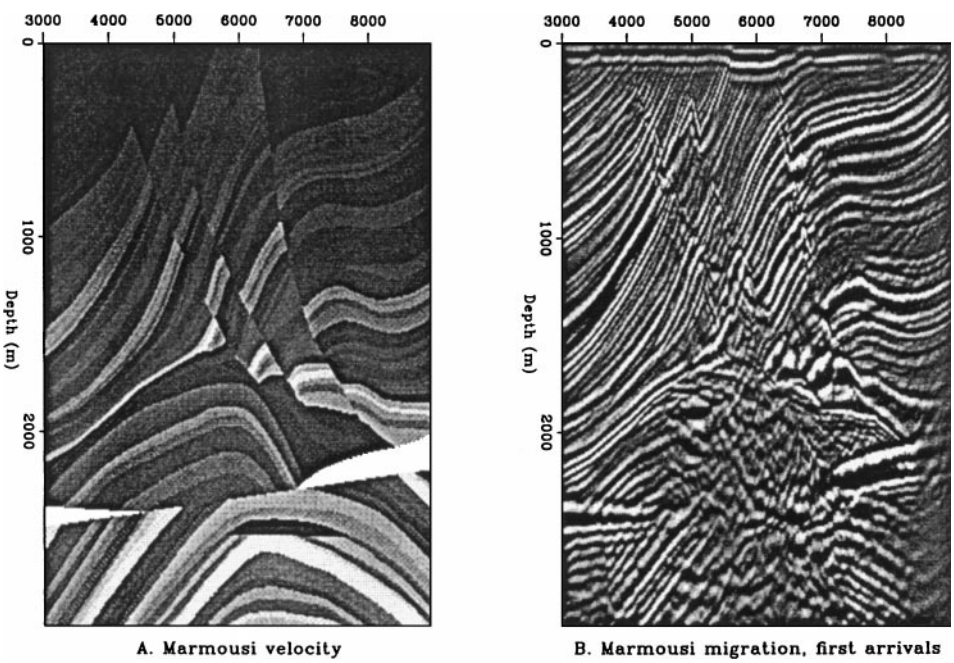

FIG. 29. Velocity model and migrated image (Marmousi data set).

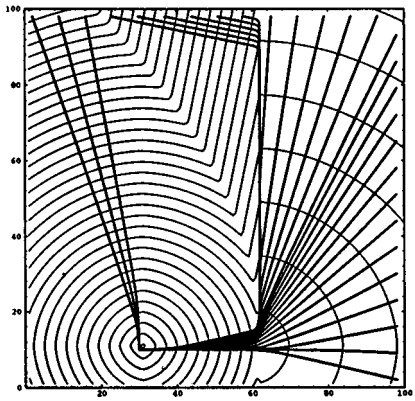

FIG. 30. Region on left is slower than region on right.

anticlinal area is missed. For further discussion of this and other features, see [85], as well as [92].

8.4. *Removing Headwaves: Toward Computing Most Energetic Arrivals*

As discussed above, one of the issues associated with using the Eikonal equation to compute arrival times is that the solution is typically limited to first arrivals. These may not be the most energetic arrivals; a noticeable case comes in the presence of sharp discontinuities. Consider a discontinuity separating a slow region from a fast region, as in Fig. 30.

A disturbance that starts on the left in the slower material will travel quickly once it reaches the discontinuity line between the two materials, and the true shortest path for points located significantly above the source may traverse this discontinuity. Such arrivals, known as headwaves, contain little energy and can negatively influence migration analysis. Recently, algorithms have been developed which remove headwaves; see, for example, the approach taken by Popovici ([92]).

Here, we introduce a variation on the fast marching method ([83]) which can be used to suppress headwaves in many situations. The key idea is to build a filter in the fast marching update procedure which represses headwave transmission; done carefully, one can remove headwaves from the calculation. This can be done without a priori information about or determination of the interface discontinuities. In Fig. 31, we show two calculations; the left

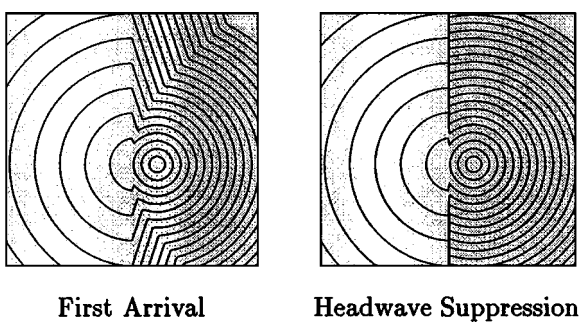

FIG. 31. Suppression of headwaves.

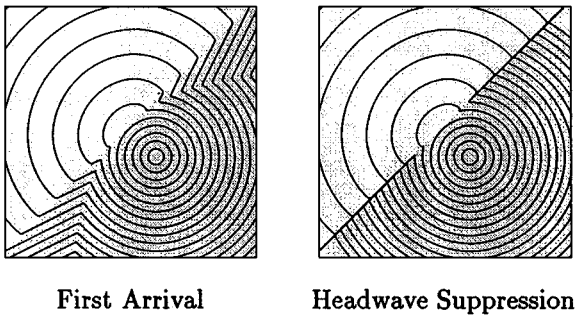

FIG. 32. Suppression of headwaves by a diagonal structure.

figure is the true first arrival, while the right figure shows suppression of headwaves. In Fig. 32, we show that the results do not depend on the orientation of the discontinuities; again, the left figure is the true first arrival, while the right figure shows suppression of headwaves. Finally, in Fig. 33, we show headwave suppression against a curved boundary interface; the interior of the circle is a faster material. For further details, see Sethian [83].

9. Optimal Structural Boundary Design

The third application of these methodologies concerns the boundary design of a loaded elastic structure. The goal is to find efficient designs which satisfy certain constraint equations while minimizing other variables, such as the total weight. These results and discussion are taken from Sethian and Wiegmann [88]; we refer the interested reader to that work for considerably more detail, explanations, and examples.

By way of illustration, consider a clamped and loaded cantilever (see Fig. 34). Suppose our goal is to remove as much material as possible from the original shape, while still making sure that the compliance (defined as the yield under the load) or the maximal stress in the structure stays below a certain threshold value. We can start with the original perforated structure and compute the stress; as illustration, the stress contours on the original design are shown in Fig. 35. We can then try to add and remove material in order to reduce the weight in such a way that the compliance or stress does not rise above a given user-prescribed level. Different designs (that is, newly introduced, removed, or reshaped holes) will give different compliance and stresses in the design. Our approach is to devise a systematic way

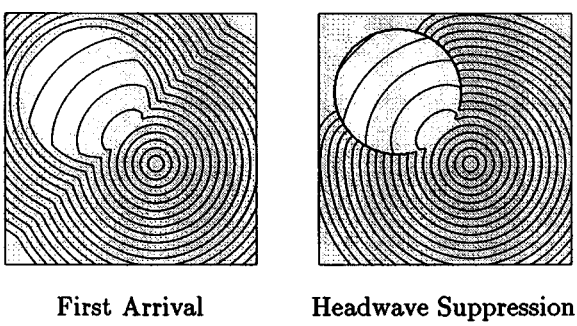

FIG. 33. Suppression of headwaves by a circular structure.

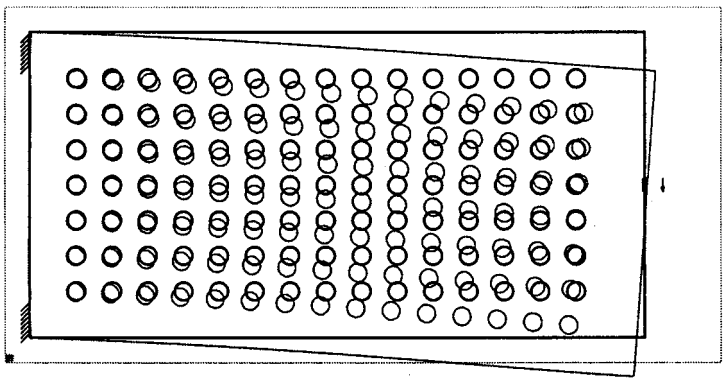

FIG. 34. Bending of the initial design of a cantilever with 105 circular holes. Parts of the left boundary are clamped; on the rest of the boundary, including all holes, the traction is specified, with nonzero loading on a small portion about the center of the right boundary. The bending is beyond the regime of small-displacement elastostatics and chosen only to illustrate the behavior. The larger rectangle is the computational domain, with a 320×160 grid indicated in the lower left corner.

to add and remove material. This requires an accurate technique to compute the stresses for a given multiply connected domain and an accurate technique to remove or reshape existing boundaries and to introduce new ones. We use the narrow-band level set method to add and subtract material, and a version of the *explicit jump immersed interface method* to compute the stress in arbitrary domains.

Our goal is to find a design configuration that minimizes the total weight while keeping the compliance below a certain prescribed value.

9.1. *Overview of Computational Approach*

As a general outline, the algorithmic approach is as follows:

• In the first step, the explicit jump immersed interface method is applied to the equations of 2D linear elastostatics in the displacement formulation. These problems on arbitrary domains are solved quickly and without mesh generation by domain embedding and the use of fast elastostatic solvers. In brief, in [88] a general technique is given for solving the linear

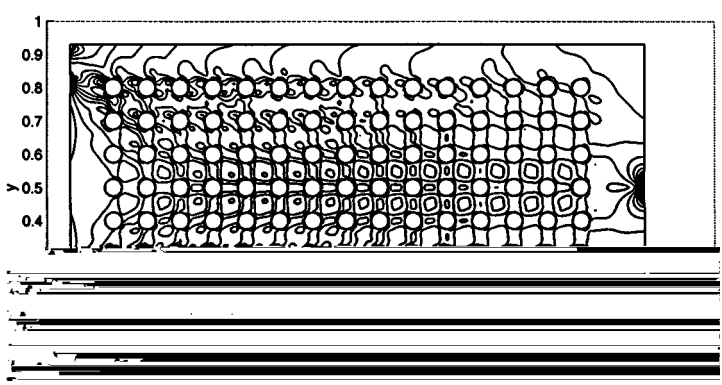

FIG. 35. Stress contours for initial configuration shown in Fig. 34.

elastostatic equations in the displacement formulation and differencing the displacements. This explicit jump immersed interface method (see [100]) is a finite difference technique on uniform grids, after LeVeque and Li's *immersed interface method* ([42]), that is capable of dealing with non-grid-aligned boundaries with the same truncation error as interior differences. The biggest benefit of this approach is that it is easy to add material (with some subgrid resolution) at hole boundaries with high stress. In particular, this approach allows one to start with designs that have holes cut "in the wrong place," and see these holes disappear.

- In the second step, the given design is modified. The narrow-band level set method is used to alter the shape, with velocities depending on the stresses in the current design. These stresses can be found from the displacements that were found in the first step. Boundary motion and merging as well as the introduction of new holes are all performed using this grid function. This approach also allows the detection of regions that have become separated from the nontrivial boundary conditions and have to be dropped from the computations. Criteria are provided for advancing the shape in an appropriate direction and to correct the evolving shape when given constraints are violated.

The goal is to solve the two-dimensional Lamé equations, where $\mathbf{u} = (u, v)$ are the displacements in x and y, respectively, and

$$\begin{aligned} -\mu(\Delta u + u_{xx} + v_{xy}) - \lambda(u_{xx} + v_{xy}) &= f^u \text{ in } \Omega, \\ -\mu(\Delta u + u_{xy} + v_{yy}) - \lambda(u_{xy} + v_{yy}) &= f^v \text{ in } \Omega. \end{aligned} \tag{24}$$

Here μ and λ are the Lamé constants, $\mathbf{f} = (f^u, f^v)$ are body forces, and Ω is an open, connected but not necessarily simply connected domain. We will also write (with $C = \mu/(\mu + \lambda)$)

$$\begin{aligned} C\Delta u + u_{xx} + v_{xy} &= -\frac{f^u}{\mu + \lambda} \text{ in } \Omega, \\ C\Delta v + u_{xy} + v_{yy} &= -\frac{f^v}{\mu + \lambda} \text{ in } \Omega. \end{aligned} \tag{25}$$

Displacement boundary conditions are

$$\mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_1 \subset \partial\Omega. \tag{26}$$

Here $\bar{\mathbf{u}} = (\bar{u}, \bar{v})^T$ are given functions on Γ_1, the part of $\partial\Omega$, the boundary of Ω, where displacements are given. Traction boundary conditions are

$$\sigma(\mathbf{u})\mathbf{n} = \mathbf{g} \text{ on } \Gamma_2 \subset \partial\Omega. \tag{27}$$

One assumes that the coefficients, geometry, and boundary values are such that the problem has a unique solution.

For concreteness, we may think of Eq. (27) as realized in Cartesian coordinates. Then $\mathbf{u} = (u, v)$ is the vector of displacements in the x and y directions, σ is the stress tensor expressed in (x, y) coordinates, $\mathbf{n} = (n_1, n_2)$ is the inward normal to the boundary (given in (x, y) coordinates), and $\mathbf{g}$ is the vector of surface forces applied at that boundary, also given in (x, y) coordinates.

We solve for the displacements and difference them to find the (symmetric) stress tensor, with Lamé constants μ and λ,

$$\sigma = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathbf{T}}) + \lambda \operatorname{trace}(\nabla \mathbf{u}) I.$$

From the stress tensor, we can then calculate the von Mises stress as

$$S = \sqrt{\sigma_{11}^2 + \sigma_{22}^2 - \sigma_{11}\sigma_{22} + 3\sigma_{12}^2}.$$

We extend the von Mises stress from the grid to the boundaries by a least-squares extrapolation method. This provides us with a velocity field F defined away from the interface with which to advance the level set function.

9.2. *Brief Technical Comments*

9.2.1. Problem setup and elliptic issues. Mesh generation issues are avoided by separating the representation of the boundary from the uniform computational grid. To keep the data structures simple and to allow use of fast elastostatic solvers on rectangular domains [99], the problem is posed on a larger, rectangular domain R with zero normal boundary conditions. The boundary conditions on the original boundary are rewritten as jump conditions that introduce discontinuities in the displacements inside R. The choice of jump and boundary conditions forces the extended solution to vanish on the extension but to match the solution inside the structure. On the level of the linear algebra, a Schur-complement (as previously used, e.g., in [43, 100]) reduces the number of variables from proportional to the grid points to proportional to the length of the boundary normalized by the mesh width. We also note that in [88], derivative estimation and corrections are carried out to third order for the purpose of achieving an $O(h^2)$ truncation error at all points, including points neighboring the boundary; in addition, the work introduces a fast elastostatic solver which is of considerable use in its own right. For details, see [88].

9.2.2. Design alteration. Once the displacement and stresses are found, this yields a velocity field which may be extended to the nearby level set grid points to advance the interface. The motion of this interface corresponds to removal material in regions of low stress and to added material in regions of high stress. The removal rate determines the closed stress contours along which new holes are cut and also the velocity of the boundary motion. It is increased only after no new holes are cut and the design boundaries have stabilized. When the constraint is violated, the removal rate is decreased to add more material in regions of high stress and remove less material in regions of low stress. The approach in [88] uses a narrow-band level set method to update the interface and the various holes, as well as an extension velocity formulation to move the neighboring level sets. We terminate when this procedure can no longer improve the weight while satisfying the compliance.

9.2.3. Algorithm flow. The algorithmic flow is as follows:

MAIN ALGORITHM

1. Initialize; find stresses in initial design.
2. While termination criteria are not satisfied do
   a. Cut new holes.
   b. Move boundaries.

c. Find displacements, stresses, etc.
d. If the constraints are violated reduce removal rate of material and revert to previous iteration.
e. Update removal rate.

Thus, the methodology presented in [88] attacks optimal design problems with hard constraints using a evolutionary approach based on calculating derivatives, displacements, and stresses associated with solving the associated relevant elliptic problems.

We shall not go any further into the derivation of the appropriate jump conditions nor the algorithmic details of the explicit immersed interface method and refer the reader to the original work in [88].

9.3. *Results*

We show two results from [88]. First, we study the constrained design of a short cantilever. A cantilever of ratio 1 : 3 is clamped everywhere on the left boundary and vertically loaded on the mid 6% of the right boundary. The rest of the right boundary and the top and bottom boundaries are traction free. The problem was chosen because it is a standard test problem for structural topology design, (for example, see [61, 101]), with a known solution for a simpler pin-jointed two-truss problem [62]. The optimal height in that case is twice the width of the structure.

In Fig. 36a, taken from [88], we show clamping, loading, and stresses in the initial design. Figure 36b shows the improved design under the combination explicit jump immersed interface method and the narrow-band level set method.

In a different calculation, Fig. 37 shows the design of a long cantilever from a perforated structure. From theoretical considerations [6, 12], a truss-like structure is expected to develop for certain optimal low-weight structures. To give the flavor of one such simulation, we show the time sequence of the truss evolution. Again, for complete details on equations of motion, numerical schemes, measurement tests, and additional examples, see the original work [88].

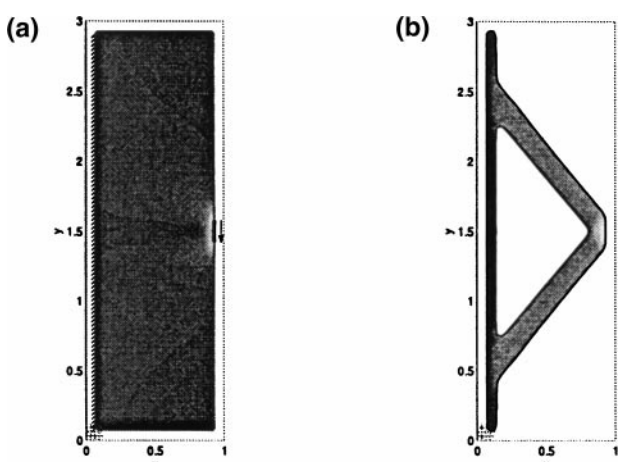

FIG. 36. Improved shape for short cantilever. (a) Stress distribution in initial design; (b) improved design.

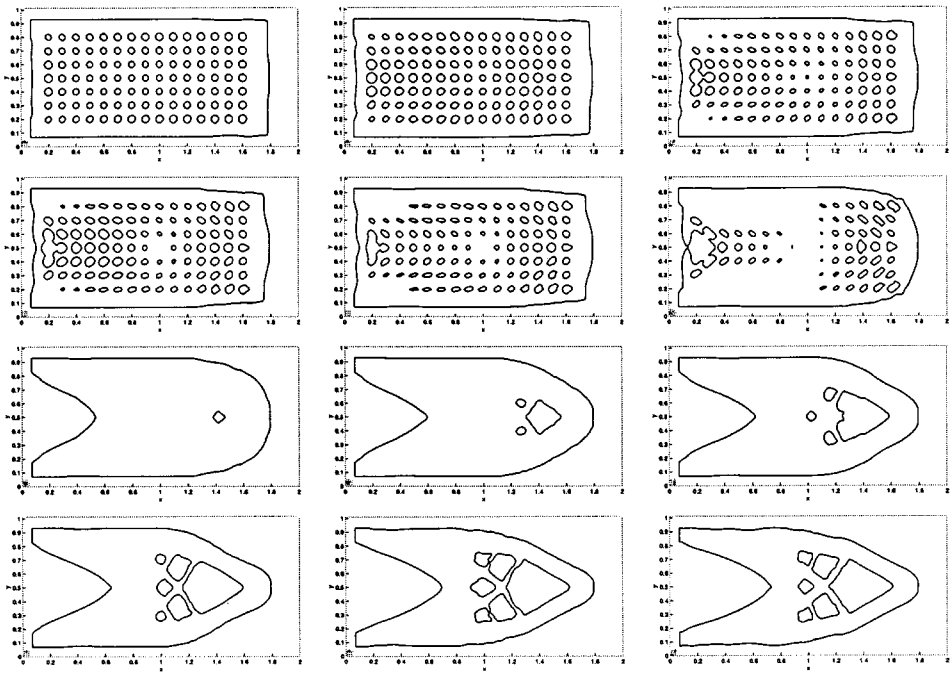

FIG. 37. Evolution of truss structure.

ACKNOWLEDGMENTS

All calculations were performed at the University of California at Berkeley and the Lawrence Berkeley Laboratory. The detailed applications of level set and fast marching methods discussed in this work are joint with D. Adalsteinsson, D. Chopp, R. Malladi, M. Popovici, and A. Wiegmann.

REFERENCES

1. D. Adalsteinsson and J. A. Sethian, A fast level set method for propagating interfaces, *J. Comput. Phys.* **118**(2), 269 (1995).
2. D. Adalsteinsson and J. A. Sethian, A unified level set approach to etching, deposition and lithography I: Algorithms and two-dimensional simulations, *J. Comput. Phys.* **120**(1), 128 (1995).
3. D. Adalsteinsson and J. A. Sethian, A unified level set approach to etching, deposition and lithography II: Three-dimensional simulations, *J. Comput. Phys.* **122**(2), 348 (1995).
4. D. Adalsteinsson and J. A. Sethian, A unified level set approach to etching, deposition and lithography III: Complex simulations and multiple effects, *J. Comput. Phys.* **138**(1), 193 (1997).
5. D. Adalsteinsson and J. A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.* **148**, 2 (1999).
6. G. Allaire and R. V. Kohn, Optimal design for minimum weight and compliance using external microstructures, *Euro. J. Mech. A/Solids* **12**, 839 (1993).
7. L. Ambrosio and H. M. Soner, Level set approach to mean curvature flow in arbitrary co-dimension, *J. Diff. Geom* **43**(4), 693 (1996).
8. F. Aminzadeh, N. Burkhard, J. Long, T. Kunz, and P. Duclos, Three dimensional SEG/EAEG models—An update, *Leading Edge* **15**, 131 (1996).
9. S. Angenent, T. Ilmanen, and D. L. Chopp, A computed example of nonuniqueness of mean curvature flow in R^3, *Comm. Partial Diff. Eqn.* **20**(11-1), 1937 (1995).

10. T. Aslam, J. Bzdil, and D. Stewart, Level set methods applied to modeling detonation shock dynamics, *J. Comput. Phys.* **126**, 390 (1996).
11. T. J. Barth and J. A. Sethian, Numerical schemes for the Hamilton–Jacobi and level set equations on triangulated domains, *J. Comput. Phys.* **145**(1), 1 (1998).
12. M. P. Bendsøe and R. Haber, The Michell layout problem as a low volume fraction limit of the homogenization method for topology design: An asymptotic study, *J. Struct. Optimization* **6**, 263 (1993).
13. A. Bourlioux, A coupled level-set volume of fluid algorithm for tracking material interfaces, presented at the Sixth International Symposium on Computational Fluid Dynamics, Sept. 4–8, 1995, Lake Tahoe, NV.
14. A. Bourgeois, M. Bourget, P. Lailly, M. Poulet, P. Ricarte, and R. Versteeg, Marmousi, model and data, in *Proceedings of the 1990 EAEG Workshop on Practical Aspects of Seismic Data Inversion, 1991*.
15. J. U. Brackbill, D. B. Kothe, and C. Zemach, A continuum method for modeling surface tension, *J. Comput. Phys.* **100**, 335 (1992).
16. M. R. Brewer, R. Malladi, G. Pankiewicz, B. Conway, and L. Tarassenko, Methods for large-scale segmentation of cloud images, presented at 1997 EUMETSAT Meteorological Satellite Data Users' Conference, Brussels, 1997.
17. B. Bunner and G. Tryggvason, Direct numerical simulations of three-dimensional bubbly flows, *Phys. Fluids* **11**(8), 1967 (1999).
18. V. Caselles, F. Catte, T. Coll, and F. Dibos, A geometric model for active contours in image processing, *Numer. Math.* **66**, 1 (1993).
19. Y. C. Chang, T. Y. Hou, B. Merriman, and S. J. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, *J. Comput. Phys.* **124**, 449 (1996).
20. S. Chen, B. Merriman, S. Osher, and P. Smereka, A simple level set method for solving Stefan problems, *J. Comput. Phys.* **138**, 8 (1997).
21. Y. Chen, Y. Bi, and T. Jiang, The liquid bridge with Marangoni effect, *Commun. Nonlinear Sci. Numer. Simu.* **1**(1), 48 (1996).
22. D. L. Chopp, Computing minimal surfaces via level set curvature flow, *J. Comput. Phys.* **106**, 77 (1993).
23. D. L. Chopp, Numerical computation of self-similar solutions for mean curvature flow, *J. Exp. Math.* **3**(1), 1 (1994).
24. D. L. Chopp and J. A. Sethian, Flow under curvature: Singularity formation, minimal surfaces, and geodesics, *J. Exp. Math.* **2**(4), 235 (1993).
25. D. L. Chopp and J. A. Sethian, Motion by intrinsic Laplacian of curvature, *Interfaces Free Bound.* **1**, 107 (1999).
26. D. L. Chopp, A. Tongen, and J. A. Sethian, Fast approximations of surface diffusion, ESAM, Northwestern University, 2000, preprint.
27. M. G. Crandall and P. L. Lions, Two approximations of solutions of Hamilton–Jacobi equations, *Math. Comp.* **167**(43), 1 (1984).
28. M. G. Crandall and P.-L. Lions, Viscosity solutions of Hamilton–Jacobi equations, *Trans. Am. Math. Sci.* **277**, 1 (1983).
29. E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1**, 269 (1959).
30. A. Esmaeeli and G. Tryggvason, Direct numerical simulations of bubbly flows. I. Low Reynolds number arrays, *J. Fluid Mech.* **377**, 313 (1998).
31. A. Esmaeeli and G. Tryggvason, Direct numerical simulations of bubbly flows. II. Moderate Reynolds number arrays, *J. Fluid Mech.* **385**, 325 (1999).
32. R. P. Fedkiw, T. Aslam, and Xu Shaojie, The ghost fluid method for deflagration and detonation discontinuities, *J. Comput. Phys.* **154**(2), 393 (1999).
33. J. Glimm, D. Saltz, and D. H. Sharp, Two-phase modelling of a fluid mixing layer, *J. Fluid Mech.* **378**, 119 (1999).
34. J. Glimm, J. W. Grove, X. L. Li, and K. M. Shyue, Three-dimensional front tracking, *SIAM J. Sci. Comp.* **19**(3), 703 (1998).

35. J. Helmsen, E. G. Puckett, P. Colella, and M. Dorr, Two new methods for simulating photolithography development, in *SPIE 1996. International Symposium on Microlithography* (SPIE, Bellingham, WA, June 1996), Vol. 2726.
36. C. W. Hirt and B. D. Nicholls, Volume of Fluid (VOF) method for dynamics of free boundaries, *J. Comput. Phys.* **39**, 201 (1981).
37. E. Holm and H. Langtangen, A method for simulating sharp fluid interfaces in groundwater flow, preprint, 1998.
38. R. Kimmel, A. Amir, and A. M. Bruckstein, Finding shortest paths on surfaces using level sets propagation, *IEEE Trans. Pattern Anal. Machine Intell.* **17**(6), 635 (1995).
39. R. Kimmel and J. A. Sethian, *Fast Marching Methods for Robotic Navigation with Constraints*, Center for Pure and Applied Mathematics Report (University of California, Berkeley, May 1996).
40. R. Kimmel and J. A. Sethian, Fast marching methods on triangulated domains, *Proc. Natl. Acad. Sci.* **95**, 8341 (1998).
41. R. Kimmel and J. A. Sethian, Fast Voronoi diagrams and offsets on triangulated surfaces, in *Proceedings, AFA Conference on Curves and Surfaces: Curve and Surface Design*, edited by P. Laurent, P. Sablonniere, and L. Schumaker (Vanderbilt Press, 2001), to appear.
42. R. J. LeVeque and Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal.* **13**, 1019 (1994).
43. Z. Li, A fast iterative algorithm for elliptic interface problems. *SIAM J. Numer. Anal.* **35**(1), 230 (1998).
44. X. L. Li, Study of three-dimensional Rayleigh–Taylor instability in compressible fluids through level set method and parallel computation, *Phys. Fluids A* **5**(1), 1904 (1993).
45. W. E. Lorenson and H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Comput. Graphics* **21**, 4 (1987).
46. R. Malladi and J. A. Sethian, *A Unified Approach for Shape Segmentation, Representation, and Recognition*, Center for Pure and Applied Mathematics, Report 614 (University of California, Berkeley, 1994).
47. R. Malladi and J. A. Sethian, Image processing via level set curvature flow, *Proc. Natl. Acad. Sci.* **92**(15), 7046 (1995).
48. R. Malladi and J. A. Sethian, A unified approach to noise removal, image enhancement, and shape recovery, *IEEE Trans. Image Process.* **5**(11), 1554 (1996).
49. R. Malladi and J. A. Sethian, An $O(N \log N)$ algorithm for shape modeling, *Proc. Natl. Acad. Sci.* **93**, 9389 (1996).
50. R. Malladi, J. A. Sethian, and B. C. Vemuri, Evolutionary fronts for topology-independent shape modeling and recovery, in *Proceedings of Third European Conference on Computer Vision, Stockholm, Sweden*, Lecture Notes in Computer Science (Springer-Verlag, Berlin/New York, 1994), Vol. 800, pp. 3–13.
51. R. Malladi, J. A. Sethian, and B. C. Vemuri, Shape modeling with front propagation: A level set approach, *IEEE Trans. Pattern Anal. Machine Intell.* **17**(2), 158 (1995).
52. B. Merriman, J. Bence, and S. J. Osher, Motion of multiple junctions: A level set approach, *J. Comput. Phys.* **112**(2), 334 (1994).
53. B. Milne, *Adaptive Level Set Methods Interfaces*, Ph.D. thesis (Department of Mathematics, University of California, Berkeley, 1995).
54. W. Mulder, S. J. Osher, and J. A. Sethian, Computing interface motion in compressible gas dynamics, *J. Comput. Phys.* **100**, 209 (1992).
55. W. Noh and P. Woodward, A simple line interface calculation, in *Proceedings, Fifth International Conference on Fluid Dynamics*, edited by A. I. van de Vooran and P. J. Zandberger (Springer-Verlag, Berlin/New York, 1976).
56. S. Osher and J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
57. S. Osher and C. Shu, High-order nonoscillatory schemes for Hamilton–Jacobi equations, *J. Comput. Phys.* **28**, 907 (1991).
58. S. Popinet and S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, *Int. J. Numer. Meth. Fluids* **30**(6), 775 (1999).

59. E. G. Puckett, A volume-of-fluid interface tracking algorithm with applications to computing shock wave refraction, in *Proceedings of the 4th International Symposium on Computational Computational Fluid Dynamics, Davis, California, 1991*.
60. W. E. A. Rietveld and A. J. Berkhout, Prestack depth migration by means of controlled illumination, *Geophysics* **59**(5), 801 (1994).
61. D. Reynolds, J. McConnachie, P. Bettess, W. C. Christie, and J. W. Bull, Reverse adaptivity—a new evolutionary tool for structural optimization, *Int. J. Numer. Meth. Eng.* **45**, 529 (1999).
62. G. I. N. Rozvany, M. P. Bendsøe, and U. Kirsch, Layout optimisation of structures, *Appl. Mech. Rev.* **48**(2), 41 (1995).
63. C. Rhee, L. Talbot, and J. A. Sethian, Dynamical study of a premixed V flame, *J. Fluid Mech.* **300**, 87 (1995).
64. E. Rouy and A. Tourin, A viscosity solutions approach to Shape–From–Shading, *SIAM J. Numer. Anal.* **29**(3), 867 (1992).
65. F. Santosa, A level set approach for inverse problems involving obstacles, *ESIAM Control Optim. Calculus Var.* **1**, 17 (1996).
66. W. A. Schneider, Jr., Robust and efficient upwind finite-difference traveltime calculations in three dimensions, *Geophysics* **60**, 1108 (1995).
67. A. Sarti, C. Ortiz, S. Lockett, and R. Malladi, *A Unified Geometric Model for 3D Confocal Image Analysis in Cytology*, LBL Report (Lawrence Berkeley National Laboratory, University of California, May, 1998), in *Geometric Methods in Biomedical Image Processing*, edited by R. Mallad (Springer-Verlag, Berlin/New York, 2001).
68. J. A. Sethian, *An Analysis of Flame Propagation* Ph.D. dissertation (Department of Mathematics, University of California, Berkeley, 1982).
69. J. A. Sethian, Curvature and the evolution of fronts, *Commun. Math. Phys.* **101**, 487 (1985).
70. J. A. Sethian, Numerical methods for propagating fronts, in *Variational Methods for Free Surface Interfaces*, edited by P. Concus and R. Finn (Springer-Verlag, New York, 1987).
71. J. A. Sethian, *Parallel Level Set Methods for Propagating Interfaces on the Connection Machine*, TMC Technical Report (1989).
72. J. A. Sethian, Numerical algorithms for propagating interfaces: Hamilton–Jacobi equations and conservation laws, *J. Diff. Geom.* **31**, 131 (1990).
73. J. A. Sethian, Curvature flow and entropy conditions applied to grid generation, *J. Comput. Phys.* **115**, 440 (1994).
74. J. A. Sethian, Algorithms for tracking interfaces in CFD and material science, *Annu. Rev. Comput. Fluid Mech.* 1995.
75. J. A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Natl. Acad. Sci.* **93**(4), 1591 (1996).
76. J. A. Sethian, Fast marching level set methods for three-dimensional photolithography development, in *Proceedings, SPIE 1996 International Symposium on Microlithography* (SPIE, Bellingham, WA, June 1996).
77. J. A. Sethian, *A Review of the Theory, Algorithms, and Applications of Level Set Methods for Propagating Interfaces* (Acta Numerica, Cambridge University Press, Cambridge, UK 1996).
78. J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences* (Cambridge University Press, Cambridge, UK, 1996).
79. J. A. Sethian, Tracking interfaces with level sets, *Am. Sci.* May–June, 254 (1997).
80. J. A. Sethian, *Fast Marching Methods and Level Set Methods for Propagating Interfaces*, von Karman Institute Lecture Series, Computational Fluid Mechanics (1998).
81. J. A. Sethian, *Level Set Methods and Fast Marching Methods* (Cambridge University Press, Cambridge, UK, 1999).
82. J. A. Sethian, *Fast Marching Methods, SIAM Rev.* **41**, 199, July (1999).
83. J. A. Sethian, Prediction of energetic arrivals, in preparation.
84. J. A. Sethian and D. Adalsteinsson, An overview of level set methods for etching, deposition, and lithography development, *IEEE Trans. Semicond. Dev. 1996* **10**(1), 167 (1997).

85. J. A. Sethian and M. Popovici, Fast marching methods applied to computation of seismic travel times, *Geophysics* **64**, 2 (1999).
86. J. A. Sethian and J. D. Strain, Crystal growth and dendritic solidification *J. Comput. Phys.* **98**, 231 (1992).
87. J. A. Sethian and A. Vladimirsky, Extensions to triangulated fast marching Methods, *Proc. Natl. Acad. Sci.* **97**, 5699 (2000).
88. J. A. Sethian and A. Wiegmann, Structural and boundary design via level set and immersed interface methods, *J. Comput. Phys.* **163**, 489 (2000), doi:10.1006/jcph.2000.6581.
89. G. Son and V. K. Dhir, Numerical simulation of film boiling near critical pressures with a level set method, *J. Heat Transfer* **120**, 183 (1998).
90. M. Sussman and P. Smereka, *Axisymmetric Free Boundary Problems*, preprint (1998).
91. M. Sussman, P. Smereka, and S. J. Osher, A level set method for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
92. 3DGeo Corporation, *Computing and Imaging Using Fast Marching Methods*, 3DGeo Corporation, Internal Report (June 1998).
93. Technology Modeling Associates, *Three-Dimensional Photolithography Simulation with Depict 4.0*. Technology Modeling Associates, Internal Documentation (January 1996).
94. Terrain, *Topography Simulation for IC Technology*; Reference Manual (Avant! Corporation, Fremont, CA, 1998).
95. J. van Trier and W. W. Symes, Upwind finite-difference calculations of traveltimes, *Geophysics* **56**(6), 812 (1991).
96. J. N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, *IEEE Trans. Autom. Control* **40**, 1528 (1995).
97. J. Vidale, Finite-difference calculation of travel times, *Bull. Seismol. Soc. Am.* **78**(6), 2062 (1988).
98. J. Vidale, Finite-difference calculation of traveltimes in three dimensions, *Geophysics* **55**, 521 (1990).
99. A. Wiegmann, *Fast Poisson, Fast Helmholtz and Fast Linear Elastostatic Solvers on Rectangular* Parallelepipeds, Technical Report LBNL-43565 (Lawrence Berkeley National Laboratory, MS 50A-1148, June 1999), submitted for publication.
100. A. Wiegmann and K. P. Bube, The explicit-jump immersed interface method: Finite difference methods for PDE with piecewise smooth solutions, *SIAM J. Numer. Anal.*, in press.
101. Y. M. Xie and G. P. Steven, *Evolutionary Structural Optimization* (Springer-Verlag, Berlin/New York, 1997).
102. H. Zhang, L. L. Zheng, V. Prasad, and T. Hou, A curvilinear level set formulation for highly deformable free surface problems with application to solidification, *Numer. Heat Transfer B* **34**, 1 (1997).
103. H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, A variational level set approach to multiphase motion, *J. Comput. Phys.* **127**, 179 (1996).
104. H.-K. Zhao, B. Merriman, S. Osher, and L. Wang, Capturing the behaviour of bubbles and drops using the variational level set approach, *J. Comput. Phys.* **143**(2), 495 (1998).
105. J. Zhu and P. D. Ronney, Simulation of front propagation at large non-dimensional flow disturbance intensities, *Combust. Sci. Technol.* **100**, 183 (1995).
106. J. Zhu and J. A. Sethian, Projection methods coupled to level set interface techniques, *J. Comput. Phys.* **102**, 128 (1992).