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

J.A. Sethian *

Dept. of Mathematics
University of California, Berkeley 94720

Feb. 20, 2000

Abstract

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 two aspects of materials sciences: semiconductor process simulations 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 can 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.
- Subtle feedback between the physics and chemistry off the interface and the position/motion of the front itself.

One approach to formulating, modeling, and building computational techniques for some aspects of these problems is provided by level set methods, introduced by Osher and Sethian [52]. These techniques work by embedding the propagating interface as the zero level set of a time-dependent,

*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 under grant FDN00014-96-1-0381.

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, medical imaging, A general overview of the theory, numerical approximation and range of applications may be found in [75].

Level set methods rely in part on the theory of curve and surface evolution given in [63] and on the link between front propagation and hyperbolic conservation laws discussed in [64]. They recast interface motion as a time-dependent Eulerian initial value partial differential equation, and 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 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 can be developed. 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 numerical techniques, known as Fast Marching Methods [69], have been developed for solving 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 [26], 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 Narrow Band level set method introduced by Adalsteinsson and Sethian in [1]. In the case of Fast Marching methods, 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 two in particular. First, we discuss interface propagation techniques for process simulation in semiconductor manufacturing, and focus 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 re-emission and re-deposition processes. Here, we follow closely the work of Adalsteinsson and Sethian, [2, 3, 4]. Second, we discuss the application of level set techniques to optimal structural topology design. Here, the goal is to design materials which can carry given loads and minimize the amount of material involved.

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 $\vec{u} = (u, v)$ 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 [75]) the equations of motion in terms of individual components $\vec{x} = (x, y)$ as

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

  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.

- **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

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

  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

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

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. It requires special attention when marker particles collide; these 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 straightforwardly applies in multiple dimensions, and handles 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 $\nabla\chi$ in order 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

In order 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 provided by the work on the evolution of curves and surfaces and the link between hyperbolic conservation laws and propagation equations, see Sethian [62, 63, 64]; leading up to the introduction of level set methods by Osher and Sethian in [52].

- Second, the Osher-Sethian level set technique which discretizes the above requires an additional space dimension to carry the embedding, and hence is computationally inefficient for many problems. This is rectified through 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, 16, 17, 84, 47, 58, 79, 91]. 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 particles 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 [14], Esmaeeli and Tryggvason [27, 28], and Glimm et al. [30, 31]). Volume-of-fluid techniques, starting with the

initial work of Noh and Woodward [51] (see also [33]), have been used to handle shock interactions and fluid interfaces (see, for example, Puckett [55] and Popinet and Zaleski [54]). Level set techniques have been applied to a large collection of problems; general reviews may be found in [71, 72, 75, 74]; a popular review may be found in [73] and an introductory web page may be found at www.math.berkeley.edu/∼sethian/level_set.html. 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 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

In order to build up to the numerical implementation of the level set method introduced in [52], 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 non-differentiability is intimately connected to the notion of appropriate weak solutions. The goal is to construct numerical techniques which naturally account for this non-differentiability in the construction of accurate and efficient approximation schemes, and admit physically correct non-smooth solutions.

In [62, 63], 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 Figure 1. In Figure 1(a), 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 (Eqns. 1) given by the geometric view. This a perfectly reasonable view of the solution, however 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 an 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 1(b) shows this alternate weak solution. Roughly speaking, one wants to remove the "tail" from the "swallowtail" (see [63].) 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 and the envelope of these wave fronts always corresponds to the "first arrivals". This will automatically produce the solution given on the right in Figure 1. This is the approach taken in [63].

Another way to obtain the solution is through the notion of an entropy condition proposed in [62, 63]; 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' construction given in Figure 1.

3.2 Curvature-Driven Limits and Viscous Hyperbolic Conservation Laws

Yet another way of obtaining this non-differentiable weak solution after the occurrence of the singularity is through the limit of curvature-driven flows. This is what was done in [63, 64], and clearly

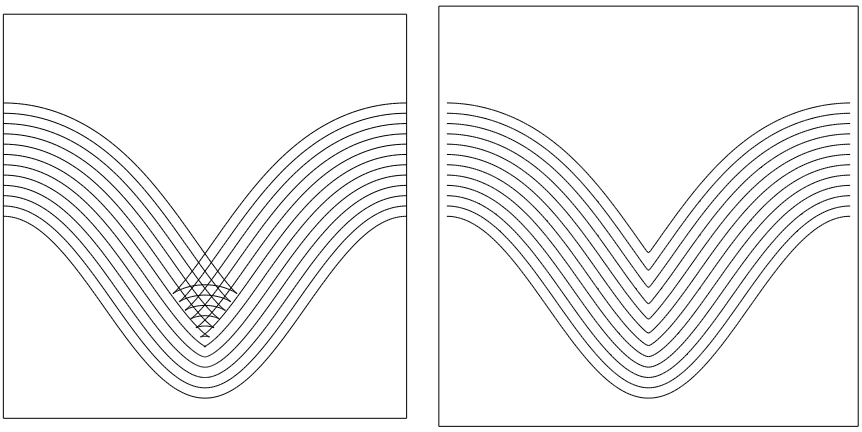


Fig. 1(a) Swallowtail ($F = 1.0$) Fig. 1(b) Entropy solution ($F = 1.0$)

Figure 1: Cosine curve propagating with unit speed.

indicates the link with hyperbolic conservation laws.

Following those discussions, consider now a speed function of the form $F = 1 - \epsilon\kappa$, 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 [63], a curvature evolution equation can be written 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 C^∞, as shown in Figure 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 Figure 2(b).

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 [64], 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

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

and thus the equation of motion becomes

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

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

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

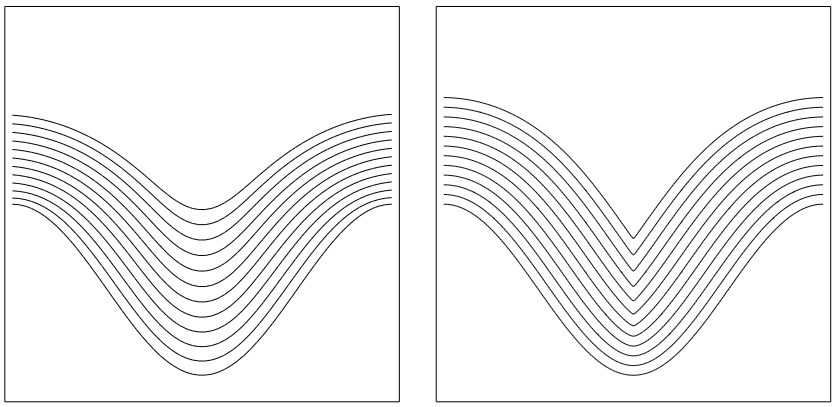


Fig. 2(a) $F = 1 - 0.25\kappa$ Fig. 2(b) Entropy solution ($F = 1.0$)

Figure 2: Entropy solution is the limit of viscous solutions.

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,

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

Thus, as shown in [64], 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.

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 accurately track intricate development. Thus, the strategy laid out in [64] was to transfer this technology to front propagation problems, and led up to the level set method introduced in [52].

II. Basic Algorithms for Interface Advancement

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

The above discussion focussed on curves which remain graphs. The numerical Osher-Sethian "level set method" 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 multi-dimensional 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; first the embedding of the interface as the zero level set of a higher dimensional function, and second, 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, $\Gamma(t=0) : [0,\infty) \to R^N$, 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. If this propagating interface is embedded as the zero level set of a higher dimensional function ϕ, that is, let $\phi(x, t=0)$, where $x \in R^N$ is defined by

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

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 [52]. As discussed in [62, 63, 64], 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 $\phi(x, y, t) = 0$ 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,

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

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 [52]. One of the simplest first order scheme is given as

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

where

$$\nabla^{+} = \left[\begin{array}{c} \max(D_{ijk}^{-x}, 0)^2 + \min(D_{ijk}^{+x}, 0)^2 + \\ \max(D_{ijk}^{-y}, 0)^2 + \min(D_{ijk}^{+y}, 0)^2 + \\ \max(D_{ijk}^{-z}, 0)^2 + \min(D_{ijk}^{+z}, 0)^2 \end{array} \right]^{1/2}$$

$$\nabla^{-} = \left[\begin{array}{c} \max(D_{ijk}^{+x}, 0)^2 + \min(D_{ijk}^{-x}, 0)^2 + \\ \max(D_{ijk}^{+y}, 0)^2 + \min(D_{ijk}^{-y}, 0)^2 + \\ \max(D_{ijk}^{+z}, 0)^2 + \min(D_{ijk}^{-z}, 0)^2 \end{array} \right]^{1/2}$$

Higher order schemes are available, see [52].

The above formulation reveals two central embeddings.

1. First, in the initialization step (Eqn. 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 "re–initialization".

2. Second, the construction of the initial value PDE given in Eqn. 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

$$\phi_t + F^{ext}|\nabla\phi| = 0, \tag{13}$$

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

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

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

Both of these issues need to be confronted in order 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 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 non-linear 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 [19], and used in recovering shapes from images in Malladi, Sethian and Vemuri [46]. The idea is straightforward, and can be best understood by means of figures.

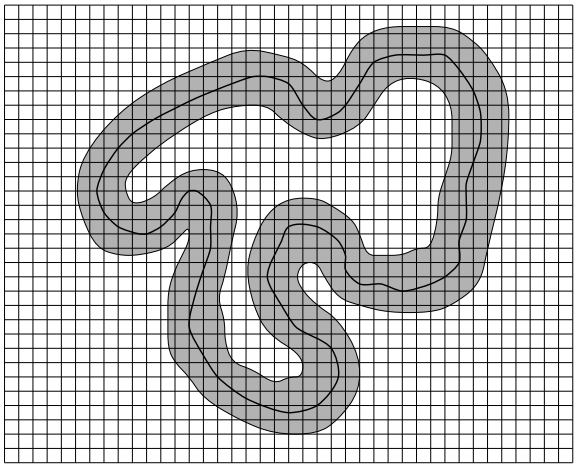


Figure 3: Grid points in dark area are members of narrow band.

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 this array (dark grid points in Figure 4 are located in a narrow band around the front of a user-defined width) (see Figure 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 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 "re-initialization".

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.

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 multi-dimensions. 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 speed-up is substantial; 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].

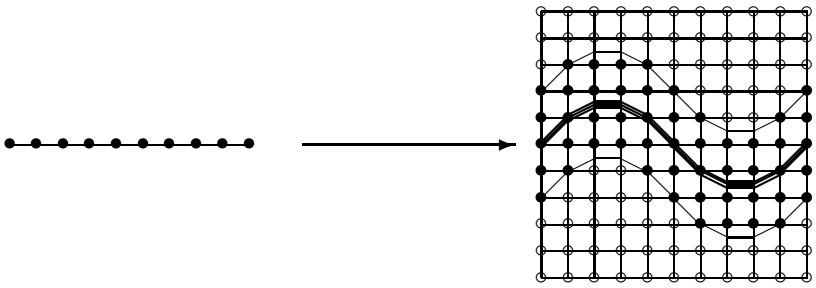


Figure 4: Pointer array tags interior and boundary band points

4.4 Constructing Extension Velocities

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

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

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

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

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

1. **No natural speed function:** In some physical problems, the velocity is given only at the front itself. For example, semi-conductor 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, 3, 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. **Sub-grid resolution:** 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:** In some problems, the speed of the interface needs to 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:** 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. The reason to maintain the signed distance function is that by keeping a uniform separation for the level sets around the front, calculation of variables such as curvature becomes more accurate. The algorithm to be presented avoids all re-initialization, which can often perturb the front, and the problem of bunching or stretching is greatly ameliorated.

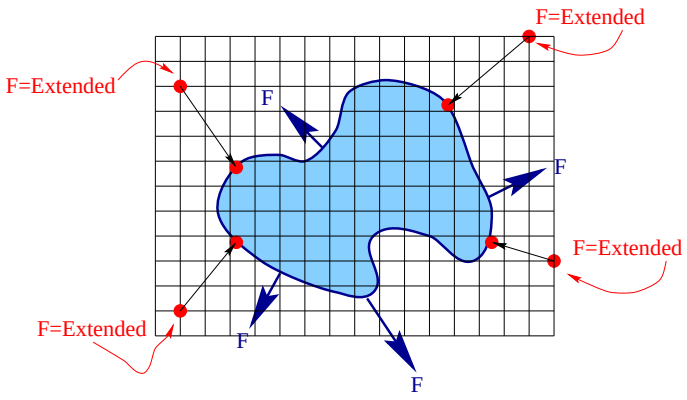


Figure 5: Constructing extension velocities.

How much freedom does one have in the construction of this extension velocity F^{ext}? Beyond the requirement that equal the velocity on the front itself, there is considerable freedom. The original level set calculations in [52] 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 non-geometric/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, Talbot, and Sethian [58] 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 Chen, Hou, Merriman, and Osher [16] and Sussman, Smereka, and Osher [84]. In these simulations, some bunching and flattening of the level set function occurs. This is repaired at every time step through a re-initialization process which rebuilds the signed distance function using an iterative process given in [84].

When there is no choice available for an extension velocity, Malladi, Sethian, and Vemuri [47] 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 [79] 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 "non-physical" 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, Merriman, Osher, and Smereka [17] 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 re-initialization 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? First, it should match the given velocity on the front itself. Second, it is desirable if 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 one builds an extension velocity which satisfies

$$\nabla F_{\text{ext}} \cdot \nabla \phi = 0. \tag{15}$$

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, for example, Zhao et al.[91]), 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 zero because $|\nabla\phi(x, t = 0)| = 1$. Thus, one solution satisfies $|\nabla\phi| = 1$; this plus a uniqueness result for this differential equation shows 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^n_{ij}, one first constructs a signed distance function $\bar{\phi}^n_{ij}$ around the zero level set. Simultaneous with this construction, one then constructs the extension velocity F_{ext} satisfying Eqn. 15. 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 a method of 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 re-initialize 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 re-initializes 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 re-initialization 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 in order 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 like 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 re-initialized. 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

To summarize, two ideas which underpin level set methods are the link between schemes for hyperbolic fronts and propagating interfaces given in [63, 64], 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. In order 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

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

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 Re-initialization and Extension Velocities

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

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

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 re-initializing level set functions and constructing extension velocities. We refer the reader to [76] and [75] for a large collection of applications based on this technique.

5.1 Fast Marching Methods

Consider the following upwind finite difference scheme for the Eikonal equation, namely

$$\begin{bmatrix} \max(D^{-x}_{ijk}u, -D^{+x}_{ijk}u, 0)^2 + \\ \max(D^{-y}_{ijk}u, -D^{+y}_{ijk}u, 0)^2 + \\ \max(D^{-z}_{ijk}u, -D^{+z}_{ijk}u, 0)^2 \end{bmatrix}^{1/2} = F_{ijk}, \tag{16}$$

which was discussed by Rouy and Tourin [59]. One approach to solving this finite difference scheme (see [59]), 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 [69] is connected to Huygen's principle. The viscosity solution to the Eikonal equation $|\nabla u(x)| = F(x)$ can be interpreted through Huygen'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 [26], 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, and fans 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. 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 Eqn. 16 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 Figure 6):

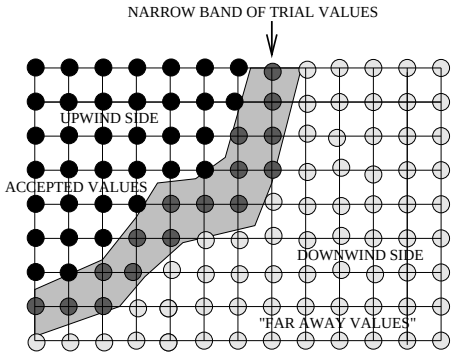


Figure 6: Upwind construction of Accepted Values

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 Eqn. 16. 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 u.
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 u at all *Considered* neighbors of $Trial$ by solving the piecewise quadratic equation according to Eqn. 16.
4. Add point $Trial$ to *Accepted*; remove from *Considered*
5. Return to top until the *Considered* set is empty.

This is the Fast Marching Method given by Sethian in [69]. 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 u. An efficient scheme for doing so, discussed in detail in [75], 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 re-order the heap in $O(\log N)$ steps. Thus, 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 re-ordered each time the values are changed.

The Fast Marching Method evolved in part from examining the limit of the Narrow Band level set method 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 [76], first order unstructured meshes by Kimmel and Sethian [38], and higher order unstructured meshes by Sethian and Vladimirsky [80]. Some early applications include photolithography in [70], a comparison of a similar approach with volume-of-fluid techniques in [32], a fast algorithm for image segmentation in [45] and computation of seismic travel times by Sethian and Popovici [78]; see also [87] for a different Dijkstra-like algorithm 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 higher order Fast Marching Method introduced in [75, 76], we include it here for completeness. Folllowing that discussion, we consider now the switch functions defined by (the expressions are similar in y and z)

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

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

We can then use these operators in the Fast Marching Method, namely,

$$\begin{bmatrix} \max\left[\left[D_{ijk}^{-x}T + \mathrm{switch}_{\mathrm{ijk}}^{-\mathrm{x}}\frac{\Delta \mathrm{x}}{2}\mathrm{D}_{\mathrm{ijk}}^{-\mathrm{x}-\mathrm{x}}\mathrm{T}\right], -\left[D_{ijk}^{+x}T - \mathrm{switch}_{\mathrm{ijk}}^{+\mathrm{x}}\frac{\Delta \mathrm{x}}{2}\mathrm{D}_{\mathrm{ijk}}^{+\mathrm{x}+\mathrm{x}}\mathrm{T}\right], 0\right]^2 \\ + \\ \max\left[\left[D_{ijk}^{-y}T + \mathrm{switch}_{\mathrm{ijk}}^{-\mathrm{y}}\frac{\Delta \mathrm{y}}{2}\mathrm{D}_{\mathrm{ijk}}^{-\mathrm{y}-\mathrm{y}}\ \mathrm{T}\right], -\left[D_{ijk}^{+y}T - \mathrm{switch}_{\mathrm{ijk}}^{+\mathrm{y}}\frac{\Delta \mathrm{y}}{2}\mathrm{D}_{\mathrm{ijk}}^{+\mathrm{y}+\mathrm{y}}\mathrm{T}\right], 0\right]^2 \\ + \\ \max\left[\left[D_{ijk}^{-z}T + \mathrm{switch}_{\mathrm{ijk}}^{-\mathrm{z}}\frac{\Delta \mathrm{z}}{2}\mathrm{D}_{\mathrm{ijk}}^{-\mathrm{z}-\mathrm{z}}\ \mathrm{T}\right], -\left[D_{ijk}^{+z}T - \mathrm{switch}_{\mathrm{ijk}}^{+\mathrm{z}}\frac{\Delta \mathrm{z}}{2}\mathrm{D}_{\mathrm{ijk}}^{+\mathrm{z}+\mathrm{z}}\mathrm{T}\right], 0\right]^2 \end{bmatrix}^{1/2} \tag{17}$$

$$= \frac{1}{F_{ijk}}.$$

This scheme attempts to use a second order one-sided upwind stencil whenever points are available, but reverts to a first order scheme in the other cases. It provides higher accuracy in regions of smoothness; the ultimate accuracy depends on the relationship of causality to shock lines in the solution. For details and discussion, see [75, 76].

5.2 Using Fast Marching Methods for Re-initialization and Extension Velocities

We can now use the techniques given by Adalsteinsson and Sethian [5] which exploit Fast Marching Methods to both re-initialize 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

$$\nabla\phi^{\mathrm{temp}} \cdot \nabla F_{\mathrm{ext}} = 0,$$

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 level set function ϕ^n). The most accurate way to build values to initialize 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.

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

$$\nabla F_{\text{ext}} \cdot \nabla \phi^{\text{temp}} = 0. \tag{18}$$

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 Eqn. 18. 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 satisfy an upwind version of Eqn. 18, 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

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

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 allows one to efficiently re-initialize and build extension velocities; higher order Fast Marching Methods 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. [91, 68], multiple junctions by Merriman et al.[48], level set methods for unstructured meshes by Barth and Sethian, including terms for curvature flow [10], adaptive mesh refinement schemes by Milne [49], higher order Fast Marching Methods [76], Fast Marching Methods for manifolds by Kimmel and Sethian [38] as well as certain types of non-Eikonal static Hamilton-Jacobi equations by Sethian and Vladimirsky [80], level

set flows in arbitrary co-dimension by Ambrosio and Sonar [6], hybrid methods, including coupled level set/volume-of-fluid techniques by Bourlioux [11] and marker/level set methods by Hou et al. [35], parallel versions [65], and extensions to motion under the intrinsic Laplacian of curvature by Chopp and Sethian [22, 23]. We refer the reader to these papers, the review in [75], 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 re-initializations, 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 re-initialization 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 re-initialization.** Re-initialization has a tendency to move the location of the interface. While higher order methods can help, including those that attempt to either re-distribute mass or solve an associated constraint problem, our experience is that the best approach is to limit re-initialization. This is one of the reasons that the size of the narrow band in the Narrow Band method is to chosen large enough to limit re-initialization, rather than 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 second order in time Heun's method.

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 $O(N^3 k)$, where k is the width of this narrow band, providing the optimal technique for implementing level set methods. On the other hand, 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, time step restrictions in terms of CFL conditions based on the speed F influences 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 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 re–used.

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, in order to obtain the speed F on the interface

A flow chart for the implementation is shown in Figure 7

III. Two 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, 3, 4, 32, 77, 70], geometry and minimal surfaces [7, 19, 20, 21], combustion and detonation [8, 9, 29, 58, 93, 94], fluids and surface tension-driven flows [12, 16, 41, 50, 82, 83, 84, 90, 91, 92, 94], shape recognition and segmentation [13, 15, 42, 44, 47, 61], crystal growth [17, 79], liquid bridges [18], groundwater flow [34], constructing geodesics [36, 38], robotic navigation and path planning [37], inverse problems [60], grid generation [67], and seismology [78].

In Figure 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 perspective on how the theory, algorithms, and applications have evolved. The text and bibliography of [75] gives a somewhat more complete sense of the literature and the range of work underway.

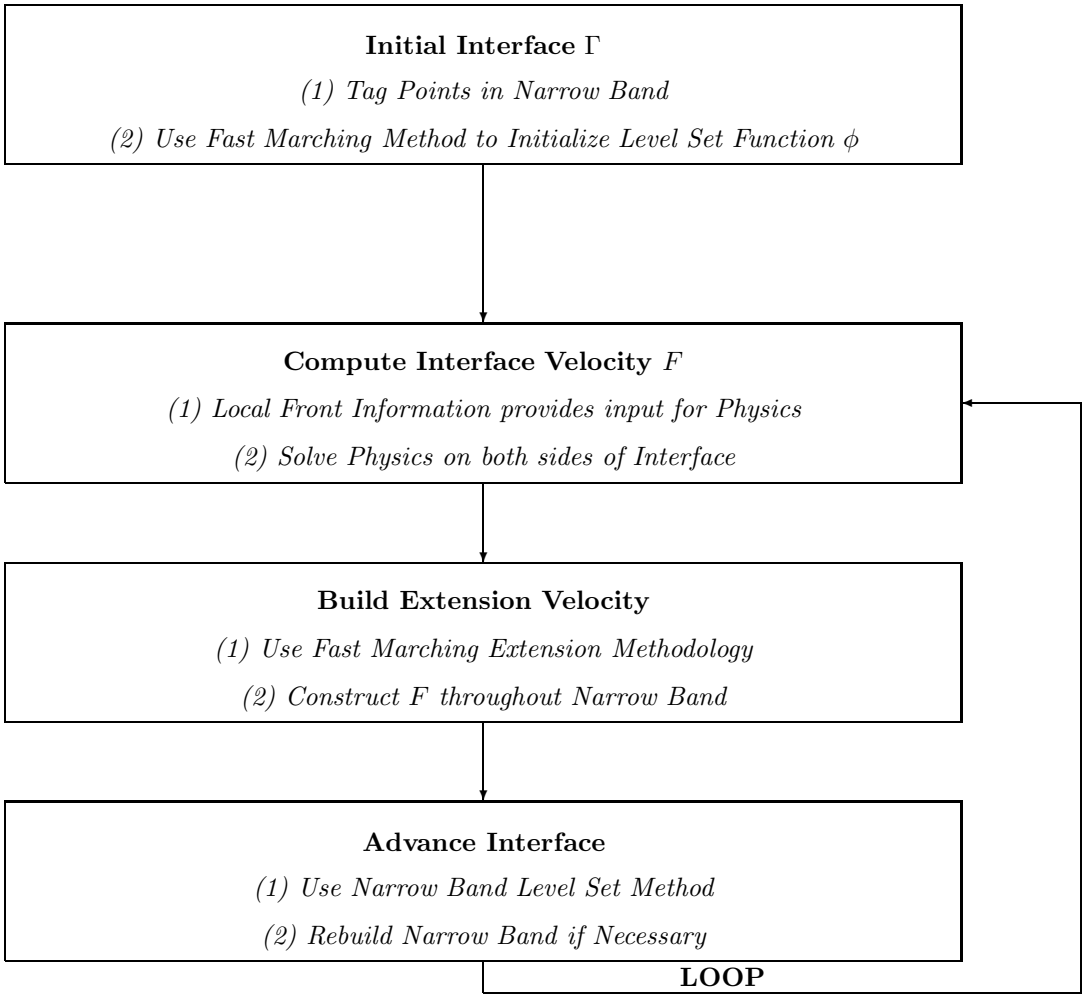


Figure 7: Flow Chart for Implementing Narrow Band Level Set Methods

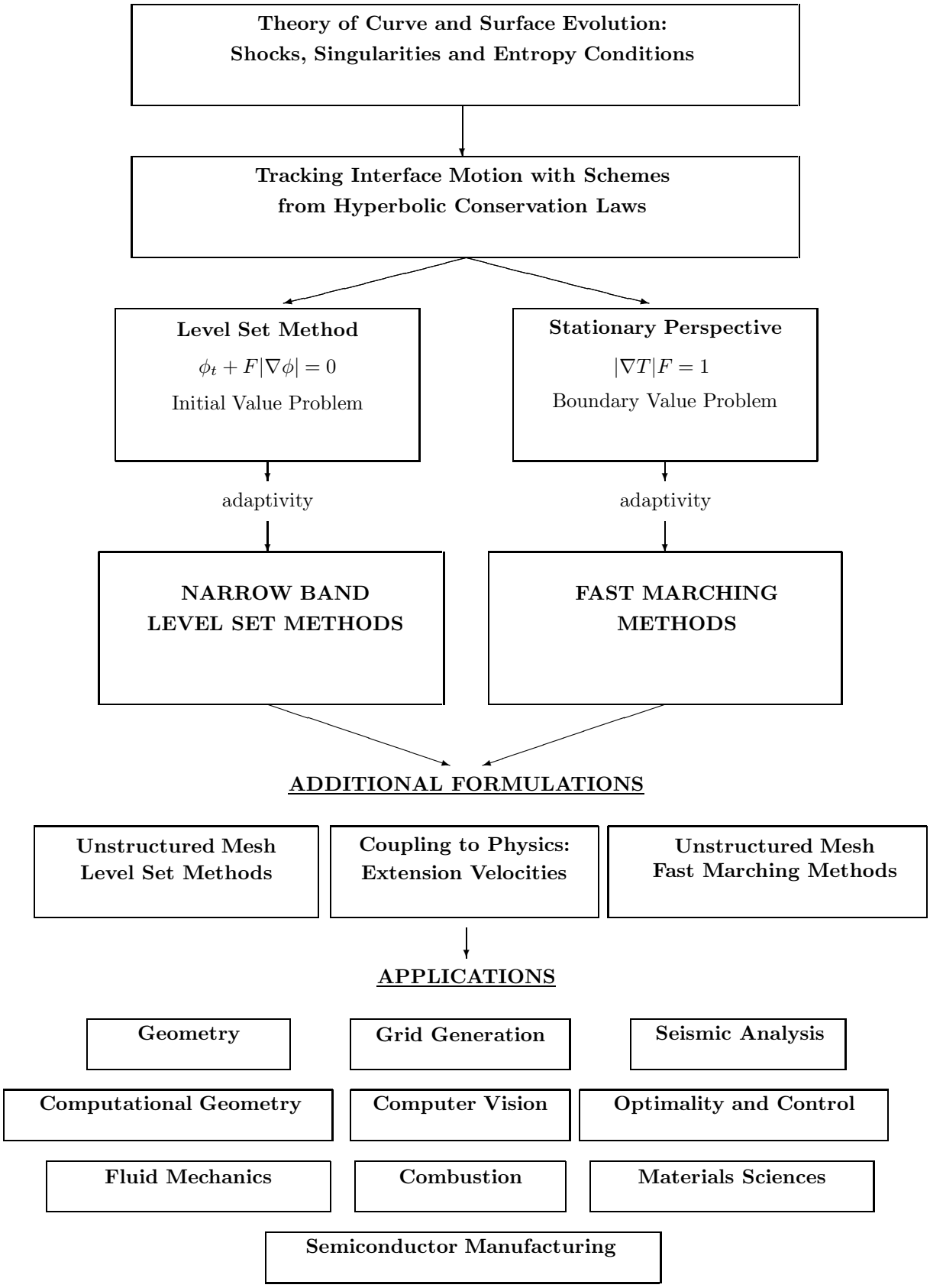


Figure 8: Algorithms and Applications for Interface Propagation

In the next sections, we discuss two applications in detail. The first, semiconductor processing, is chosen because it requires much of the above methodology in order to obtain the accuracy, efficiency and robustness required in semiconductor manufacturing, and because the results have been so closely matched with experiment. The second, 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 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.

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" that is sensitive to light 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 a large collection by physical factors, including the visibility of the etching/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 interface 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, 3, 4]; complete details may be found therein (see [77] 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 build-up 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 re-emitted back into the domain. This process is known as "re-emission" 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 "re-deposition". 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 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 Figure 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 indicator function $M_\Upsilon(x, x')$ is assigned which indicates whether the point x on the initial profile can be seen by the source point x'.

7.2 Equations of motion for etching/deposition

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

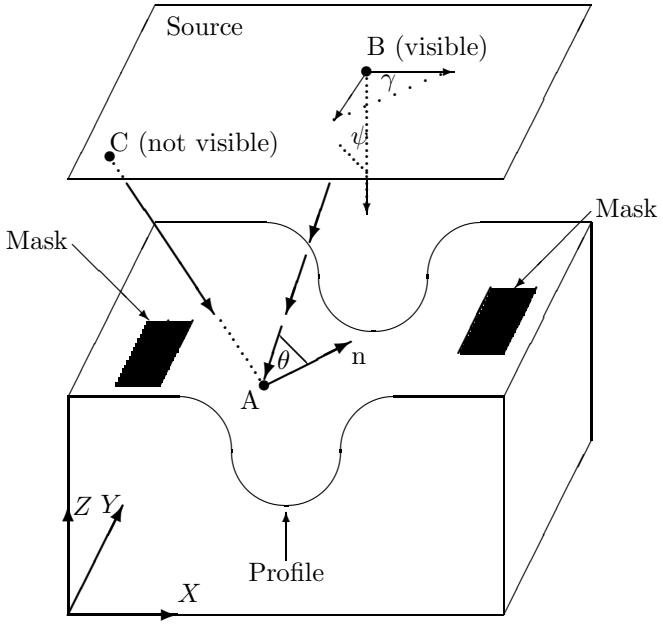


Figure 9: Variables and setup.

7.2.1 Etching

We consider two separate types of etching:

- $F^{Etching}_{Isotropic}$: *Isotropic etching.* Uniform etching, also known as chemical or wet etching.
- $F^{Etching}_{Direct}$: *Direct etching.* 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:

- $F^{Deposition}_{Isotropic}$: *Isotropic deposition.* Uniform deposition, also known as chemical or wet deposition.
- $F^{Deposition}_{Direct}$: *Direct deposition.* Deposition from an external source; this can be either a collection of point sources, or from 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.
- $F^{Deposition}_{Re-deposition}$: *Re-deposition.* 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 re-deposition flux function can depend on such factors as the local angle. A re-deposition

coefficient, $\beta_{Re-deposition}$ which can range from zero to unity represents the fraction of re-deposition that results from the etching process. A value of $\beta_{Re-deposition} = 1$ means that nothing is re-deposited and everything sticks.

- $F^{Deposition}_{Re-emission}$: *Re-emission deposition.* Particles are deposited by direct deposition might not stick and are may be re-emitted into the domain. The amount of particles re-emitted depends on a sticking coefficient $\beta_{Re-emission}$. If $\beta_{Re-emission} = 1$, nothing is re-emitted.

In Figure 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

$$F = F^{Etching}_{Isotropic} + F^{Etching}_{Direct} + F^{Deposition}_{Isotropic} + F^{Deposition}_{Direct} + F^{Deposition}_{Re-deposition} + F^{Deposition}_{Re-emission}. \tag{19}$$

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 re-deposition particles arising from the etching process; thus this term requires computing an integral over the profile itself. The sixth term, $F^{Deposition}_{Re-emission}$ is more problematic. Since every point on the front can act as a deposition source of re-emitted particles that do not stick, the total flux function 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 $\vec{n}$ be the unit normal vector at the point x, and finally, let $\vec{\alpha}$ 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:

$$F = \left[\begin{array}{c} \mathrm{Flux}_{Isotropic}^{Etching} \\ + \\ \int_{Source} \mathrm{Flux}_{Direct}^{Etching}(r,\psi,\gamma,\theta,x)\Upsilon(x,x')(\vec{n}\cdot\vec{\alpha})dx' \\ + \\ \mathrm{Flux}_{Isotropic}^{Deposition} \\ + \\ \int_{Source} \mathrm{Flux}_{Direct}^{Deposition}(r,\psi,\gamma,\theta,x)\Upsilon(x,x')(\vec{n}\cdot\vec{\alpha})dx' \\ + \\ \int_{\Omega}(1-\beta_{Re-deposition})\mathrm{Flux}_{Re-deposition}^{Deposition}(r,\psi,\gamma,\theta,x)\Upsilon(x,x')(\vec{n}\cdot\vec{\alpha})dx' \\ + \\ \int_{\Omega}(1-\beta_{Re-emission})\mathrm{Flux}_{Re-emission}^{Deposition}(r,\psi,\gamma,\theta,x)\Upsilon(x,x')(\vec{n}\cdot\vec{\alpha})dx' \end{array} \right] \tag{20}$$

7.4 Evaluating the terms

The integrals are performed in a straightforward manner. The front is located by constructing the zero level set of ϕ; in two dimensions it is represented 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 re-deposition, the integral over the entire profile is calculated by computing the visibility to all other control points, and the corresponding re-deposition 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 re-emission term

The sixth and last term is somewhat more time-consuming to evaluate, since it requires evaluation of the flux $Flux_{Re-emission}^{Deposition}$ 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, non-symmetric matrix which must be solved at every time step in order 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 multiply, 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 of 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

In order 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 coming 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, 3, 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 signed 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, 3, 4].

7.4.3 Surface diffusion

An additional physical effect comes from surface diffusion, which relates to the motion of metal boundaries, and corresponds to motion by the second derivative of curvature. Thus, we need to add an additional term of the form

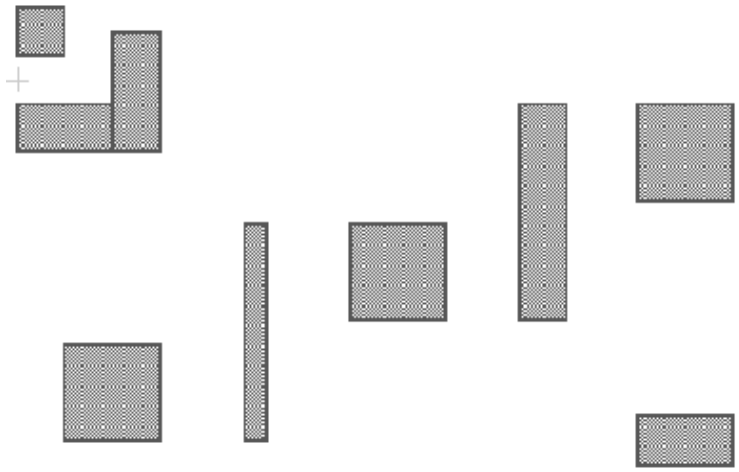
$$F = 1 + \epsilon \kappa_{\alpha\alpha}, \tag{21}$$

where α is an arc-length parameterization. The problem is delicate because Eqn. 21 is a time-dependent fourth order partial differential equation, and the presence of the fourth derivative requires 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 $O(\Delta t/\Delta h^4)$. We make use of the methodology given by Chopp and Sethian in [22]. Approximations and fast methods for solving this sort of flow may be found in [23].

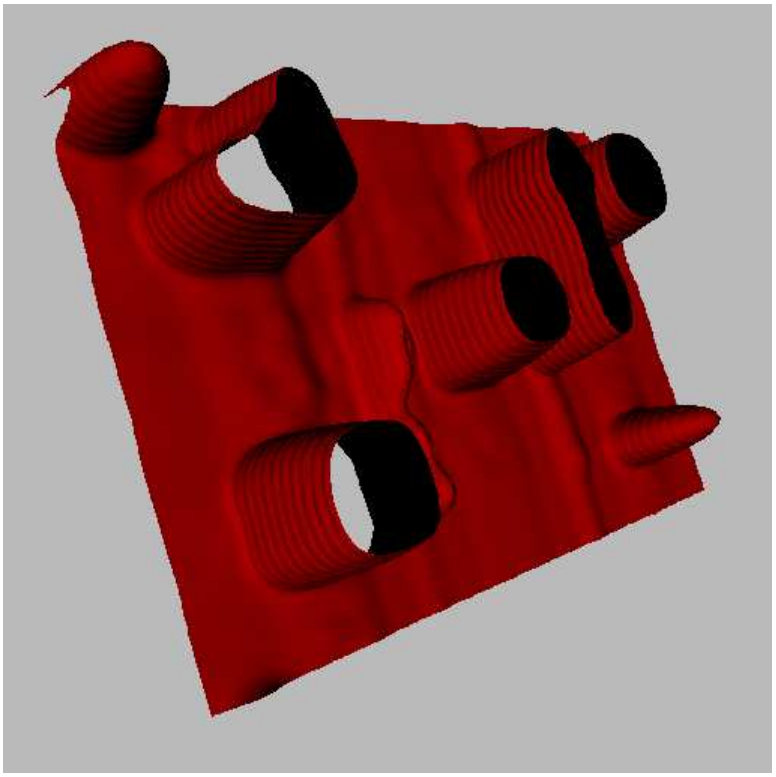
7.5 Results

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. In order 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 bake modules of TMA's Depict 4.0 [85] has been coupled to the Fast Marching Method. Figure 10(a) shows the top view of a mask placed on the board. The dark areas correspond to areas that are exposed to light. The standing waves in the etching profile are due to factors such as the reflectivity of the surface. In Figure 10(b) a view of the developed profile is shown from underneath; the etching of the holes and the presence of standing waves can be seen easily. For further results, see [70].



(a) Masking pattern



(b) Lithographic development: View from below

Figure 10: Lithographic development using Fast Marching Method.

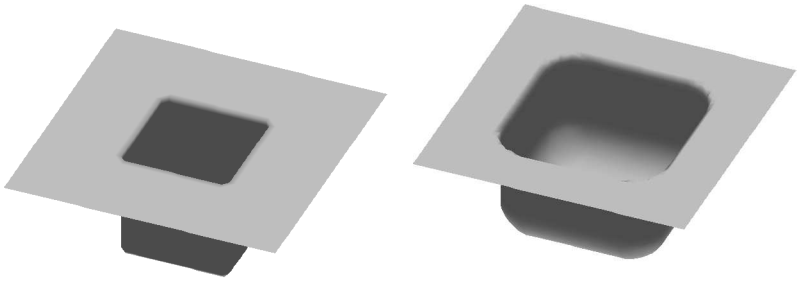


Figure 11: Isotropic etching into a hole.

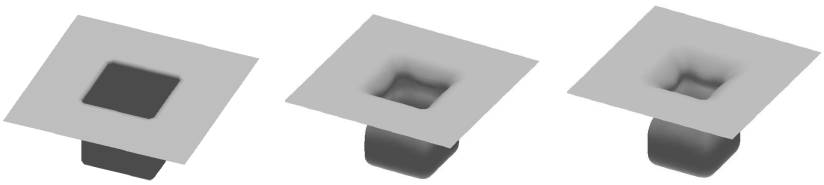


Figure 12: Source deposition into a hole.

7.5.2 Etching and deposition

Next, we show a straightforward calculation of isotropic etching into a hole, taken from [3]. In Figure 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 Figure 12, we show two 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.

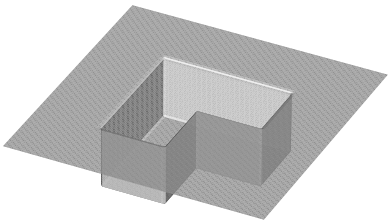
We end the basic calculations with the modeling (Figure 13) of the effect of non-convex sputter etch/ion milling of a saddle surface. The non-convex speed law $F = (1 + 4\sin^2(\theta))\cos\theta$ causes faceting of sharp corners and rounded polishing; for details of this effect, see [3]. Here, we use schemes non-convex Hamiltonians given in [53] for the level set update.

7.5.3 Complex simulations

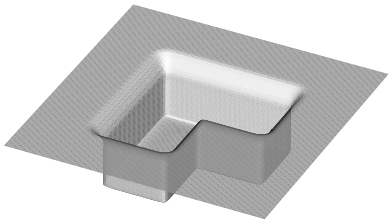
Next, we include an example of three-dimensional effects of re-deposition. The initial shape is a double-L, and we consider a combination of two cosine flux deposition sources. That is, the initial flux at each point is given by

$$Flux(x) = \cos^5(\theta_1)\cos(\theta_2) + \cos(\theta_1)\cos(\theta_2); \tag{22}$$

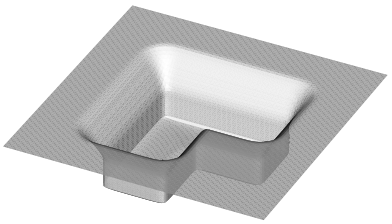
in addition, the second deposition term is given a sticking coefficient of 0.1, thus we also consider the effects of re-deposition. Here, θ_1 is the angle that the vector v from x to y makes with the normal at x, and θ_2 is the angle that the vector v makes with the vertical. The results are shown after some time evolution in Fig. 14b; a two-dimensional cross-sectional cut is shown in Figure 14c. For more simulations, see [4].



Initial Shape: $T = 0$

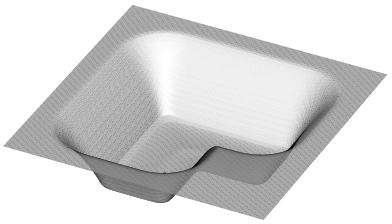


$F = [1 + 4\sin^2(\theta)]\cos(\theta) \quad T = 2$

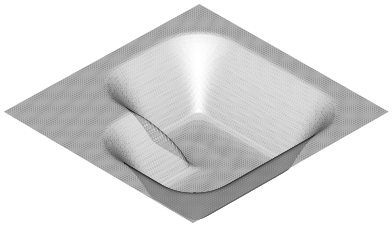


$F = [1 + 4\sin^2(\theta)]\cos(\theta) \quad T = 4$

$F = [1 + 4\sin^2(\theta)]\cos(\theta) \quad T = 6$

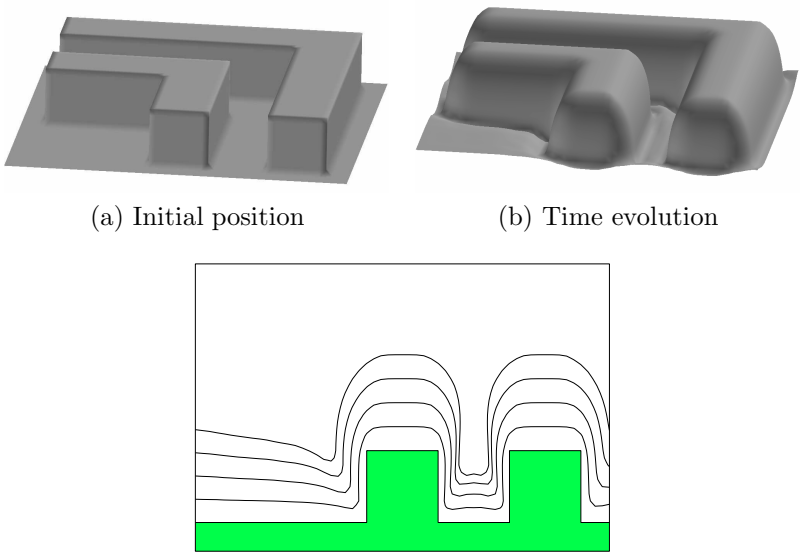


$F = [1 + 4\sin^2(\theta)]\cos(\theta) \quad T = 8$



Final rotated

Figure 13: Downward saddle under sputter etch.



(a) Initial position (b) Time evolution

(c) 2d cross-section

Figure 14: Three-dimensional evolution under cosine source distribution with sticking coefficient 0.1.

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. Figure 15 and Figure 16 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 [69].

	50 by 50			100 by 100		
Test	Run time	Steps	time/step	Run time	Steps	time/step
Lithography (Fast Marching)	6.9ms	NA	NA	26ms	NA	NA
Isotropic (Narrow Band)	82ms	24	34ms	0.4s	49	8ms
Unidirectional (with visibility)	0.4s	17	23ms	2.3s	34	70ms
Etching and re-deposition	1.7s	25	68ms	14s	51	0.3s
Deposition and re-deposition (Iterative model)	1.1s	17	65ms	12s	39	0.3s

Figure 15: Two-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)	0.16s	NA	NA	2.1s	NA	NA
Isotropic (Narrow Band)	1.3s	8	0.16s	13.6s	24	0.6s
Unidirectional (with visibility)	16.7s	24	0.7s	270s	47	5.7s
Etching and re-deposition	224s	12	19s	260m	25	10m
Deposition and re-deposition (Iterative model)	265s	11	24s	290m	23	12.6m

Figure 16: Three-dimensional timings.

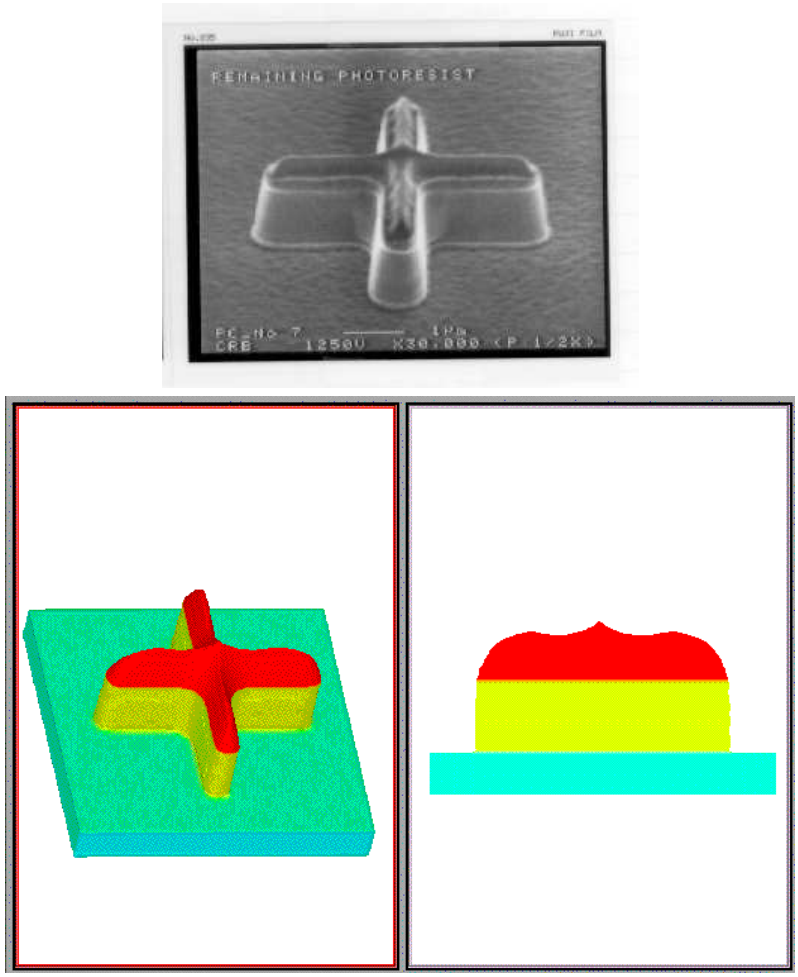


Figure 17: Ion-milling: experiment (top) vs. simulation (bottom).

7.6 Validation with experimental results

We end with a collection of applications of the Level Set/Fast Marching methodology comparing simulations with experiment analyzing various aspects of surface thin film physics. All the simulations in this section are performed using TERRAIN;[1] a commercial version of these techniques built by Technology Modeling Associates and specifically designed for process simulation. For further details about this code and its capabilities, see [86].

7.6.1 Ion milling

We begin with a comparison with experiment of an ion-milling process. Figure 17 shows an experiment on the top and a simulation at the bottom. We note that both the simulation and the experiment show the crossing non-convex curves on top of the structures, the sharp points, and the sloping sides.

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

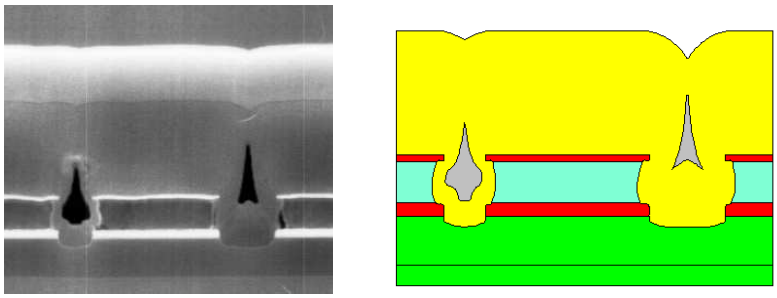


Figure 18: PECVD, small-scale structure: experiment (left) vs. simulation (right).

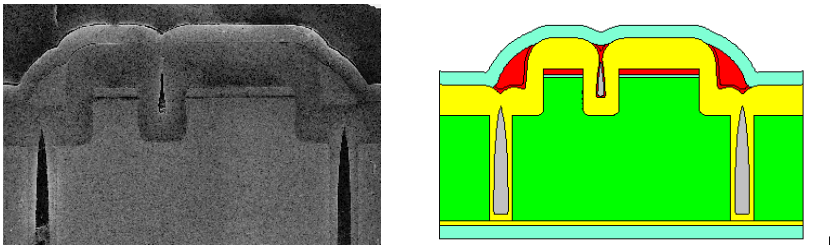


Figure 19: PECVD, small-scale structure: experiment (left) vs. simulation (right).

7.7 Plasma-enhanced chemical vapor deposition

Next, we show comparison with experiment of two plasma-enhanced chemical vapor deposition (PECVD) simulations. We show a series of experiments. First, two smaller structure calculations are used to verify the ability to match experiment. Figures 18 and 19 show these results. Figures 20 and 21 show more simulations for more complex structures.

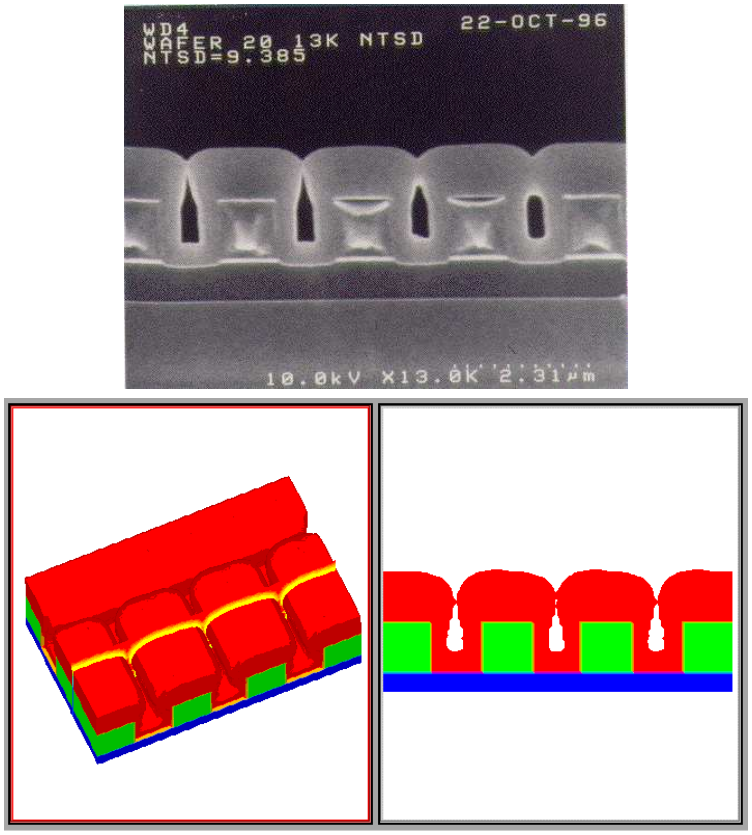


Figure 20: PECVD: experiment (top) vs. simulation (bottom).

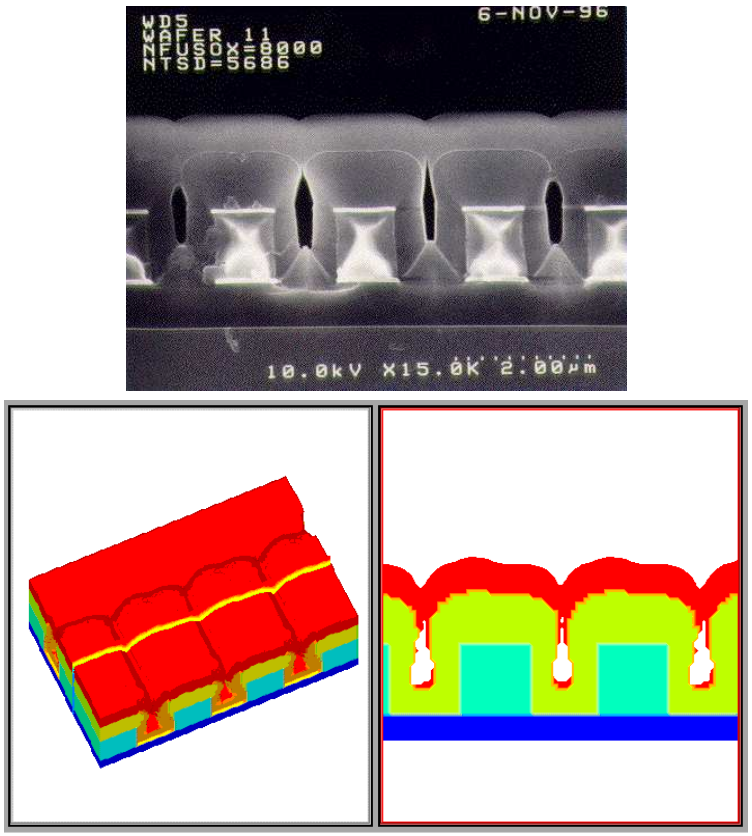


Figure 21: PECVD: experiment (top) vs. simulation (bottom).

7.8 Spin-on-glass

Next, in Figure 22, we show a spin-on-glass (SOG) simulation; in which the spin deposition is shown in sequence on top of a complex structure.

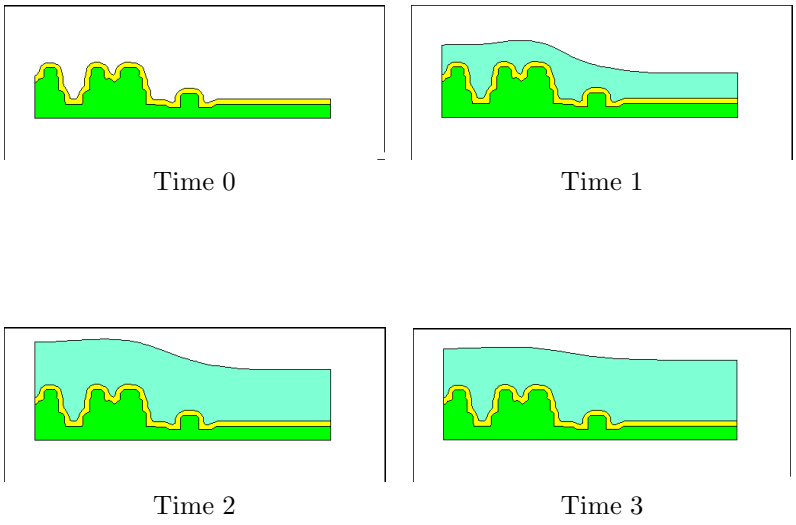
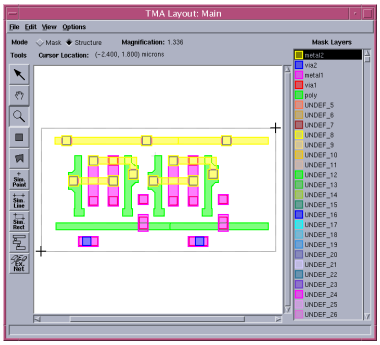


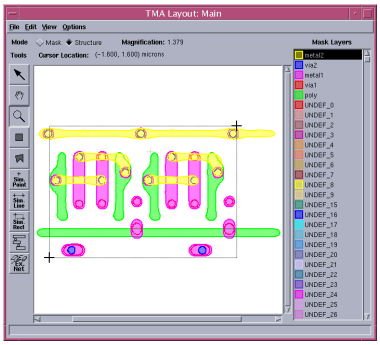
Figure 22: Spin-on-Glass: simulation time sequence.

7.9 SRAM simulations

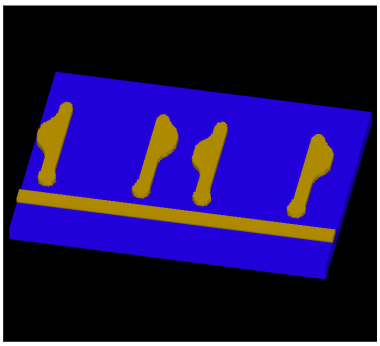
Finally, we show SRAM comparisons between experiment and simulations for both small structures (Figure 23) and large structures (Figure 24). Each figure shows the original layout together with the actual pattern printed through photolithography, followed by the sequential processing steps.



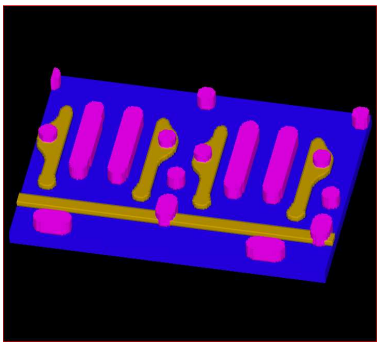
Original



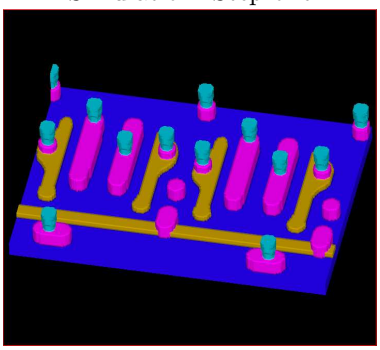
Printed layout



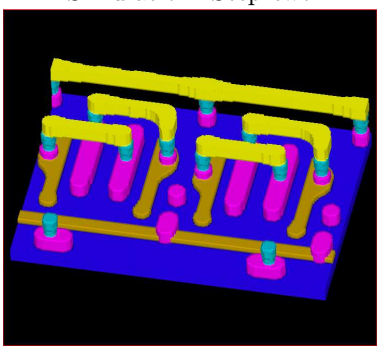
Simulation: Step one



Simulation: Step two

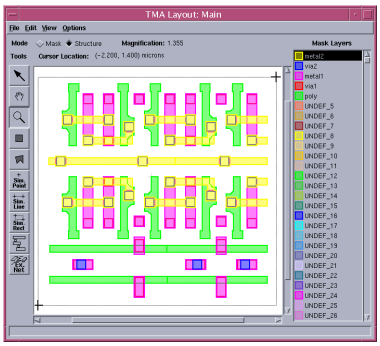


Simulation: Step three

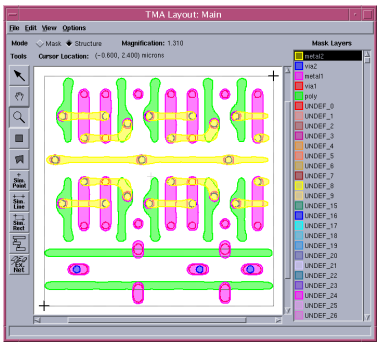


Simulation: Step four

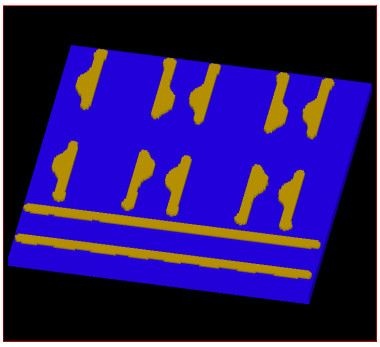
Figure 23: SRAM simulation: Experiment and simulation.



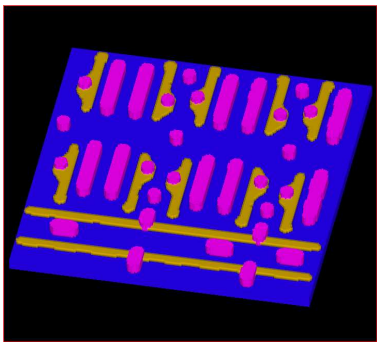
Original



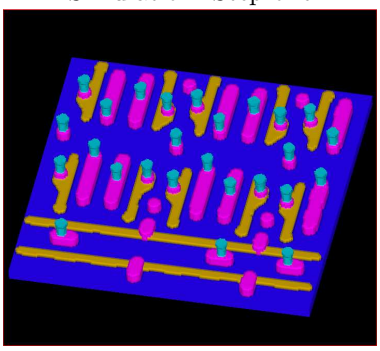
Printed layout



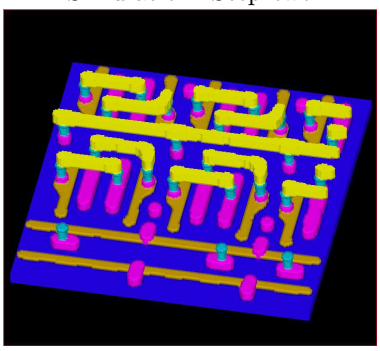
Simulation: Step one



Simulation: Step two



Simulation: Step three



Simulation: Step four

Figure 24: SRAM simulation: Experiment and simulation.

8 Optimal Structural Boundary Design

The second 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 are all taken from Sethian and Wiegmann [81]; 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 Figure 25). Suppose our

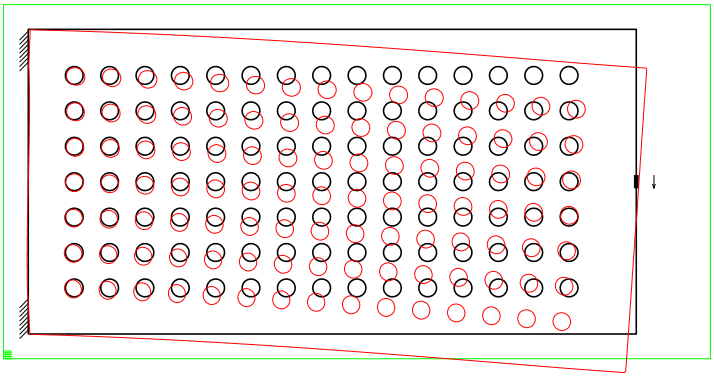


Figure 25: Bending of the initial design of a cantilever with 105 circular holes. Parts of the left boundary are clamped; on the rest of the boundary, including all holes, the traction is specified, with nonzero loading on a small portion about the center of the right boundary. The bending is beyond 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.

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 Figure 26. We can then try to add and remove material in order to reduce the weight in such a way that the compliance or stress do 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 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.

8.1 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 [81] a general technique (the Explicit Jump Immersed Interface

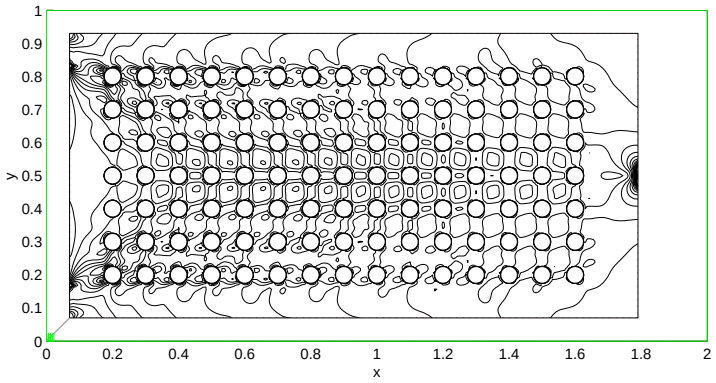


Figure 26: Stress contours for above initial configuration

Method) is given for solving the linear elastostatic equations in the displacement formulation and differencing the displacements using the Explicit Jump Immersed Interface Method (see [88]), a finite difference technique on uniform grids after LeVeque and Li's Immersed Interface Method ([40]) 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.

8.1.1 Summary of Equations and Strategy

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

$$\begin{aligned} -\mu\left(\Delta u + u_{xx} + v_{xy}\right) - \lambda\left(u_{xx} + v_{xy}\right) &= f^u \text{ in } \Omega, \\ -\mu\left(\Delta v + u_{xy} + v_{yy}\right) - \lambda\left(u_{xy} + v_{yy}\right) &= f^v \text{ in } \Omega. \end{aligned} \tag{23}$$

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 $C = \mu/(\mu + \lambda)$),

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

Displacement boundary conditions are

$$\mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_1 \subset \partial\Omega, \tag{25}$$

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

$$\sigma(\mathbf{u})\mathbf{n} = \mathbf{g} \text{ on } \Gamma_2 \subset \partial\Omega. \tag{26}$$

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

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 [81]. Armed with these algorithmic solvers, the sequence of events is as follows:

Main Algorithm

1 Initialize; find stresses in initial design.

2 While termination criteria are not satisfied do

3 Cut new holes.

4 Move boundaries.

5 Find displacements, stresses, etc.

6 If the constraints are violated reduce removal rate of material and revert to previous iteration.

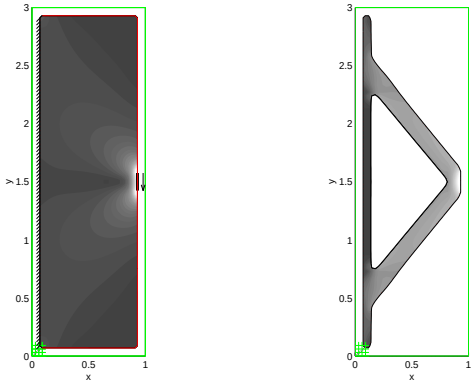
7 Update removal rate.

The approach in [81] 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.

8.2 Results

We show one result from [81], which is 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 [89, 56]), with a known solution for a simpler pin–jointed two–truss problem [57]. The optimal height in that case is twice the width of the structure.

In Figure 27a, taken from [81], we show clamping, loading and stresses in the initial design. Figure 27b shows the improved design under the combination Explicit Jump Immersed Interface Method and the Narrow Band Level Set Method. A large number of additional examples may be found in the original work [81].



Stress distribution in initial design Improved design

Figure 27: Improved Shape for Short Cantilever

Acknowledgements

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, and A. Wiegmann.

References

[1] Adalsteinsson, D., and Sethian, J.A., *A Fast Level Set Method for Propagating Interfaces*, J. Comp. Phys., 118, 2, pp. 269–277, 1995.

[2] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography I: Algorithms and Two-dimensional Simulations*, J. Comp. Phys., 120, 1, pp. 128–144, 1995.

[3] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography II: Three-dimensional Simulations*, J. Comp. Phys., 122, 2, pp. 348–366, 1995.

[4] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography III: Complex Simulations and Multiple Effects*, J. Comp. Phys., 138, 1, pp. 193-223, 1997.

[5] Adalsteinsson, D., and Sethian, J.A., *The Fast Construction of Extension Velocities in Level Set Methods*, 148, 1999, pp. 2-22.

[6] Ambrosio, L., and Soner, H.M., *Level Set Approach to Mean Curvature Flow in Arbitrary Co-dimension*, J. Diff. Geom, 43, 4, pp. 693–737, 1996.

[7] Angenent, S., Ilmanen, T., and Chopp., D.L., T., *A Computed Example of Nonuniqueness of Mean Curvature Flow in* R^3 , Comm. Partial Diff. Eqns., 20, 11-1, pp. 1937–1958, 1995.

[8] Aslam, T., Bzdil, J., and Stewart, D., *Level Set Methods Applied to Modeling Detonation Shock Dynamics*, J. Comp. Phys., 126, pp.390-409, 1996.

[9] Barles, G., *Remarks on a Flame Propagation Model*, INRIA Report 464, 1985.

[10] Barth, T.J., and Sethian, J.A., *Numerical Schemes for the Hamilton-Jacobi and Level Set Equations on Triangulated Domains*, J. Comp. Phys., 145, 1, pp. 1–40, 1998.

[11] Bourlioux, A., *A Coupled Level-Set Volume of Fluid Algorithm for Tracking Material Interfaces*, Sixth International Symposium on Computational Fluid Dynamics, Sept. 4-8, 1995, Lake Tahoe, NV.

[12] Brackbill, J.U., Kothe, D.B., and Zemach, C., *A Continuum Method for Modeling Surface Tension*, J. Comp. Phys., 100, pp. 335–353, 1992.

[13] Brewer, M.R., Malladi, R., Pankiewicz, G., Conway, B., and Tarassenko, L., *Methods for Large-Scale Segmentation of Cloud Images*, 1997 EUMETSAT Meteorological Satellite Data Users' Conference, Brussels, 1997.

[14] Bunner, B.; Tryggvason, G., *Direct numerical simulations of three-dimensional bubbly flows*, Physics of Fluids, 11, 8, pp. 1967-9, 1999.

[15] Caselles, V., Catte, F., Coll, T., and Dibos, F., *A Geometric Model for Active Contours in Image Processing*, Numer. Math., 66, pp. 1–31, 1993.

[16] Chang, Y.C., Hou, T.Y., Merriman, B., and Osher, S.J., *A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows*, Jour. Comp. Phys., 124, pp. 449-464, 1996.

[17] Chen, S., Merriman, B., Osher, S., and Smereka, P., *A Simple Level Set Method for Solving Stefan Problems*, J. Comp. Phys., 138, pp. 8–29, 1997.

[18] Chen, Y., Bi, Y., and Jiang, T., *The Liquid Bridge with Marangoni Effect*, Communications in Nonlinear Science and Numerical Simulation, 1, 1, pp. 48–51, 1996.

[19] Chopp, D.L., *Computing Minimal Surfaces via Level Set Curvature Flow*, Jour. of Comp. Phys., 106, pp. 77–91, 1993.

[20] Chopp, D.L., *Numerical Computation of Self-Similar Solutions for Mean Curvature Flow* J. Exper. Math., 3, 1, pp. 1–15, 1994.

[21] Chopp, D.L., and Sethian, J.A., *Flow Under Curvature: Singularity Formation, Minimal Surfaces, and Geodesics*, Jour. Exper. Math., 2, 4, pp. 235–255, 1993.

[22] Chopp, D.L., and Sethian, J.A., *Motion by Intrinsic Laplacian of Curvature*, CPAM Report PAM-746, Dept. of Mathematics, Univ. of California, Interfaces and Free Boundaries, 1, 1999.

[23] Chopp, D.L., Tongen, A., and Sethian, J.A., *Fast Approximations of Surface Diffusion*, submitted for publication, J. Comp. Phys., Jan., 2000.

[24] Crandall, M.G. and Lions, P.L., Two Approximations of Solutions of Hamilton-Jacobi Equations, Math. Comp., 167, 43, pp. 1–19, 1984

[25] Crandall, M.G., and Lions, P-L., *Viscosity Solutions of Hamilton–Jacobi Equations*, Tran. AMS, 277, pp. 1–43, 1983.

[26] Dijkstra, E.W., *A Note on Two Problems in Connection with Graphs*, Numerische Mathematic, 1:269–271, 1959.

[27] Esmaeeli, A. and Tryggvason, G., *Direct numerical simulations of bubbly flows. I. Low Reynolds number arrays*, J. Fluid Mechanics, 377, pp. 313-45, 1998

[28] Esmaeeli, A. and Tryggvason, G., *Direct numerical simulations of bubbly flows. II. Moderate Reynolds number arrays*, J. Fluid Mechanics, 385, pp. 325-358, 1999.

[29] Fedkiw, R.P., Aslam, T. and Shaojie, Xu, *The ghost fluid method for deflagration and detonation discontinuities.*, J. Comp. Phys., 154, 2, pp. 393-427, 1999.

[30] Glimm, J., Saltz, D., and Sharp, D.H., *Two-phase modelling of a fluid mixing layer.*, J. Fluid Mech., 378, pp.119-43, 1999.

[31] Glimm, J.; Grove, J.W , Xiao Lin Li, and Keh-Ming Shyue, *Three-dimensional front tracking*, SIAM Journal on Scientific Computing, 19, 3, pp.703-27., 1998.

[32] Helmsen, J., Puckett, E.G., Colella, P., and Dorr, M., *Two new methods for simulating photolithography development*, SPIE 1996 International Symposium on Microlithography, SPIE, v. 2726, June, 1996.

[33] Hirt, C.W., and Nicholls, B.D., *Volume of Fluid (VOF) Method for Dynamics of Free Boundaries*, J. Comp. Phys., 39, pp. 201–225, 1981.

[34] Holm, E., and Langtangen, H., *A Method for Simulating Sharp Fluid Interfaces in Groundwater Flow*, submitted for publication, Advances in Water Resouces, April, 1998.

[35] Hou, T.Y., Li, Zhilin, L., Osher, S.J., and Zhao. H.K., *A Hybrid Method for Moving Interfaces Problems with Application to the Hele-Shaw Flow*, J. Comp. Phys., 134, 2, pp. 236–52, 1997.

[36] Kimmel, R., Amir, A., and Bruckstein, A.M., *Finding Shortest Paths on Surfaces Using Level Sets Propagation*, IEEE Trans. Patt. Anal. Machine Intell., 17, 6, pp. 635–640, 1995.

[37] Kimmel, R., and Sethian, J.A., *Fast Marching Methods for Robotic Navigation with Constraints*, Center for Pure and Applied Mathematics Report, Univ. of California, Berkeley, May 1996, submitted for publication, Int. Journal Robotics Research, 1998.

[38] Kimmel, R., and Sethian, J.A., *Fast Marching Methods on Triangulated Domains*, Proc. Nat. Acad. Sci., 95, pp. 8341-8435, 1998.

[39] Kimmel, R., and Sethian, J.A., *Fast Voronoi Diagrams and Offsets on Triangulated Surfaces*, submitted for publication, CAD Special issue on Offsets, Sweeps, and Minkowsky Sums, G. Elber, Ed., July, 1998.

[40] LeVeque, R.J., and Li, Z., *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources*, SIAM J. Num. Anal., 13, pp. 1019–1044, 1994.

[41] Li, X.L., *Study of Three-Dimensional Rayleigh-Taylor Instability in Compressible Fluids Through Level Set Method and Parallel Computation*, Phys. Fluids A, 5, 1, pp. 1904–1913, 1993.

[42] Malladi, R., and Sethian, J.A., *A Unified Approach for Shape Segmentation, Representation, and Recognition*, Center for Pure and Applied Mathematics, Report 614, University of California, Berkeley, 1994

[43] Malladi, R., and Sethian, J.A., *Image Processing via Level Set Curvature Flow*, Proc. Natl. Acad. of Sci., 92, 15, pp. 7046–7050, 1995.

[44] Malladi, R., and Sethian, J.A., *A Unified Approach to Noise Removal, Image Enhancement, and Shape Recovery*, IEEE Trans. on Image Processing, 5, 11, pp. 1554–68, 1996.

[45] Malladi, R., and Sethian, J.A., *An $O(N \log N)$ Algorithm for Shape Modeling*, Proc. Nat. Acad. Sci., Vol. 93, pp. 9389-9392, 1996.

[46] Malladi, R., Sethian, J.A., and Vemuri, B.C., *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, 800, pp. 3–13, 1994.

[47] Malladi, R., Sethian, J.A., and Vemuri, B.C., *Shape Modeling with Front Propagation: A Level Set Approach*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 17, 2, pp. 158–175, 1995.

[48] Merriman, B., Bence, J., and Osher, S.J., *Motion of Multiple Junctions: A Level Set Approach*, Jour. Comp. Phys., 112, 2, pp. 334–363, 1994.

[49] Milne, B. *Adaptive Level Set Methods Interfaces*, PhD. Thesis, Dept. of Mathematics, University of California, Berkeley, CA., 1995.

[50] Mulder, W., Osher, S.J., Sethian, J.A., *Computing Interface Motion in Compressible Gas Dynamics*, Jour. Comp. Phys., 100, pp. 209–228, 1992.

[51] Noh, W., and Woodward, P., *A Simple Line Interface Calculation.* Proceedings, Fifth International Conference on Fluid Dynamics, Eds. A.I. vn de Vooran and P.J. Zandberger, Springer-Verlag, 1976.

[52] Osher, S., and Sethian, J.A., *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton–Jacobi Formulations*, Journal of Computational Physics, 79, pp. 12–49, 1988.

[53] Osher, S., and Shu, C., *High-Order Nonoscillatory Schemes for Hamilton–Jacobi Equations*, Jour. Comp. Phys., 28, pp. 907–922, 1991.

[54] Popinet, S. and Zaleski, S., *A front-tracking algorithm for accurate representation of surface tension*, Int. J. Numer. Meth. Fluids, 30, 6, pp. 775-93, 1999.

[55] Puckett, E.G., *A Volume-of-Fluid Interface Tracking Algorithm with Applications to Computing Shock Wave Refraction*, Proceedings of the 4th International Symposium on Computational Computational Fluid Dynamics, Davis, California, 1991.

[56] 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. Engng., 45:529—552, 1999.

[57] G. I. N. Rozvany, M. P. Bendsøe, and U. Kirsch. *Layout optimisation of structures*, Applied Mechanics Review, 48(2):41—119, 1995.

[58] Rhee, C., Talbot, L., and Sethian, J.A., *Dynamical Study of a Premixed V flame*, Jour. Fluid Mech., 300, pp. 87–115, 1995.

[59] Rouy, E. and Tourin, A., *A Viscosity Solutions Approach to Shape-From-Shading*, SIAM J. Num. Anal, 29, 3, pp. 867–884, 1992.

[60] Santosa, F., *A Level Set Approach for Inverse Problems Involving Obstacles*, ESIAM Control Optimization and Calculus of Variations, 1, pp. 17–33, 1996.

[61] Sarti, A., Ortiz, C., Lockett, S., and Malladi, R., *A Unified Geometric Model for 3D Confocal Image Analysis in Cytology*, LBL Report, Lawrence Berkeley National Laboratory, University of California, May, 1998, SIBGRAPI 98 proceedings, Rio de Janeiro, 1998, submitted to IEEE Trans. on Biomedical Engineering.

[62] Sethian, J.A., *An Analysis of Flame Propagation*, Ph.D. Dissertation, Dept. of Mathematics, University of California, Berkeley, CA, 1982.

[63] Sethian, J.A., *Curvature and the Evolution of Fronts*, Comm. in Math. Phys., 101, pp. 487–499, 1985.

[64] Sethian, J.A., *Numerical Methods for Propagating Fronts*, in Variational Methods for Free Surface Interfaces, Eds. P. Concus and R. Finn, Springer-Verlag, NY, 1987.

[65] Sethian, J.A., *Parallel Level Set Methods for Propagating Interfaces on the Connection Machine*, Unpublished manuscript, 1989.

[66] Sethian, J.A., *Numerical Algorithms for Propagating Interfaces: Hamilton–Jacobi Equations and Conservation Laws*, Journal of Differential Geometry, 31, pp. 131–161, 1990.

[67] Sethian, J.A., *Curvature Flow and Entropy Conditions Applied to Grid Generation*, J. Comp. Phys., 115, pp. 440–454, 1994.

[68] Sethian, J.A., *Algorithms for Tracking Interfaces in CFD and Material Science*, Annual Review of Computational Fluid Mechanics, 1995.

[69] Sethian, J.A., *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93, 4, pp.1591–1595, 1996.

[70] Sethian, J.A., *Fast Marching Level Set Methods for Three-Dimensional Photolithography Development*, Proceedings, SPIE 1996 International Symposium on Microlithography, Santa Clara, California, June, 1996.

[71] Sethian, J.A., *A Review of the Theory, Algorithms, and Applications of Level Set Methods for Propagating Interfaces*, Acta Numerica, Cambridge University Press, 1996.

[72] Sethian, J.A., Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences, First Edition, Cambridge University Press, 1996.

[73] Sethian, J.A., *Tracking Interfaces with Level Sets*, American Scientist, pp. 254–263, May–June, 1997.

[74] Sethian, J.A., *Fast Marching Methods and Level Set Methods for Propagating Interfaces* von Karman Institute Lecture Series, Computational Fluid Mechanics, 1998.

[75] Sethian, J.A., Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999.

[76] Sethian, J.A., *Fast Marching Methods*, SIAM Review, 41, July, 1999.

[77] Sethian, J.A., and Adalsteinsson, D., *An Overview of Level Set Methods for Etching, Deposition, and Lithography Development*, IEEE Transactions on Semiconductor Devices, 1996. 10, 1, pp.167-184, 1997.

[78] Sethian, J.A., and Popovici, M., *Fast Marching Methods Applied to Computation of Seismic Travel Times*, Geophysics, 64, 2, 1999.

[79] Sethian, J.A. and Strain, J.D., *Crystal Growth and Dendritic Solidification* J. Comp. Phys., 98, pp. 231–253, 1992.

[80] Sethian, J.A., and Vladimirsky, A., *Extensions to Triangulated Fast Marching Methods*, submitted for publication, Proc. Nat. Acad. Sci., Nov. 1999.

[81] Sethian, J.A., and Wiegmann, A., *Structural and Boundary Design via Level Set and Immersed Interface Methods*, submitted for publication, J. Comp. Phys., Nov. 1999.

[82] Son, G., and Dhir,V.K., *Numerical Simulation of Film Boiling Near Critical Pressures with a Level Set Method*, J. Heat Transfer, 120, pp. 183–192, 1998.

[83] Sussman, M., Smereka, P., *Axisymmetric Free Boundary Problems*, preprint, 1998.

[84] Sussman, M., Smereka, P. and Osher, S.J., *A Level Set Method for Computing Solutions to Incompressible Two-Phase Flow*, J. Comp. Phys. 114, pp. 146–159, 1994.

[85] Technology Modeling Associates, Three-Dimensional Photolithography Simulation with Depict 4.0, Technology Modeling Associates, Internal Documentation, January 1996.

[86] Terrain, Topography simulation for IC technology; Reference manual, Avant! Corporation, Fremont, CA, U.S.A., 1998.

[87] Tsitsiklis, J.N., *Efficient Algorithms for Globally Optimal Trajectories*, IEEE Transactions on Automatic Control, Volume 40, pp. 1528-1538, 1995.

[88] A. Wiegmann and K. P. Bube. *The Explicit–Jump Immersed Interface Method: Finite Difference Methods for PDE with piecewise smooth solutions*, , to appear, SIAM J. Numer. Anal., 2000.

[89] Y. M. Xie and G. P. Steven. Evolutionary Structural Optimization, Springer, 1997.

[90] Zhang, H., Zheng, L.L., Prasad, V., and Hou, T., *A Curvilinear Level Set Formulation for Highly Deformable Free Surface Problems with Application to Solidification*, Numerical Heat Transfer, Part B, Vol. 34, pp. 1–20, 1997.

[91] Zhao, H-K., Chan, T., Merriman, B., and Osher, S., *A Variational Level Set Approach to Multiphase Motion*, Jour. Comp. Phys., 127, pp. 179–195, (1996).

[92] Zhao, H-K., Merriman, B., Osher, S., and Wang, L., *Capturing the Behaviour of Bubbles and Drops Using the Variational Level Set Approach*, J. Comp. Phys., 143, 2, pp. 495–518, 1998.

[93] Zhu, J., and Ronney, P.D., *Simulation of Front Propagation at Large Non-dimensional Flow Disturbance Intensities*, Comb. Sci. Tech., 100, pp. 183–201, 1995.

[94] Zhu, J., and Sethian, J.A., *Projection Methods Coupled to Level Set Interface Techniques*, J. Comp. Phys., 102, pp. 128–138, 1992.