Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow [*]

Frank Losasso [†] Ronald Fedkiw [†] Stanley Osher [‡]

May 3, 2005

Abstract

Since the seminal work of [92] on coupling the level set method of [69] to the equations for two-phase incompressible flow, there has been a great deal of interest in this area. That work demonstrated the most powerful aspects of the level set method, i.e. automatic handling of topological changes such as merging and pinching, as well as robust geometric information such as normals and curvature. Interestingly, this work also demonstrated the largest weakness of the level set method, i.e. mass or information loss characteristic of most Eulerian capturing techniques. In fact, [92] introduced a partial differential equation for battling this weakness, without which their work would not have been possible. In this paper, we discuss both historical and most recent works focused on improving the computational accuracy of the level set method focusing in part on applications related to incompressible flow due to both its popularity and stringent accuracy requirements. Thus, we discuss higher order accurate numerical methods such as Hamilton-Jacobi WENO [46], methods for maintaining a signed distance function, hybrid methods such as the particle level set method [27] and the coupled level set volume of fluid method [91], and adaptive gridding techniques such as the octree approach to free surface flows proposed in [56].

[*]Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-97-1-0027, ONR N00014-03-1-0071, ONR N00014-02-1-0720, ARO DAAD19-03-1-0331, NSF DMS-0106694, NSF ITR-0121288, NSF IIS-0326388, NSF ACI-0323866, NSF ITR-0205671 and NIH U54 RR021813.

[†]Computer Science Department, Stanford University, Stanford, CA 94305.

[‡]Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90095.

1 Introduction

The level set method computes the motion of an interface Γ of codimension one that bounds a region $\Omega^- \subset R^n$. The level set function $\phi(\vec{x}, t)$ is a Lipschitz continuous function with the following properties:

$$\begin{aligned} \phi(\vec{x}, t) > 0 \qquad & \text{for } \vec{x} \notin \Omega^- \\ \phi(\vec{x}, t) \leq 0 \qquad & \text{for } \vec{x} \in \Omega^-. \end{aligned}$$

Note that $\phi = 0$ is typically included with the negative numbers so that the interface Γ lies in between $\phi > 0$ and $\phi = 0$. Although embedding an interface in a higher dimensional set may at first seem inefficient, it is this higher dimensionality that provides for the automatic handling of topological changes and the robust computation of geometric information.

The interface motion is given by a velocity field $\vec{u}(\vec{x}, t)$, which can be a direct function of the interface geometry or specified externally. The velocity field is typically defined in a band local to the interface with the bandwidth depending on the discretization. The elementary equation used for interface evolution is

$$\phi_t + \vec{u} \cdot \nabla \phi = 0. \tag{1}$$

Equation 1 was introduced for level set advection by Osher and Sethian in the original level set paper [69] and denoted the *level set equation.* About twenty-five years earlier, [57] discussed this same equation in the context of combustion, and that community still largely refers to it as the *G-equation* (replacing ϕ with G). Although [57] and others in the combustion community, see especially [102] and related work, originally focused on the G-equation in the context of theory and asymptotic analysis, the numerical techniques pioneered by [69] later enabled that community to explore the G-equation in numerical simulations as well. It is interesting to note that [69] cited [57] in the context of combustion as an application area, apparently without noticing that the level set equation already had a name and a widespread *non*-numerical following in combustion. On another note, two rather obscure works [25, 26] (published about 10 years before [69]) have recently surfaced bearing a remarkable resemblance to the level set method.

The discretization of the level set equation can lead to a significant numerical dissipation that usually manifests itself as a loss of mass (or volume) in areas of high curvature or other under-resolved regions. Those who use competing methods typically cite this mass loss as the main reason for doing so. One alternative to level set methods is the front tracking approach, see e.g. [99] which pioneered numerical methods for incompressible two-phase flow. Lagrangian front tracking methods do not suffer from the typical accuracy problems characteristic of Eulerian methods, since the discretized interface consists of particles that can be advected by simply solving the ordinary differential equation $\vec{x}_t = \vec{u}(\vec{x})$. The drawback however is that the elements that make up the interface (segments or triangles in two or three spatial dimensions, respectively) can become highly distorted leading to a loss of accuracy as the interface is deformed. Moreover, if the interface changes topology, some difficult remeshing is required (especially in three spatial dimensions) to restore the interface elements to a usable and accurate state. We refer the interested reader to [104] which uses a combination of a mesh refinement technique and a least-squares based scheme to smooth deforming elements and to [40] which describes a simplified front tracking algorithm and compares it to the level set method. In an attempt to improve the accuracy of the level set method, [27] proposed hybridizing the Eulerian level set method with tracked Lagrangian particles. This particle level set method draws its accuracy from the tracked particles and derives its connectivity from the level set approach combining the best aspects of both methods. The result is a robust, accurate method that is simple to implement even in three spatial dimensions.

Another alternative to the level set method is the volume of fluid (VOF) method, see for example [8, 73]. The basic idea behind this method is to discretize the equations for conservation of volume in either conservative flux or equivalent form resulting in near perfect volume conservation except for small over and under shoots. The main disadvantage of the VOF method is that it suffers from the numerical errors typical of Eulerian schemes such as the level set method. The imposition of a volume preservation constraint does not eliminate these errors, but instead changes their symptoms replacing mass loss with inaccurate mass motion leading to small pieces of fluid nonphysically being ejected as floatsam or jetsam, artificial surface tension forces that cause parasitic currents, and an inability to accurately calculate geometric information such as normals and curvature. Possibly the most disturbing aspect of the VOF method is that the reconstructed interface is discontinuous between cells. [91] proposed hybridizing the VOF and level set methods relying on the VOF method to better preserve volume and the level set method to provide more accurate geometric information such as normals and curvature. One drawback to their approach is that both methods are Eulerian and have similar numerical difficulties, as opposed to the particle level set method which mixes an Eulerian scheme with a Lagrangian one.

In order to obtain more accuracy in regions of the flow where it is required, adaptive methods can be employed. The goal of adaptive grid techniques is to save on memory and/or processor time by placing more computational nodes in areas that are under-resolved. There are essentially three separate approaches to adaptive level set methods. The most trivial approach involves allocating the full amount of memory everywhere, but only computing near the interface as in [21, 1, 71]. The next simplest strategy to implement are the adaptive mesh refinement (AMR) techniques pioneered by [7] and later by [6]. In this approach, a number of uniform grids of different resolutions are used. When more detail is required in a specific region of the flow, a fine grained uniform patch is placed in that region. See [88] for AMR used in conjunction with two-phase incompressible flow. While AMR is the preferred technique for flows with shock waves where one wants to minimize adjacent grid cells with different resolution to avoid spurious shock reflections (see e.g. [20]), it is wasteful in the sense that many fine grid cells are introduced in each patch even when only a few may be required. A more efficient method is octree based, and a number of two dimensional quadtree based level set methods were proposed in [86, 85]. [74] claims to have the first octree based implementation of single phase incompressible flow without level set or interfaces, and [56] proposes the first octree algorithms for free surface flows using the level set method. Octree based methods are much more general than AMR techniques, but the price of this generality is a significantly more complicated data structure which can pose difficulties for algorithm implementation and computational efficiency. We address some of these inefficiencies and discuss possible remedies below. Other interesting work includes the adaptive mesh redistribution of [93] and the finite element based fluids of [94, 19, 18]. We also refer the interested reader to the recent work of [41] which presents a quadtree based implementation of the VOF method.

Level set methods have enjoyed widespread popularity in the computational physics community. For example, the ghost fluid method (GFM) originally developed in [33] for two-phase compressible flow captures and preserves discontinuities across an interface represented by a level set. Using the GFM, these level set algorithms have been extended to detonations and deflagrations [34], mixed compressible and incompressible flow [11], two-phase incompressible flow [50, 54], low speed flames [65], Stefan problems for crystal growth [39, 38, 37], free surface flows [30], etc. For example, an extension of the GFM was used to couple solid materials with strength to chemically reacting compressible flow in order to study the dynamic response of materials under loads generated by solid explosives. This methodology is the "workhorse" behind the Department of Energy ASCI center at Caltech for studying solid fluid interactions, see e.g. [32, 5, 4]. Also notable is the recent work on a fully conservative version ghost fluid method [66].

Besides computational physics, level set techniques have been exploited in a number of other areas as well. [68] (see also [96]) compiles an interesting set of applications in computer vision. For example, [12] (see also [103]) proposed a level set model for active contours to detect objects whose boundaries are not necessarily defined by a gradient. This work was based on the Mumford-Shah minimal partition functional [62]. The main drawback of this algorithm is the computational expense incurred from the parabolic nature of a nonlinear partial differential equation. To remedy this, [36] showed a connection between the model proposed in [12] and a basic k-Means algorithm with a nonlinear diffusion preprocessing step decreasing the computational cost by a few orders of magnitude. The key observation was that only the sign of the level set function is needed, and thus it suffices to alternate between positive and negative values in order to produce segmentation results. Later work by [83] obtained similar results with related ideas, and now others are working in this direction as well, see e.g. [31].

Computer graphics has recently become a major application area for the level set method and its derivatives. For example, [63] used level sets to define a framework for efficient and intuitive surface editing operations. This framework alleviated some of the most troublesome aspects of surface editing. For example, self intersections cannot occur eliminating the need to detect and correct them as a post-process, and the topological genus of the surface can be changed trivially without consideration of explicit mesh connectivity. These features make the method extremely simple and efficient. A traditional drawback of implicit functions in regard to constructive solid geometry and surface editing is the inability to represent fine or sharp features, and the surface reconstruction technique of [51] attempts to identify these features and insert the appropriate geometry. This was improved upon by the dual contouring algorithm of [49] which uses Hermite data to reproduce the desired features without having to resort to feature identification metrics. Computer graphics researchers have also started to use level set methods to simulate water and and other liquids as introduced in [35, 29], as well as fire [64]. More recent works include adaptive techniques [56], solid/fluid coupling [42], vortex methods for rough water [78], and methods addressing control [76]. These techniques have been used in a number of recent films including mud for “Shrek” [35], liquid terminators in “Terminator 3” [76], wine in “Pirates of the Carribean” [76], the tar monster in “Scooby Doo 2” [101], water in “The Day After Tomorrow” [45], the fish bowl in “The Cat and the Hat” [23], etc.

There are countless other interesting applications of level set methods, most of which are beyond the scope of this article. In particular, several attempts have been made at extending the method to handle objects of higher codimension [9, 48, 17], there has been interesting work on inverse problems [10], and a few authors have combined level sets with finite element methods in order to accurately describe cracks and fracture [100, 60]. We refer the interested reader to the recent book [67] and the references therein.

2 Level Set Methods

Given a velocity field $\vec{u}(\vec{x})$ and a level set function $\phi(\vec{x})$, we evolve the interface forward in time according equation 1. To accomplish this, $\vec{u}(\vec{x})$ needs to be defined (at least) in a band about the interface. In many instances, such as two-phase incompressible flow, the velocity may already be defined throughout the domain. However, in other cases such as free surface flows where the velocity is defined on one side of the interface only, or Stefan problems where the jump conditions that lead to the interface velocity are defined only at the interface itself, one needs to extrapolate the velocity field to fill a band of a few grid cells on both sides of the interface. [105] demonstrated that a signed distance function tends to remain a signed distance function if the velocity at each point is defined to be equal to the velocity at the closest interface point, and a number of authors have proposed one-way and two-way extrapolation techniques for accomplishing this, see e.g. [16, 33, 2].

There are countless methods for evolving $\phi(\vec{x})$ forward in time, and the simplest scheme consists of either forward Euler time integration with spatial upwinding or a fully multidimensional method of characteristics semi-Lagrangian (see e.g. [84]) approach. The first approach has a CFL time step restriction of $\triangle t < \triangle x / \max |u|$, while the second is unconditionally stable. Unfortunately, both methods suffer from rather severe numerical dissipation resulting in significant mass or characteristic information loss, and a significant portion of the level set research has been dedicated towards methods for alleviating this difficulty. In fact, this has been the major criticism levied against level set methods for incompressible flow. In the following we discuss some of these remedies including higher order accurate finite difference schemes, preservation of signed distance functions, the coupled level set VOF method, the particle level set method, and adaptive grid techniques.

2.1 Higher Order Accuracy

In order to alleviate numerical dissipation for hyperbolic conservation laws, [44] introduced higher order accurate numerical methods based on essentially nonoscillatory (ENO) polynomial interpolation of data. These methods were extended to a finite difference framework in [81, 82] where the numerical flux functions were calculated directly from a divided difference table of the pointwise data. [81, 82] also introduced the notion of total variation diminishing Runge-Kutta (TVD-RK). [69] extended the ENO method to Hamilton-Jacobi equations using the fact that Hamilton-Jacobi equations in one spatial dimension are integrals of conservation laws. This work was generalized in [70].

ENO methods choose the smoothest possible stencil out of the potential candidates, e.g. for third order ENO the smoothest of the three possible candidates is chosen. [53] extended the ENO schemes of [81, 82] proposing to take a convex combination of all possible stencils weighted by relative smoothness. In the case of third order ENO, this allows one to obtain fourth order accuracy. The basic idea is to weight the scheme towards central differencing in smooth regions and towards ENO in regions with large gradients. Later, [47] showed that a better choice of weights could lead to improved results, for example in the third order ENO case one can obtain fifth order accuracy. WENO was extended to Hamilton-Jacobi equations in [46]. This was a significant improvement as it reduces the error by approximately an order of magnitude (for typical grid resolutions) when compared to the third order HJ-ENO method. When combined with third order TVD-RK, the HJ-WENO method proposed in [46] is considered to be state-of-the-art for evolving the level set equation.

2.2 Making Signed Distance Functions

Unfortunately, it turns out that higher order accurate approximations (i.e. ENO and WENO) of the level set equation are not enough. Certain flow fields can cause the level set function to generate large gradients polluting the accuracy of the finite difference approximations as shown in [61]. Thus, researchers have focused on methods to keep ϕ from developing large gradients. In fact, it turns out to be even more useful to force ϕ to approximate a signed distance function with $|\nabla\phi| = 1$.

2.2.1 Reinitialization

In the original paper [69], the level set function was initialized as $\phi = 1 \pm d^2$ where d is the distance to the interface and the "$\pm$" sign depends on which side of the interface a grid point is located on. This was later changed to be a signed distance function. After the advection of the interface, it is uncommon for the level set function to remain a signed distance function, which means that ϕ needs to be reinitialized at regular intervals in order to limit numerical dissipation. A simple and accurate technique is to calculate how far each grid point is from the zero isocontour of the level set directly. This technique is quite expensive in practice, and as such cannot be used in large real world examples or with schemes that require frequent reinitialization. A more practical alternative is to evolve the interface in the normal direction at unit speed keeping track of the amount of time that it takes for the interface to cross over each grid point. The crossing time gives the distance value for a grid point. If we evolve the interface in both the normal and negative normal direction at the same time, we obtain an equation of the form

$$\phi_\tau + S(\phi_0)(|\nabla\phi| - 1) = 0 \tag{2}$$

which is commonly known as the reinitialization equation, proposed in [92]. Here $S(\phi_0)$ is a smeared out sign function which is positive approaching 1 in Ω^+ and negative approaching -1 in Ω^-. Standard Hamilton-Jacobi solvers can be used on this equation including fifth order HJ-WENO and third order TVD-RK. One difficulty encountered when solving equation 2 is that the interface moves by a small amount due to numerical truncation error. In an attempt to alleviate this difficulty, [89, 90] proposed a reinitialization method that attempts to preserve the partial volume cut by the interface in each cell. Although this method was shown to improve the quality of the numerical results of HJ-ENO significantly, it does not have a significant effect on the quality of results obtained with the more accurate fifth order HJ-WENO.

2.2.2 Fast Marching Method

Instead of solving the partial differential equation 2 for a number of time steps in fictitious time τ, one can instead start at the interface and march outwards creating a signed distance function in a single sweep through the data. The first method of this type was proposed by [97, 98], which extended Dijkstra's method for computing the taxicab metric to an algorithm for computing Euclidean distance.

In order to apply this algorithm, one first needs to initialize all the grid points adjacent to the interface with a "correct" ϕ value. These nodes are then considered *done*, all their neighbors are considered *close* and all other nodes are considered far. Next, all grid nodes in the *close* set have tentative values of ϕ calculated using only values from *done* points. Then the smallest of these is added to the *done* set, all its neighbors are added to the *close* set, and the tentative values of these neighbors are calculated or updated using the fact that there is now one more node in the *done* set. The algorithm is terminated when a thick enough band of points around the interface has been added to the *done* set. The selection of the grid node that has the smallest tentative ϕ value is the slowest part of the algorithm, and it is common to use a heap in order to efficiently select this point.

Although this method is only first order accurate, it is straightforward to extend it to higher order accuracy as was done in [80] using an equivalent finite difference formulation of the method (i.e. from [79]). When using the higher order accurate version of the method, it is important to exercise more care when initializing the values adjacent to the interface. [22] proposed using higher order interpolants and Newton iteration to initialize these points more accurately. See also [37].

In order to illustrate the importance of accurately initializing the points adjacent to the interface, consider the following example. First, consider a lower order accurate method where each grid point that needs to be initialized searches in each Cartesian direction to see if the interface crosses the edge connecting it to its neighbors. If so, we use linear interpolation along the edge to find the location of the interface keeping only the closest intersection point in each Cartesian grid direction. This results in anywhere from one to three intersection points, and in the case of three points we pass a plane through them and use the closest point on the plane as the distance. In the case of two points we pass a line through them and calculate the closest point on the line, and in the case of one point we use the distance to it. Then we use the fast marching method to compute signed distance value for points in a band near the interface, and use a ray tracing algorithm to render the final result. The ray tracing algorithm uses a root finding algorithm to calculate the location of the zero level set, and then uses the normal calculated at that point for the shading. If we start with an analytic signed distance function for a sphere, the results obtained after applying the fast marching method ten times are shown in figure 1. Even though the interface moves by only a small amount, the grid induced aliasing errors are quite apparent.

In order to alleviate these problems, we propose finding the interface

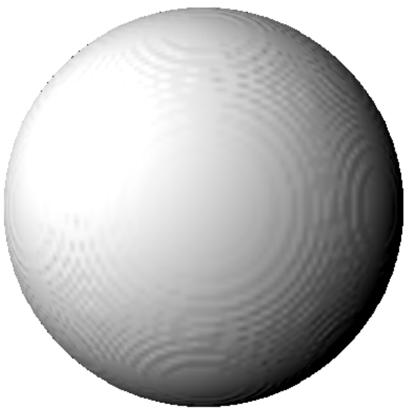
Figure 1: An analytic sphere after ten repeated applications of the fast marching method with standard initialization. Resolution is $100 \times 100 \times 100$.

with a simple unbiased search algorithm. We place a particle at the point in question, and then move it in the normal direction towards the interface by an amount equal to the absolute ϕ value at that point. Then we interpolate a new ϕ value, calculate a new normal at the new particle location, and move the particle towards the interface once again. We continue this process until the ϕ value at the particle location is within some tolerance of zero, and then use the distance from the final particle location to the original location as the distance value at the original point. If this process does not converge after a certain number of iterations or the distance value calculated is larger than a threshold, we use the cruder distance estimate given above. The threshold we use is given by the closest intersection point found using linear interpolation in the Cartesian grid directions, i.e. by the up to three points calculated in the cruder version of the algorithm. Across a wide range of numerical experiments, this algorithm seems to remove the typical "ringing artifacts" seen in figure 1. Figure 2 illustrates how this new method removes these artifacts.

2.3 Hybrid Methods

In the quest to prevent mass loss, both the level set evolution equation 1 and the reinitialization equation 2 can be discretized with fifth order accurate HJ-WENO schemes in space and third order accurate TVD-RK schemes in

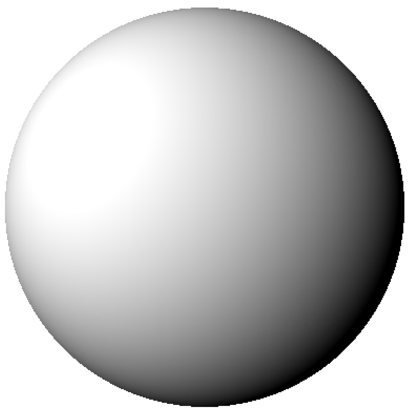

Figure 2: An analytic sphere after ten repeated applications of the fast marching method with modified initialization. Resolution is $100 \times 100 \times 100$.

time. Unfortunately, for many applications, this is still not enough accuracy. For example, figure 3 (reprinted from [27]) shows the "Enright test" where a sphere is evolved in an incompressible flow field which contains a temporal term that reverses the velocity half way through the calculation so that the final state should be identical to the initial state. One can see that about eighty percent of the mass is lost even with the use of fifth order accurate HJ-WENO and third order accurate TVD-RK on both level set evolution and reinitialization. Thus, researchers looked for other methods for improving mass loss as opposed to pursuing even higher order accurate schemes. A degree of success was achieved by hybridizing the level set method with other interface tracking methods, i.e. the VOF method and Lagrangian particle tracking.

2.3.1 Coupled Level Set Volume of Fluid Method

In [91], an algorithm for combining the level set method with the VOF method was proposed, and the results obtained were superior to those obtained by using either method on its own. In the VOF method, a volume fraction variable F is defined in each cell representing the fraction of the cell occupied by one of the fluids. It takes on a value of 1 in one fluid, 0 in the other, and intermediate values near the interface. The volume fraction is evolved in time using a conservative flux based (or equivalent) discretization

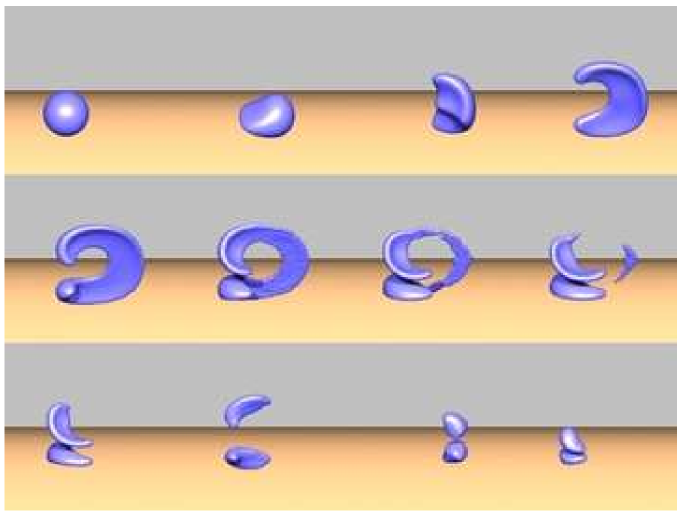

Figure 3: Even with higher order accurate numerical methods, the level set method can exhibit large amounts of mass loss. Here about eighty percent of the mass is nonphysically lost in an incompressible flow field. Reprinted from [27].

of the conservation law for volume valid for incompressible velocity fields. If truncation error dictates a value of F that is not between 0 and 1, F is clamped to this range losing a negligible amount of mass in the process. This small amount of mass loss is typically an order of magnitude or more less than that lost in level set methods and is typically not an issue. After advection, a piecewise linear interface representation is constructed by first calculating the gradient of F in each interfacial cell to find a normal direction, and then adjusting the plane associated with this normal until it cuts the cell in a manner that agrees with the local value of F. Note that this leads to an interface representation that is discontinuous between cells, and this discontinuity is one of the major drawbacks of the VOF method hindering its ability to robustly and accurately calculate geometric information. It also leads to small pieces of fluid called flotsam and jetsam nonphysically breaking off from the interface moving around with erroneous velocities. For more on the VOF method, see the recent [73].

In [91], the piecewise linear interface is used to initialize a new signed distance function ϕ at each time step. Then both F and ϕ are advected forward in time, and the values of ϕ are used to more accurately construct a gradient and normal for use in the piecewise linear interface reconstruction. Moreover, in cells with $|\phi| > \Delta x$, one can assume that there is no interface present and clamp F to 0 or 1 (appropriately) regardless of the current value of F minimizing the appearance of flotsam and jetsam, but creating more mass loss than a standard VOF method. As an added bonus, ϕ can be used to calculate geometric information such as curvature in a more efficient and robust fashion than either using F or the discontinuous piecewise linear interface representation. Although this method does obtain improved results over both the level set only method and the VOF only method, the reconstructed interface appears noisy and lacks time coherence as can be seen in [58]. Moreover, the only way to remove unsightly and inaccurate flotsam and jetsam is to delete it from the calculation nonphysically removing mass. Finally, the advection of both ϕ and F, the piecewise linear interface reconstruction, and the subsequent reconstruction of ϕ based on this piecewise linear interface are all quite complicated and do not make the best candidates for extension to adaptive and/or unstructured grids.

2.3.2 Particle Level Set Method

The particle level set (PLS) method was introduced in [27] as a numerical scheme that combines the accuracy benefits of Lagrangian front tracking with the simplicity and efficiency of the level set method. Lagrangian marker particles are passively advected with the flow and used to rebuild the level set function in under-resolved regions where mass or information is lost. One of the main benefits of the PLS method is that the reconstructed interface still enjoys all the nice properties of the level set method including continuity and smoothness (especially as opposed to VOF methods), robust geometry information, temporal coherence, etc. Also, since the particles are not topologically connected, the method does not suffer from the complexity associated with standard front tracking methods, while still maintaining the main benefit of accurately tracked particle locations. The general idea of the particle level set method is to randomly seed two sets of massless marker particles in a band near the interface. "Positive" particles are placed on the $\phi > 0$ side, and "negative" particles are placed on the $\phi < 0$ side. The particles are passively advected using velocities interpolated trilinearly to the particle location and at least second order accurate Runge-Kutta time integration.

When seeded, each particle is given a radius equal to its distance from the interface clamped by some minimum and maximum values. Each particle is thought to define a level set of its own, identically zero on the surface of a sphere given by its radius. For positive particles, the interior of the

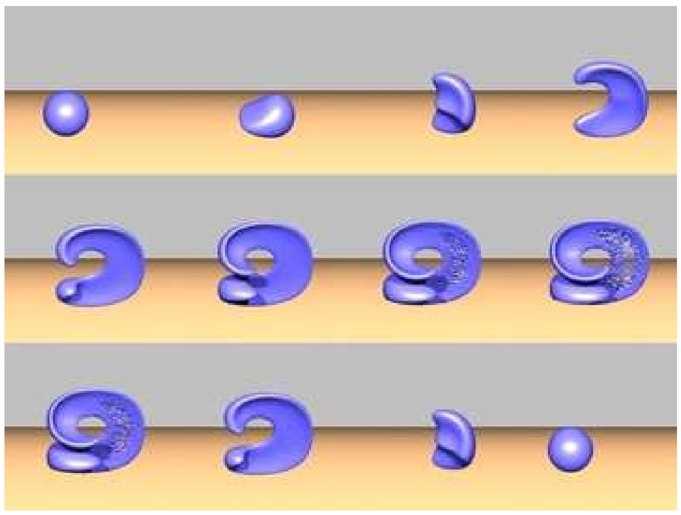

Figure 4: The particle level set method does a much better job of preserving mass even when lower order accurate finite difference formula are used. Here only about one percent of the mass is lost. Reprinted from [27].

sphere is positive and the exterior is negative. This is reversed for negative particles. After advecting both the particles and the level set, these particle spheres are used to correct the level set values. First, one independently constructs corrected level set functions ϕ^+ and ϕ^- corresponding to ϕ corrected by either positive or negative particles respectively. The corrections are done in a node by node fashion considering all the particles near the node and the value of ϕ dictated by the sphere around each particle. When constructing ϕ^+, the nodal value of ϕ is taken to be the largest potential value considering all nearby positive particle spheres and the current ϕ value at the node. ϕ^- is constructed independently with negative particles and the minimum potential value. Then the two corrected level set functions ϕ^+ and ϕ^- are resolved into a single level set by choosing the one with minimum magnitude at each grid point. These same types of corrections are applied to reinitialization as well, except that the particle velocity is considered to be zero for the reinitialization step. After reinitialization, the level set values are used to modify the particle radii so that there is a closed feedback loop in the process. Figure 4 shows the "Enright test" using the particle level set method. Note the extreme improvement in the computed result where the mass loss has been lowered from about eighty percent to around one percent.

Recently, [28] showed that the particle level set method does not necessitate high order accurate methods for the level set advection and reinitialization. In fact, they showed that basic first order accurate semi-Lagrangian advection was adequate for level set evolution while the first order accurate fast marching method could be used for reinitialization with almost no reduction in accuracy. The only modification that seemed to adversely affect the accuracy of the method was the reduction of the particle time integration from second to first order accuracy. Thus, it seems that the particle level set method relies on the particles for accuracy and the level set for connectivity truly combining the best of both methods as desired. Moreover, the ability to use simple semi-Lagrangian advection with no CFL restriction along with the fast marching method makes the particle level set method a good candidate for both adaptive and unstructured grids.

2.4 Spatially Adaptive Methods

Typically, one only needs level set values near the interface and this has led a number of researchers to optimize their calculations by only updating ϕ in a band near the interface. See for example [21, 1, 71]. An interesting method not considered by these authors would be to first use the fast marching method to obtain a temporary value of the level set $\hat{\phi}$ in a band slightly larger than required, and then solve the reinitialization equation using the *unmodified* ϕ values in the band of interest and the fixed $\hat{\phi}$ values as an outer boundary condition.

While all these methods reduce the computational time required, they still require an inordinate amount of memory. In an attempt to alleviate this, truly adaptive techniques are required. Patch based adaptive mesh refinement (AMR) techniques [7, 6] are prevalent, and [88] used patch based AMR in the context of two-phase incompressible flow. However, it is limiting to restrict the adaptivity to block structured meshes as is made clear by the following quote "The obvious first choice, suggested by the nested hierarchical nature of the grid itself, is to use an octree in 3D or quadtree in 2D." from [3].

2.4.1 Octree Based Methods

[86, 85] proposed a number of techniques for moving interfaces on tree based data structures. Quadtree meshes resolve the interface with almost optimal efficiency since the computational complexity of advecting and reinitializing the interface is $O(N \log N)$. This is achieved by refining near the interface

only, although using a graded tree with adjacent cells differing by at most a factor of two is a common practice due to its simplicity. T-junction nodes are nodes that do not have six (four) adjacent edges in three (two) spatial dimensions, i.e. nodes that lie on the face (edge) of a coarse cell. [85] explores the use of domain triangulations in order to ensure continuous interpolation near T-junctions. Although a continuous interpolant can be formed by triangulating the cells of the tree, this comes at the steep price of having to perform the triangulation.

[56] introduced a new technique for discretizing the particle level set method on an unrestricted octree mesh in the context of free surface flows. As noted above, the particle level set method only requires first order accurate semi-Lagrangian advection and the fast marching method for maintaining a signed distance function. The level set function and the velocity field are stored on the nodes of the octree allowing for highly efficient interpolation of variables. To avoid difficulties near T-junction nodes, these nodes were constrained to ensure continuous interpolation across cell boundaries. Although the octree particle level set method places no restrictions on the octree in terms of refinement levels, it is often beneficial to refine maximally near the zero isocontour to achieve the greatest amount of accuracy. In fact, if we restrict the octree to be finer near the interface and coarser away from it, then the T-junctions will tend to have missing nodes in the directions *away* from the interface where these nodes are not needed. That is, the fast marching method can be readily applied at T-junctions simply by ignoring the edge directions that do not exist. Moreover, when accurate level set values are only required near the interface, maximally refining in a band about the interface eliminates the presence of T-junctions there.

An *efficient* implementation of algorithms on octrees can be tricky. There are several ways of implementing any given operation, and the naive approach can be orders of magnitude slower and more difficult to implement than a less obvious approach. The terminology that we use is as follows. An octree cell has eight nodes (one at each corner), and six faces. A minimal cell is one that does not contain any children. A minimal face is one that does not contain any smaller faces within it, i.e. both cells touching the face are minimal. A minimal edge is one that does not contain any other smaller edges, i.e. all cells touching the edge are minimal. Using iterators to enumerate every minimal cell, face, edge or node in the tree greatly simplifies implementation. [49] introduced a tree traversal algorithm that efficiently enumerates every minimal edge in an octree using a number of recursive functions. This algorithm can be extended to enumerate every node in the tree simply by adding one more recursive function (*nodeProc()* in their terminology). This function takes eight possibly non-unique cells and recursively traverses the cells to find the leaf cells that the node is touching. These iterator functions are readily implemented to map a general function on every minimal cell, face, edge or node in the octree.

The operation that typically accounts for most of the computational cost in a particle level set implementation on an octree is finding the leaf cell that contains a given point. The naive implementation of this operation starts at the root cell and recursively traverses the tree finding which of the 8 child cells the point lies in. The computational complexity of this operation is $O(\log N)$ as opposed to $O(1)$ for regular array based lookups. There are several ways in which this lookup can be significantly improved. At the coarsest level we use a uniform block structured grid as opposed to a single cell, and then each cell of this grid contains an octree of its own. This type of implementation also allows the cell proportions to be decoupled from the overall domain size, since the uniform base grid can be of any dimension. The computational savings from not having to traverse the tree from a single root every time a lookup is required are significant. Several of the initial levels are skipped by performing an $O(1)$ uniform grid access followed by a significantly shorter tree traversal. For example, for simulations that are running at 1024^3 effective resolution at the finest level (octree of depth eleven), we typically use a uniform grid of 64^3 saving seven levels for every lookup. With this octree topology, leaf cell lookups are a factor of two faster in our experience.

The most common operation that is performed on the octree is the interpolation of nodal data to an arbitrary location in space. This is accomplished by finding the leaf cell that contains the point in question, retrieving the eight values for the nodes of the cell, and then trilinearly interpolating between them. The expensive part of this operation is finding the leaf cell as described above, but can be made significantly cheaper by exploiting locality noting that semi-Lagrangian advection involves finding the interpolated value at a location close to the grid node being updated. The modified version of the iterator described in [49] describes a node by the eight cells that touch it (some cells may be duplicated in the case of nodes that lie on the transition between cells of different sizes). One of these eight cells will usually contain the semi-Lagrangian sample location, and a full tree traversal may not be necessary in order to find the leaf cell. In our experience, this optimization decreases the computation time by more than an order of magnitude in a typical octree particle level set simulation. When the sample point is not in one of the cells directly touching the node, we recommend using a neighbor traversal algorithm or a precomputed neighbor

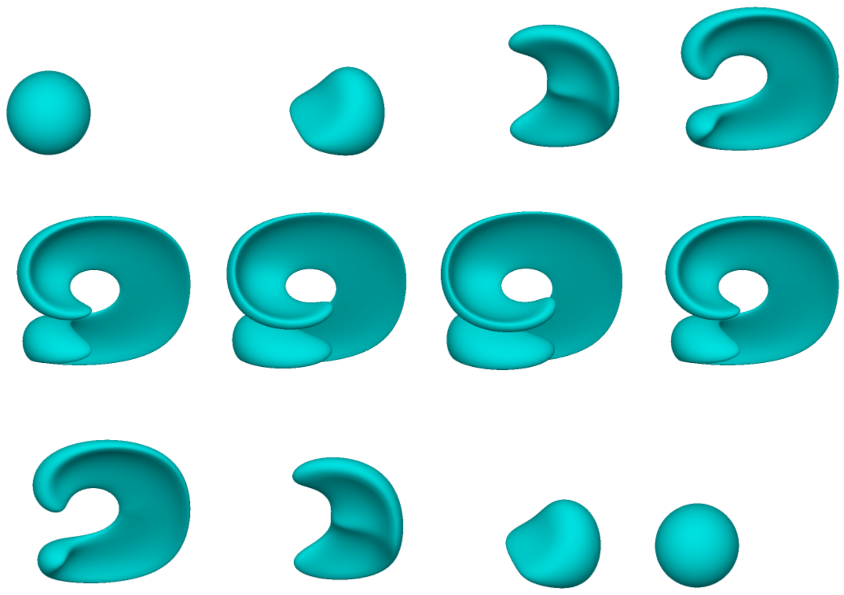

Figure 5: Semi-Lagrangian particle level set deformation test on an unrestricted octree grid. Maximum resolution by the interface is 512^3.

array in order to find the cell in question. This technique is frequently used in the case of particle advection where particles typically move about one grid cell per time step. Figure 5 shows the results of the “Enright test” on an unrestricted octree with semi-Lagrangian level set advection, the first order accurate fast marching method and second order particle advection.

We note that the octree approach can also be used for simulating objects in higher than three spatial dimensions and for simulating objects in higher codimension. See [59] for work in this direction.

3 Incompressible Flow

The incompressible flow equations are derived from the conservation of mass, momentum and energy using the divergence free condition $\nabla \cdot \vec{u} = 0$ which implies that the material in the flow does not compress or expand. Incompressible flow approximates many liquids that we interact with in everyday life and is of great interest in computational physics, computer graphics and many other fields. Level set methods can be used to track the interface between two incompressible fluids such as water and air. Each fluid is simulated with the Navier-Stokes equations and variables such as viscosity, density and pressure may be discontinuous across the interface. In the presence of discontinuities, finite differencing leads to $O(1/\Delta x)$ errors that increase with grid refinement. Thus, the boundary conditions at the interface are treated by either smearing discontinuous variables across the interface or by treating the discontinuity directly in a sharp fashion.

[72] introduced the "immersed boundary" method in order to simulate an elastic membrane immersed in an incompressible fluid flow. The main idea behind the method is to use a delta function to smear out the numerical solution about the immersed interface. This method was extended to a level set formulation of two-phase flow in [92] (see also [13]), which used the level set function to smear out the density and viscosity variables in a thin band about the interface, e.g.

$$\rho(\phi) = \rho^- + (\rho^+ - \rho^-)H(\phi) \tag{3}$$

where ρ^+ and ρ^- are the discontinuous densities and $H(\phi)$ is a smeared out Heaviside function. If similar smearing is applied to the pressure, one loses the ability to model surface tension via $[p] = \sigma\kappa$. This was remedied in both VOF methods [8] and front tracking methods [99] by adding a term to the momentum equations. In the context of level set methods, [92] added this term as

$$\frac{\delta(\phi)\sigma\kappa\vec{N}}{\rho} \tag{4}$$

where $\delta(\phi)$ is a smeared out delta function, $\vec{N}$ is the local unit normal, κ is the curvature and σ is a constant. [95] recently discovered that level set methods can suffer from $O(1)$ errors with the typical delta function approach, and thus caution should be exercised when following this methodology.

An alternative to smearing out the variables across the interface is to use a variant of the ghost fluid method [33], which treats the discontinuities across the interface directly. The ghost fluid method was extended to the variable coefficient poisson equation in [54] allowing one to solve

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p\right) = f \tag{5}$$

with jump conditions of $[p] = g$ and $[(1/\rho)\nabla p \cdot \vec{N}] = h$ given. The method preserves sharp discontinuities in both p and ∇p across the interface. Later,

[50] illustrated that this technique is suitable for solving two-phase incompressible flow without smearing the density or pressure values across the interface. Moreover, the extra delta function forcing term from equation 4 is not required, since the pressure jump is modeled directly.

3.1 Free Surface Flow

Free surface flow assumes that the motion of one fluid completely dominates the other, and thus this second fluid can be simulated as a constant pressure fluid that exerts no other stress on the interface. A typical example would be a large body of water interacting with air. An early version of free surface flow was proposed by [43] which used marker particles to track the water. This work was improved upon by [75] which proposed a subcell treatment of the pressure boundary condition, and [14] which proposed improved velocity boundary conditions at the free surface. Later, [15] proposed a method which only required tracking particles near the interface. We also note that [39] devised a technique that allows one to apply a second order accurate pressure boundary condition without changing the symmetric nature of the discretization, see [30].

[35] was the first to propose using level set methods with free surface flow, and was also the first to couple particle methods to level set methods. However, they only placed particles on the water side of the interface as opposed to [27] which placed particles on both sides of the interface. Figures 6 and 7, reprinted from [29], show the quality of results attainable using the particle level set method with free surface flow. Here, the velocity boundary condition at the interface was obtained by extrapolating the velocity from the water into the air using the level set normals to determine the extrapolation direction. [87] proposed projecting this velocity to be divergence free after extrapolating it. Further extensions include the incorporation of surface tension effects in [30].

3.2 Free Surface Flow on Octrees

Methods for octrees are not yet common. For example, [74] claims to have the first single-phase incompressible flow solver based on an octree data structure. [74] used a standard splitting procedure where first an intermediate velocity field $\vec{u}^*$ is computed ignoring the pressure term, then the pressure is solved for via

$$\nabla^2 p = \nabla \cdot \vec{u}^* / \Delta t \qquad (6)$$

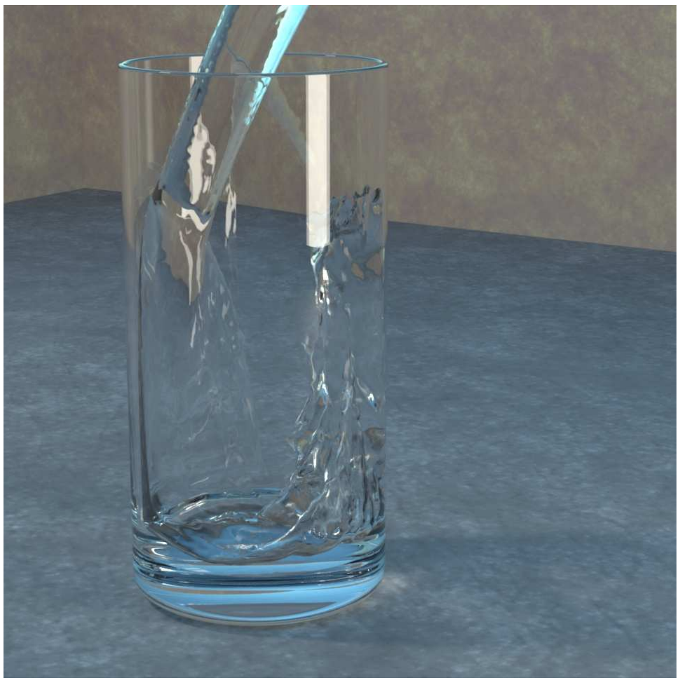
Figure 6: Simulation of a glass of water being poured. Reprinted from [29].

and used to project $\vec{u}^*$ to be divergence free,

$$\vec{u} = \vec{u}^* - \Delta t \nabla p. \tag{7}$$

A major drawback of [74] was that the proposed discretization of equation 6 resulted in a nonsymmetric coefficient matrix for the pressure. Thus, a multigrid solver was used to solve the linear system of equations for the pressure. Although multigrid methods are optimal asymptotically, the constant can be large when the solution possesses high frequencies. Interfaces, sharp angles or even thin bodies restrict the efficiency of methods that rely on multigrid solvers, see e.g. [24]. More recently, [55] noted that high frequency fields need to be smeared out in order to alleviate convergence problems, and pointed out that high order schemes have less favorable smoothing properties than low order schemes forcing them to use a combination of low and high order to achieve both reasonable accuracy and convergence. In the case of free surface flow, surface wave generation relies on horizontal pressure gradients caused by stacking different heights of high-density fluids. This behavior is excessively damped by density smearing causing increased artificial viscosity and an overly viscous flow.

Figure 7: Simulation of a glass of water being poured. Reprinted from [29].

[56] extended [74] proposing a novel symmetric discretization of equation 6 that enables the use of iterative solvers such as the preconditioned conjugate gradient (PCG) method. This both alleviates the need for nonphysical smoothing and allows one to model free surface flows without the difficulties associated with topological changes incurred during coarsening and refining domains when using multigrid methods on problems with interfaces.

Consider the discretization of equation 6 for a large cell with dimensions $\triangle x$, $\triangle y$ and $\triangle z$ neighboring small cells as depicted in figure 8. Since the discretization is closely related to the second vector form of Green's theorem that relates a volume integral to a surface integral, equation 6 is rescaled by the volume of the large cell to obtain $\mathrm{V}_{\mathrm{cell}}\triangle t\nabla^2 p = \mathrm{V}_{\mathrm{cell}}\nabla\cdot\vec{u}^*$. The right hand side now represents the quantity of mass flowing in and out of the large cell within a time step $\triangle t$ in m^3s^{-1}. This can be further rewritten as

$$V_{\mathrm{cell}}\nabla\cdot(\vec{u}^* - \triangle t\nabla p) = 0. \tag{8}$$

implying that the ∇p term is most naturally evaluated at the cell faces using the direct correspondence between the components of ∇p and $\vec{u}^*$. Moreover,

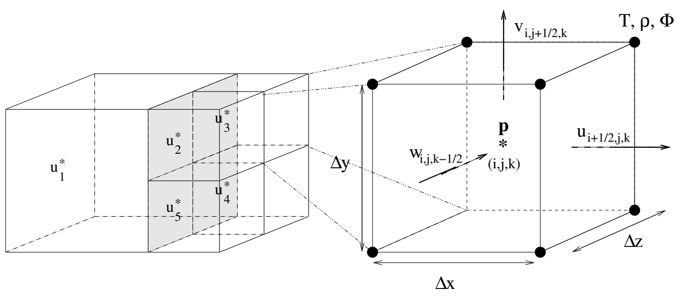

Figure 8: Depiction of a large cell neighboring four smaller cells. The u_i^* represent the x components of the intermediate velocity $\vec{u}^*$ defined at the cell faces. Right: a single computational cell. The velocity components are defined on the cell faces. The pressure is defined at the center of the cell. The density, temperature and level set function are stored at the nodes.

substituting equation 7 into equation 8 implies the desired divergence free condition

$$V_{\text{cell}} \nabla \cdot \vec{u} = 0. \tag{9}$$

In order to calculate the divergence of a cell, one can invoke the second vector form of Green's theorem to write

$$V_{\text{cell}} \nabla \cdot \vec{u}^* = \sum_{\text{faces}} (\vec{u}^*_{\text{face}} \cdot \vec{n})\, \text{A}_{\text{face}}, \tag{10}$$

where $\vec{n}$ is the outward unit normal of the large cell and A_{face} is the area of a cell face. In figure 8, the discretization of the x component of the divergence reads $\triangle x \triangle y \triangle z \partial u^*/\partial x = u_2^* \text{A}_2 + u_3^* \text{A}_3 + u_4^* \text{A}_4 + u_5^* \text{A}_5 - u_1^* \text{A}_1$, where the minus sign in front of $u_1^* \text{A}_1$ accounts for the fact that the unit normal points to the left. Thus, $\partial u^*/\partial x = ((u_2^* + u_3^* + u_4^* + u_5^*)/4 - u_1^*)/\triangle x$. The y and z directions are treated similarly.

Once, the divergence is computed at the cell center, equation 6 is used to construct a linear system of equations for the pressure. Invoking again the second vector form of Green's theorem, one can write

$$\text{V}_{\text{cell}} \nabla \cdot (\triangle t \nabla p) = \sum_{\text{faces}} ((\triangle t \nabla p)_{\text{face}} \cdot \vec{n})\, \text{A}_{\text{face}}. \tag{11}$$

Therefore, once the pressure gradient is computed at every face, we can carry out the computation in a manner similar to that of the velocity divergence above. There exist different choices in the discretization of $(\nabla p)_{\text{face}}$,

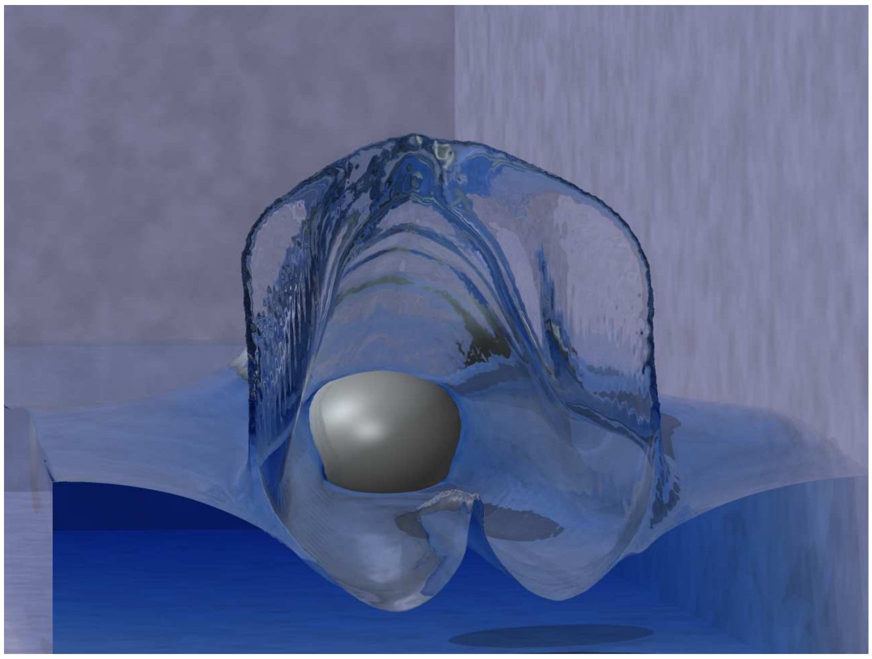

Figure 9: Simulation of an ellipse traveling through a shallow tank of water. Note how well the octree resolves the thin film. Reprinted from [56].

but only some will yield a symmetric discretization matrix. Efficient iterative methods such as PCG (see e.g. [77]) can be applied to symmetric positive definite matrices offering a significant advantage over methods for nonsymmetric linear systems. Moreover, since data access for the octree is not as convenient as for regular grids, there is a strong benefit in designing a discretization that leads to a symmetric linear system.

Consider the configuration in figure 10. In the case where two cells of the same size juxtapose each other, standard central differencing defines the pressure gradient at the face between them, as is the case for $p_y = (p_{10} - p_1)/\triangle y$. Next, consider the discretization of the pressure gradient in the x direction at the face between cell 1 and cell 2. A standard approach is to first compute a weighted average value p_a by interpolating between p_1 and p_{10}. Then, since standard differencing of $\hat{p}_x = (p_2 - p_a)/(.75\triangle x)$ does not define $\hat{p}_x$ at the cell face, but midway between the locations of p_a and p_2, a more complex discretization is typically employed. For example, one can pass a quadratic interpolant through p_a, p_2 and p_6 and evaluate its derivative at the cell face, see e.g. [16]. However, this approach yields

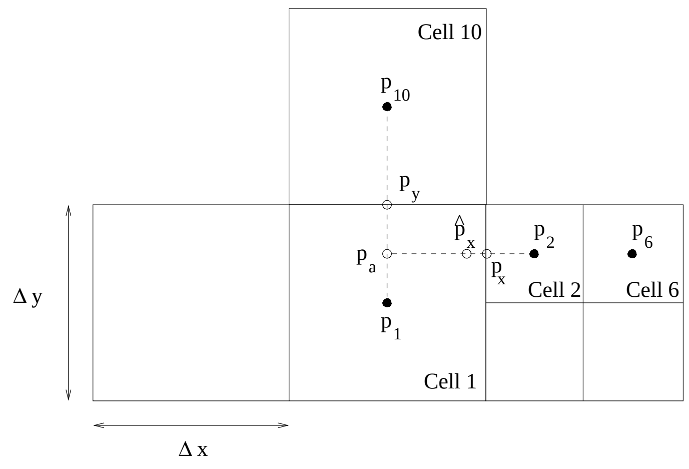

Figure 10: Discretization of the pressure gradient.

a nonsymmetric linear system that is computationally expensive to invert. The nonsymmetric nature of the linear system comes from the non-locality of the discretization, i.e. p_a depends on p_{10} and the quadratic interpolation depends on p_6. Consequently, the equation for cell 1 involves both p_{10} and p_6. It is unlikely that the equation for cell 6 will depend on p_1, since cell 6 juxtaposes another cell of the same size, namely cell 2. And even if it did, the coefficients of dependence would not be symmetric.

The situation can be improved by realizing that $O(\triangle x)$ perturbations in the pressure location still yield consistent approximations as in [39]. Therefore defining $p_x = (p_2 - p_a)/(.75\triangle x)$ *at* the cell face still yields a convergent approximation, since the location of $\hat{p}_x$ is perturbed by a small amount proportional to a grid cell. Moreover, we can avoid the dependence of p_a on values other than p_1 by simply setting $p_a = p_1$. This corresponds to an $O(\triangle x)$ perturbation of the location of p_1, and therefore still yields a convergent approximation. Thus, the discretization of p_x is simply $p_x = (p_2 - p_1)/(.75\triangle x)$. Moreover, since only p_1 and p_2 are considered, one can define $p_x = (p_2 - p_1)/\triangle$ where $\triangle$ can be defined as the size of the large cell, the size of the small cell, the Euclidean distance between p_1 and p_2, etc. Numerical test indicate that all of these choices converge.

In light of equation 11, p_x contributes to both row 1 and row 2 of the matrix representing the linear system of equations, since it is located at the cell face between cell 1 and cell 2. More precisely, the contribution to row 1

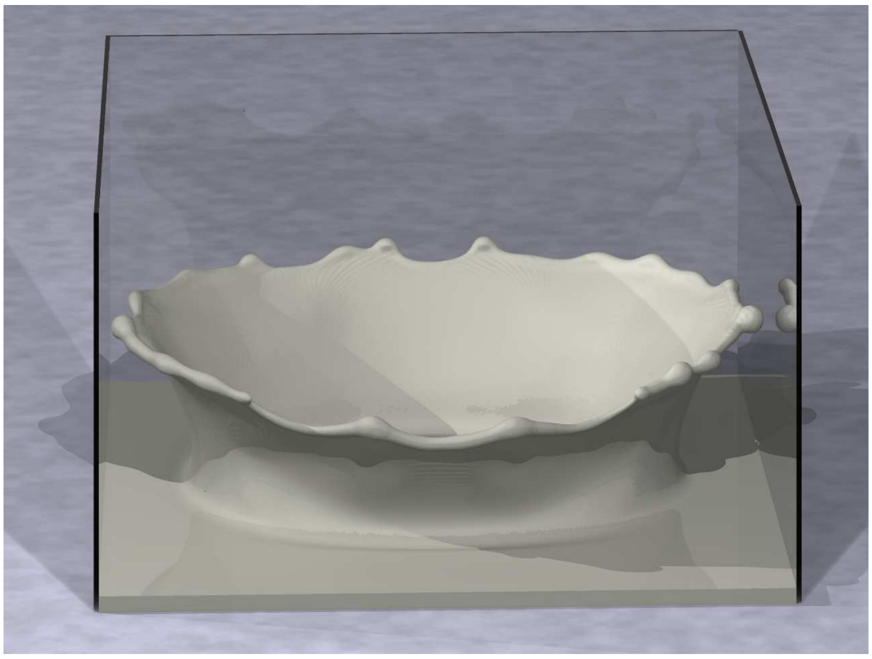

Figure 11: Simulation of a milk crown using the particle level set method on an octree data structure. Surface tension effects are incorporated as can be seen by the beads forming on the crown. Reprinted from [56].

occurs through the term

$$\triangle t p_x n_1 A_{\text{face}} = \triangle t \frac{p_2 - p_1}{\triangle}(1)A_{\text{face}},$$

since n_1, the x component of the outward normal to cell 1, points to the right (hence $n_1 = 1$). Likewise, the contribution to row 2 occurs through the term

$$\triangle t p_x n_1 A_{\text{face}} = \triangle t \frac{p_2 - p_1}{\triangle}(-1)A_{\text{face}},$$

since n_1, the x component of the outward normal to cell 2, points to the left (hence $n_1 = -1$). Therefore, the coefficient for p_2 in row 1 and the coefficient for p_1 for row 2 are identical, namely $\triangle t A_{\text{face}}/\triangle$. The same procedure is applied to all faces, and the discretization of the y and z components of the pressure gradient are carried out in a similar manner. Hence, the discretization yields a symmetric linear system that can be efficiently inverted with PCG.

Figure 9 shows how well the octree version of the particle level set method can resolve the thin films created as an ellipse slips through the water. Figure 11 shows similar results for the impact of a milk drop on a shallow layer of milk. In this calculation, the surface tension effects can be seen in the tips of the milk crown.

3.3 Second order Accuracy on Octrees

A potential difficulty with the method proposed in [56] is that it computes a different pressure gradient on each cell face, and also allows a different velocity on each face. The velocity on a face of a large cell is computed as the area weighted average of the velocities on the faces of all the adjacent small cells. Alternatively, in order to properly constrain the velocity field for interpolation, etc., it is desirable to have the velocity on the large cell face identical to all the velocities on the adjacent small cell faces. Area weighted averaging can still be used to compute the velocity on the large cell face, but this velocity then needs to be assigned to all the small cell faces as well. This also simplifies the treatment of Neumann (fixed velocity) boundary conditions, since an entire large cell face can be constrained as opposed to only portions of it.

If all the small cell faces incident on a large cell face have the same velocity, it is also important that they have the same pressure gradient when enforcing incompressibility so that they maintain the same velocity after being projected to be divergence free. Unfortunately, the method proposed by [56] yields different pressure gradients for these incident small cells. The pressure gradient on the large cell face is given by the area weighted average of the pressure gradients computed on each of the small cell faces, i.e.

$$(p_x)_L = \sum_s \frac{A_s}{A_L}\left(\frac{p_s - p_L}{\triangle_s}\right)$$

where $(p_x)_L$ is the pressure gradient on the large cell face, A_L is the area of the large cell face, p_L is the pressure in the large cell, the sum is over all incident small cell faces, A_s is the area of a small cell face, p_s is the pressure in a small cell, and $\triangle_s$ is the distance associated with a small cell face pressure gradient (e.g. half the size of the large cell plus the size of the associated small cell). To compute the pressure gradient flux into the large cell, $(p_x)_L$ is multiplied by A_L. For an incident small cell, [56] uses

$$(p_x)_s = \frac{p_s - p_L}{\triangle_s}$$

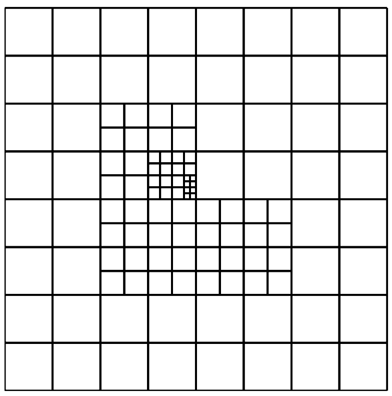
Figure 12: Initial quadtree refinement used in Poisson solver accuracy test.

as the pressure gradient and computes the flux through the small cell as $A_s(p_x)_s$. Instead, we propose using the pressure gradient of the large cell for all the small cells as well, i.e. setting $(p_x)_s = (p_x)_L$ so that the velocities of the small and large cells agree after the velocity field is projected to be divergence free.

For symmetry, first consider the coupling between the large cell and an incident small cell. When discretizing the large cell with $A_L(p_x)_L$, the coefficient of a small cell pressure is $A_s/\triangle_s$. When discretizing a particular small cell with $A_{s_o}(p_x)_L$, the coefficient of the large cell pressure is $-A_{s_o}/A_L \sum_s(-A_s/\triangle_s)$ where the extra minus sign comes from the fact that the normal points to the left (assuming the small cells are to the right of the large cell). If we use the same $\triangle_s$ for all incident small cells, e.g. we use an area weighted average

$$\triangle = \sum_s \frac{A_s}{A_L}\triangle_s$$

then $\triangle_s$ factors out of the sum, $\sum_s A_s$ is equal to and cancels out A_L, and we arrive at $A_{s_o}/\triangle$ yielding symmetry for the coupling between the large and small cells. We must also consider symmetry between pairs of small cells incident on the same large cell face, as this new scheme couples them together as well. When discretizing a particular small cell with $A_{s_o}(p_x)_L$, the coefficient of another small cell pressure is $-A_{s_o}A_s/(A_L\triangle)$. Similarly, the contribution cell s_o makes to another small cell via $A_s(p_x)_L$ is $-A_sA_{s_o}/(A_L\triangle)$ so the coupling is symmetric.

Although the discretization proposed in [56] is first order accurate, preliminary tests with this new discretization seem to indicate that it is second

L^1 error	order	L^∞ error	order
4.465×10^{-3}	$--$	5.012×10^{-3}	$--$
6.720×10^{-4}	2.73	9.777×10^{-4}	2.36
1.200×10^{-4}	2.49	2.050×10^{-4}	2.25
2.447×10^{-5}	2.29	4.626×10^{-5}	2.15
5.466×10^{-6}	2.16	1.092×10^{-5}	2.08
1.289×10^{-6}	2.09	2.648×10^{-6}	2.04

Table 1: Poisson solver accuracy on a quadtree grid.

L^1 error	order	L^∞ error	order
3.241×10^{-3}	$--$	6.990×10^{-3}	$--$
7.219×10^{-4}	2.17	1.596×10^{-3}	2.13
1.700×10^{-4}	2.08	3.841×10^{-4}	2.06
4.121×10^{-5}	2.04	9.460×10^{-5}	2.02
1.014×10^{-5}	2.02	2.354×10^{-5}	2.01

Table 2: Poisson solver accuracy on an octree grid.

order accurate. For example, consider $\nabla^2 p = e^x + e^y$ with an exact solution of $p = e^x + e^y$. We use the initial grid shown in Figure 12, and uniformly refine a number of times to obtain the results shown in Table 1. Obviously, the discretization is second order accurate in both the average and max norms. The method is second order accurate in three spatial dimensions as well, as illustrated in table 2 which demonstrates this for $\nabla^2 p = e^x + e^y + e^z$ with an exact solution of $p = e^x + e^y + e^z$.

Although we were at first surprised by the second order accurate numerical results, [52] and the references therein indicate the plausibility of this. In fact, it was [52] (and personal communications with the authors) that directly led to our first attempts to define unique pressure gradients on cell faces that could be used for both the large cell and *all* the small cells incident on a particular face.

Finally, we point out to the reader that uniformly refining an octree may not be the best way to numerically check accuracy. Repeated refinement of an octree eventually yields an octree where every large cell has face neighbors with uniform size, although the ratio between the large and small cells is no better or worse than that in the initial unrefined octree, i.e. it does not

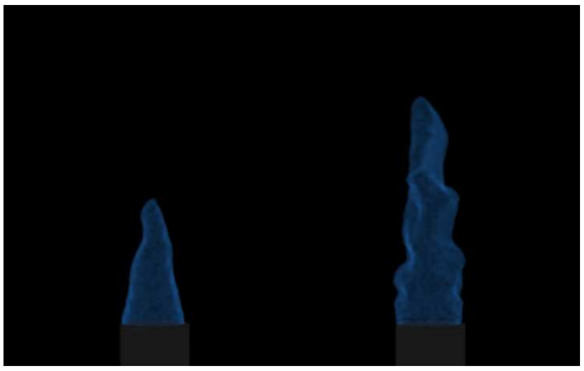

Figure 13: Blue reaction zone cores for large (left) and small (right) values of the flame reaction speed. Reprinted from [64].

become graded. To see this trivially, consider a large cell in the initial octree mesh next to any number of smaller cells. Once enough uniform refinements occur to subdivide the large cell into smaller cells at most the size of its smallest original neighbor, then all neighbors incident on a given face of a subdivided child of the large cell are trivially equal in size.

3.4 Low Speed Combustion

In the limit as the thickness of a reaction zone approaches zero, one achieves a thin flame approximation which is amenable to modeling with a level set function. The zero isocontour corresponds to the reaction front that separates the premixed fuel from the hot gaseous products, and the velocity of the flame front typically depends on both the local unit normal and curvature. [65] used the level set method and a variant of the ghost fluid method to simulate thin flame fronts that separate two incompressible fluids. The ghost fluid method is especially useful here, since the expansion across the flame front leads to a jump discontinuity in the normal component of the velocity, in addition to the jumps in density and pressure. Smearing out this velocity jump with a delta function type of formulation will generally lead to a loss of incompressibility near the interface. Figure 13 (reprinted from [64]) shows an example calculation where the zero level set is rendered as a blue flame core with fuel flowing upward into the flame front.

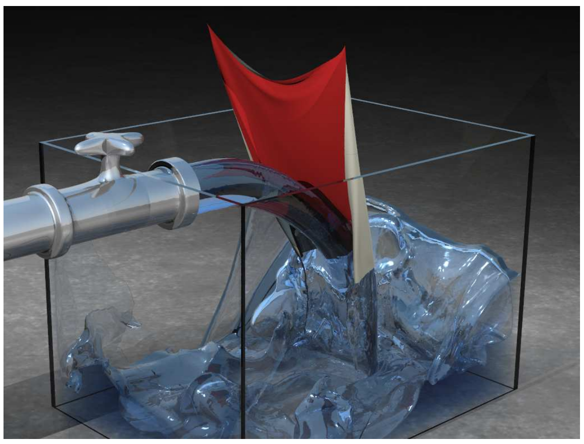

Figure 14: Thin films of water generated as water interacts with a thin deformable solid object. Reprinted from [42].

4 Conclusions

Level sets have become an established technique for interface tracking. The original level set method suffers from significant numerical dissipation, but through the use of higher order, hybrid and adaptive techniques, the method has become an incredibly powerful and accurate tool used in many application areas. Octree based spatial discretization schemes are nearly optimal, and recent work by [74], [56] and others have demonstrated that efficient octree based incompressible flow solvers are feasible, even in the context of complex objects and free surface flows.

There are may promising directions for future work, e.g. figure 14 shows thin films of water generated as water interacts with a thin deformable solid object. The strength of the method proposed in [42] lies in the fact that the solid material can be modeled with any black box method (e.g. a finite element discretization of a triangle mesh) independent of the solid/fluid coupling algorithm. Moreover, the newly proposed method works even for very thin objects that are too thin to be resolved on the background fluid simulation grid. See figure 15 that illustrates the ability of the method to prevent water from leaking through the thin deformable solid.

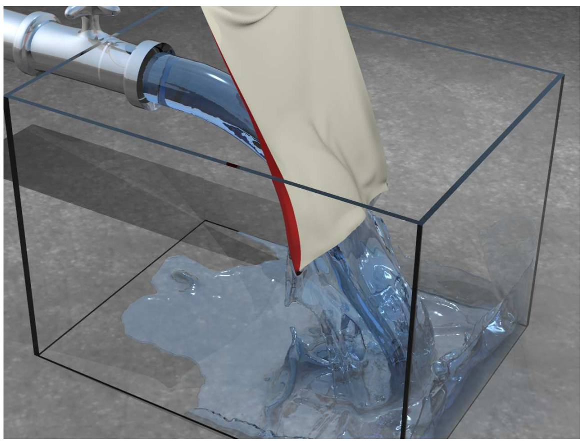

Figure 15: Water does not leak through the thin deformable solid even though it is too coarse to be represented on the background fluid simulation grid. Reprinted from [42].

References

[1] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118:269–277, 1995.

[2] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148:2–22, 1999.

[3] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Adaptive Cartesian Mesh Generation. In *CRC Handbook of Mesh Generation (Contributed Chapter)*, 1998.

[4] M. Aivazis, W. Goddard, D. Meiron, M. Ortiz, J. Pool, and J. Shepherd. A virtual test facility for simulating the dynamic response of materials. *Comput. in Sci. and Eng.*, 2:42–53, 2000.

[5] M. Arienti, P. Hung, E. Morano, and J. Shepherd. A Level Set Approach to Eulerian-Lagrangian Coupling. *J. Comput. Phys.*, 185:213–251, 2003.

[6] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[7] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[8] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modelling surface tension. *J. Comput. Phys.*, 100:335–353, 1992.

[9] P. Burchard, L-T. Cheng, B. Merriman, and S. Osher. Motion of curves in three spatial dimensions using a level set approach. *J. Comput. Phys.*, 170:720–741, 2001.

[10] M. Burger and S. Osher. A survey on level set methods for inverse problems and optimal design. In *CAM report (04-02), (in press)*, 2004.

[11] R. Caiden, R. Fedkiw, and C. Anderson. A Numerical Method for Two Phase Flow Consisting of Separate Compressible and Incompressible Regions. *J. Comput. Phys.*, 166:1–27, 2001.

[12] T. Chan and L. Vese. Active contour without edges. *IEEE Trans. on Image Processing*, 10:266–277, 2001.

[13] Y.-C. Chang, T. Hou, B. Merriman, and S. Osher. Eulerian capturing methods based on a level set formulation for incompressible fluid interfaces. *J. Comput. Phys.*, 124:449–464, 1996.

[14] S. Chen, D. Johnson, and P. Raad. Velocity boundary conditions for the simulation of free surface fluid flow. *J. Comput. Phys.*, 116:262–276, 1995.

[15] S. Chen, D. Johnson, P. Raad, and D. Fadda. A simple level set method for solving stefan problems. *Int. J. for Num. Meth.*, 25:749–778, 1997.

[16] S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving Stefan problems. *J. Comput. Phys.*, 135:8–29, 1997.

[17] L-T. Cheng, H. Liu, and S. Osher. Computational high frequency propagation using the level set method with applications to the semi-classical limit of the Schrodinger equation. *Comm. Math Sci.*, 1(3):593–621, 2003.

[18] J. Chessa and T. Belytschko. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. *Int. J. Numer. Meth. Engng.*, 58:2041–2064, 2003.

[19] J. Chessa and T. Belytschko. An extended finite element method for two-phase fluids. *ASME J. of Appl. Mech.*, 70:10–17, 2003.

[20] D.-I. Choi, J. D. Brown, B. Imbiriba, J. Centrella, and P. MacNeice. Interface conditions for wave propagation through mesh refinement boundaries. *J. Comput. Phys.*, 193:398–425, 2004.

[21] D. Chopp. Computing minimal surfaces via level set curvature flow. *J. Comput. Phys.*, 106:77–91, 1993.

[22] D. Chopp. Some improvements of the fast marching method. *SIAM J. Sci. Comput.*, 223:230–244, 2001.

[23] J. M. Cohen and M. J. Molemaker. Practical simulation of surface tension flows. In *SIGGRAPH 2004 Sketches & Applications*. ACM Press, 2004.

[24] M. Day, P. Colella, M. Lijewski, C. Rendleman, and D. Marcus. Embedded boundary algorithms for solving the poisson equation on complex domains. Technical report, Lawrence Berkeley National Laboratory (LBNL-41811), 1998.

[25] A. Dervieux and F. Thomasset. A finite element method for the simulation of a Rayleigh-Taylor instability. *Lecture Notes in Math.*, 771:145–158, 1979.

[26] A. Dervieux and F. Thomasset. Multifluid incompressible flows by a finite element method. *Lecture Notes in Phys.*, 141:158–163, 1981.

[27] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183:83–116, 2002.

[28] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83:479–490, 2005.

[29] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[30] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.*, number FEDSM2003–45144. ASME, 2003.

[31] S. Esedoglu and R. Tsai. Threshold dynamics for the piecewise constant mumford-shah functional. *CAM report (04-63)*, 2004.

[32] R. Fedkiw. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224, 2002.

[33] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[34] R. Fedkiw, T. Aslam, and S. Xu. The Ghost Fluid Method for Deflagration and Detonation Discontinuities. *J. Comput. Phys.*, 154:393–427, 1999.

[35] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001*, pages 23–30, 2001.

[36] F. Gibou and R. Fedkiw. A fast hybrid k-means level set algorithm for segmentation. 4^{th} *International Conf. on Stat., Math. and Related Fields, Honolulu (2005). Stanford Technical Report, November*, 2002.

[37] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *J. Comput. Phys.*, 202:577–601, 2005.

[38] F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19:183–199, 2003.

[39] F. Gibou, R. Fedkiw, L.-T. Cheng, and M. Kang. A second–order–accurate symmetric discretization of the poisson equation on irregular domains. *J. Comput. Phys.*, 176:205–227, 2002.

[40] J. Glimm, J. W. Grove, X. L. Li, and N.Zhao. Simple front tracking. *Contemporary Math.*, 238:133–149, 1999.

[41] D. Greaves. A quadtree adaptive method for simulating fluid flows with moving interfaces. *J. Comput. Phys.*, 194:35–56, 2004.

[42] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24, 2005.

[43] F. Harlow and J. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8:2182–2189, 1965.

[44] A. Harten, B. Enquist, S. Osher, and S Chakravarthy. Uniformly high-order accurate essentially non-oscillatory schemes III. *J. Comput. Phys.*, 71:231–303, 1987.

[45] J. Iversen and R. Sakaguchi. Growing up with fluid simulation on "The Day After Tomorrow". In *SIGGRAPH 2004 Sketches & Applications.* ACM Press, 2004.

[46] G.-S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[47] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126:202–228, 1996.

[48] S. Jin and S. Osher. A level set method for computing multivalued solutions to quasi-linear hyperbolic equations and Hamilton-Jacobi equations. *Comm. Math Sci.*, 1(3):575–591, 2003.

[49] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):339–346, 2002.

[50] M. Kang, R. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput.*, 15:323–360, 2000.

[51] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proc. of ACM SIGGRAPH 2001*, pages 56–66, 2001.

[52] K. Lipnikov, J. Morel, and M. Shashkov. Mimetic finite difference methods for diffusion equations on non-orthogonal non-conformal meshes. *J. Comput. Phys.*, 199(2):589–597, 2004.

[53] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 126:202–212, 1996.

[54] X.D. Liu, R. Fedkiw, and M. Kang. A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains. *J. Comput. Phys.*, 154:151, 2000.

[55] D. Lörstad and Laszlo Fuchs. High-order surface tension vof-model for 3d bubble flows with high density ratio. *J. Comput. Phys.*, 200:152–176, 2004.

[56] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:457–462, 2004.

[57] G. Markstein. *Nonsteady Flame Propagation.* Pergamon Press, 1964.

[58] V. Mihalef, D. Metaxas, and M. Sussman. Animation and control of breaking waves. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 315–324, 2004.

[59] C. Min. Local level set method in high dimension and codimension. *J. Comput. Phys.*, 200:368–382, 2004.

[60] N. Molino, J. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:385–392, 2004.

[61] W. Mulder, S. Osher, and J. Sethian. Computing interface motion in compressible gas dynamics. *J. Comput. Phys.*, 100:209–228, 1992.

[62] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Comm. in Pure and Appl. Math.*, 42:577–685, 1989.

[63] K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):330–338, 2002.

[64] D. Nguyen, R. Fedkiw, and H. Jensen. Physically based modeling and animation of fire. In *ACM Trans. Graph. (SIGGRAPH Proc.)*, volume 29, pages 721–728, 2002.

[65] D. Nguyen, R. Fedkiw, and M. Kang. A boundary condition capturing method for incompressible flame discontinuities. *J. Comput. Phys.*, 172:71–98, 2001.

[66] D. Nguyen, F. Gibou, and R. Fedkiw. A fully conservative ghost fluid method and stiff detonation waves. In *12th Int. Detonation Symposium, San Diego, CA*, 2002.

[67] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY.

[68] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer-Verlag, 2003. New York, NY.

[69] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[70] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:902–921, 1991.

[71] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.

[72] C. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[73] J. E. Pilliod and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *J. Comput. Phys.*, 199:465–502, 2004.

[74] S. Popinet. Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[75] P. Raad, S. Chen, and D. Johnson. The introduction of micro cells to treat pressure in free surface fluid flow problems. *J. Fluids Eng.*, 117:683–690, 1995.

[76] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directible photorealistic liquids. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 193–202, 2004.

[77] Y. Saad. *Iterative methods for sparse linear systems.* PWS Publishing, 1996. New York, NY.

[78] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24, 2005.

[79] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*, 93:1591–1595, 1996.

[80] J. Sethian. Fast marching methods. *SIAM Review*, 41:199–235, 1999.

[81] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[82] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes II (two). *J. Comput. Phys.*, 83:32–78, 1989.

[83] S. Song and T. Chan. A fast algorithm for level set based optimization. In *CAM report (02-68)*, 2002.

[84] A. Staniforth and J. Cote. Semi-Lagrangian Integration Schemes for Atmospheric Models: A Review. *Monthly Weather Review*, 119:2206–2223, 1991.

[85] J. Strain. Fast tree-based redistancing for level set computations. *J. Comput. Phys.*, 152:664–686, 1999.

[86] J. Strain. Tree methods for moving interfaces. *J. Comput. Phys.*, 151:616–648, 1999.

[87] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.*, 187:110–136, 2003.

[88] M. Sussman, A. Almgren, J Bell, P. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comput. Phys.*, 148:81–124, 1999.

[89] M. Sussman and E. Fatemi. An efficient interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM J. of Scientific Comput.*, 20:1165–1191, 1999.

[90] M. Sussman, E. Fatemi, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flows. *Computers and Fluids*, 27:663–680, 1998.

[91] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, 162:301–337, 2000.

[92] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[93] H.-Z. Tang, T. Tang, and P. Zhang. An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions. *J. Comput. Phys.*, 188:543–572, 2003.

[94] A.-K. Tornberg and B. Engquist. A finite element based level set method for multiphase flow applications. *Comput. and Vis. in Science*, 3:93–101, 2000.

[95] A.-K. Tornberg and B. Engquist. Numerical approximations of singular source terms in differential equations. *J. Comput. Phys.*, 200:462–488, 2004.

[96] Y-H. Tsai and S. Osher. Level set methods and their applications in image science. *Comm. Math Sci.*, 1(4):623–656, 2003.

[97] J. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. In *Proc. 33rd Conf. on Decision and Control*, pages 1368–1373, 1994.

[98] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40:1528–1538, 1995.

[99] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[100] G. Ventura, E. Budyn, and T. Belytschko. Vector level sets for desription of propagating cracks in finite elements. *Int. J. Num. Meth. Eng.*, 58:1571–1592, 2003.

[101] M. Wiebe and B. Houston. The tar monster: Creating a character with fluid simulation. In *SIGGRAPH 2004 Sketches & Applications*. ACM Press, 2004.

[102] F.A. Williams. The mathematics of combustion. *SIAM*, pages 97–131, 1985.

[103] A. Yezzi, A. Tsai, and A. Willsky. A fully global approach to image segmentation via coupled curve evolution equations. *J. of Vis. Comm. and Image Representation*, 13:195–216, 2002.

[104] Y. L. Zhang, K. S. Yeo, B. C. Khoo, and C. Wang. 3d jet impact of toroidal bubbles. *J. Comput. Phys.*, 166:336–360, 2001.

[105] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127:179–195, 1996.