THE MARKER LEVEL SET METHOD: APPLICATIONS TO SIMULATION OF LIQUIDS

BY VIOREL MIHALEF

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer and Information Science

Written under the direction of

Professor Dimitris Metaxas

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2007

ABSTRACT OF THE DISSERTATION

The Marker Level Set method: applications to simulation of liquids

by Viorel Mihalef

Dissertation Director: Professor Dimitris Metaxas

Interface advection methods are important tools with applications in computer graphics and computer vision, as well as in computational fluid dynamics and other engineering domains. The classic level set method in particular is one of the most widely used methods for interfacial advection; however, this method is less successful in tracking high-curvature regions and thin sheets, and it completely discards information tangential to the interface.

In this thesis we introduce a new method for advection of interfaces by an external velocity field with improved performance in the problematic areas of the classic level set method mentioned above, and present several applications to simulation of liquids. We show that our method, which we term the Marker Level Set (MLS), provides an accurate, simple, and efficient alternative to the present technology for interfacial advection. Moreover, MLS features several capabilities that are quite important for computer graphics, such as automatic surface texture transport and an easy way of generating spray and small bubbles during simulation of liquids. We introduce as well a new MLS-based level set reinitialization procedure which gives improved performance over classical reinitialization procedures used in the context of the level set method alone.

Acknowledgements

First of all, my thanks go to Dimitri Metaxas, who knowledgeably and kindly guided my steps through the PhD program, offering all the support that a student could ever ask for. His energy and openness to numerous research directions are a constant source of inspiration.

My deepest thanks go also to Mark Sussman, who generously shared his level set and CFD knowledge as well as his magic Navier-Stokes solver, enabling the Marker Level Set to "come to life" through three dimensional simulations.

I also want to thank my two other committee members, Gerard Richter and Dinesh Pai, who helped me achieve a better understanding of many research details, and offered precious advice.

My colleagues in the lab helped me so many times from the beginning to the end of the program. In no particular order other than nationality, I would like to thank Chan Su Lee, Suejung Huh, Kyung-Ha Min, Kooksang Moon, Zhen Qian, Rong Zhang, Xiaolei Huang, Zhiguo Li, Xiaoxu Wang, Atul Kanaujia, Sundara Venkataraman. Special thanks go to Sukmoon Chang, who shared so much of his knowledge, code, research images, and, most importantly, humor, kindness and intellectual openness.

I am very grateful to our departmental staff, who made life easy for me on many occasions: Carol, Maryann, Aneta, Naomi, many thanks!

My thoughts and thanks also go back to my UPenn years, where I had the luck and pleasure of being advised by a great person, a great teacher, and a great researcher: Hermann Gluck.

Finally, and in many ways most importantly, my thanks and love go to Iuliana, whose constant support, wisdom and sense of humor are invaluable to me. Chow!

Dedication

To my parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The special effects industry has always required good methods for simulating free surface flows. Such methods need to be accurate, simple to implement, efficient, versatile and to offer a good degree of control on the simulation. This thesis presents a method for interface advection that has many such desired features. We will show that this method, which we term the *Marker Level Set* (MLS), provides an accurate, simple and efficient alternative to the present technology for interfacial advection. One of the most widely used interface tracking methods by the fluid simulation effects developers is the Particle Level Set method (PLS), and we will try to compare our method with that one whenever possible. As an overview, on standard two dimensional tests our method outperforms the PLS while being theoretically more efficient and easier to implement.

At the Siggraph 2004 conference the developers that worked on the movie Day After Tomorrow (2003) explained that, for simulating the scene of the flooding of the streets of New York City, they had to constantly reinitialize the position of the foam particles traveling with the giant wave, so that they align with the wave surface. Similarly, the developers who worked on the special effects involving the Sandman in Spiderman 3 (2007) reported at the Siggraph 2007 conference that they had as well to realign the sand or other particles supposed to travel on the surface of the Sandman's deforming body. For such cases the Marker Level Set method may be a blessing, given that it features markers on the surface at any time. The markers actually *define* the surface in the MLS method, and they are coupled with a level set which helps recover a smooth surface and also helps detect topological changes.

The surface markers characteristic to the MLS method offer several important capabilities to the method, namely automatic surface texture transport and an easy way of

generating spray and bubbles during simulation of liquids by using the deleted markers. The PLS is the only other interface advection method that offers similar capabilities, however MLS can handle surface texture dynamics with less diffusion induced by the interface subflow as we will argue later on.

Although we focus in this thesis on its applications to simulation and animation of liquids (that is, computer graphics) the MLS method may prove useful in computer vision as well. For example, as a more-accurate-than-the-level-set method it could be used to improve the results obtained with the standard level set method, for problems in which corner resolution or thin strands are important to detect or to track. It can also be used to find skeletons of two or three dimensional closed regions - and we present several encouraging preliminary results in the last chapter. We will also outline directions of possible research in computational fluid dynamics that may benefit from the features of the Marker Level Set. To name only two, surfactant controlled dynamics and atomization are among the most promising.

In the following chapters we will look first at alternative methods for surface evolution, including texturing issues, then we will present the general idea and its specific implementation, together with relevant two and three dimensional validating numerical tests. The issue of redistancing and its new incarnation in the MLS framework will be discussed in chapter 4, while the following chapter will be dedicated to showcasing the strengths of MLS when coupled to a Navier-Stokes solver, stressing on its texture dynamics capabilities and ease of generation small scale structures like droplets of bubbles. Conclusions and directions for future work will round up the thesis in the final chapter.

Chapter 2

MLS in the context of alternative methods for interface advection

In this chapter we will give an overview of several representative *interface advection methods* used in computational fluid dynamics and computer graphics, and make pertinent comparisons with the Marker Level Set Method where appropriate. The setup we are considering is the following: given an initial (closed) surface (or a closed curve in two dimensions) and a smooth velocity field $\mathbf{u}$ varying in time, one would like to find the position of the surface at any instant in time, given that each point of the surface moves with a velocity given by the velocity field at that point. The surface is allowed to change topology upon self-intersection or break-up. The motion of the surface points is thus governed by the equation $d\mathbf{x}/dt = \mathbf{u}(\mathbf{x}, t)$. Dual to this explicit formulation are the formulations that describe the surface and its dynamics implicitly. The level set method is an example of such an implicit method, describing the surface as the zero level of a three-dimensional scalar function, usually denoted with ϕ. The dynamics of the surface are embedded in the dynamics of the whole level set with the *level set equation* $\phi_t + \mathbf{u} \cdot \nabla\phi = 0$. Implicit methods like the level set method are usually *Eulerian*, meaning they are numerically discretized on a fixed grid. Explicit methods on the other hand are usually *Lagrangian*, in that the numerical discretizations are done in a moving coordinate system (usually respecting the characteristics of motion - we will come back to this later in the text). Exceptions exists, like the Lagrangian Level Set method proposed in Hieber and Koumoutsakos [1], but usually one uses the duality Eulerian/Lagrangian to situate a specific method, and we will shape our next discussion in such a manner. Anticipating, our Marker Level Set method is an Eulerian-Lagrangian hybrid, using "the best of both worlds" to its advantage.

2.1 Level Set based methods.

The Eulerian part of our method uses a level set method to advect the interface, and we begin our discussion of Eulerian methods with it. Level Set Methods are widely used tools for advecting interfaces with applications in computational fluid dynamics, computer graphics, computer vision, and many other fields, as presented by books like [4–6]. The original level set approach to interface capturing was proposed by Osher and Sethian [7], and is a particular case for a general class of the so-called interface capturing methods, in which the interface is not explicitly tracked but rather reconstructed from a scalar field (the level set in our case; in general this scalar field may be thought as a "color function"). Implementation simplicity and automatic handling of topological changes of the interface are the main strengths of the original level set method. Among its weaknesses lie excessive numerical regularization, which results in rounding of the interface corners and a possible overall-large mass loss or gain. These issues are addressed in a number of papers. For example, Sussman and Puckett [8] and Sussman [9] combine the use of level sets and volume fractions for exact mass conservation; however, one drawback of this method is that it is slightly inaccurate with respect to corner advection because both level set and volume of fluid methods are Eulerian, thus displaying similar numerical difficulties to the original level set method. Enright et al. [2] propose hybridizing Lagrangian and Eulerian techniques and obtain a successful particle level set method (PLS), in which Lagrangian massless particles are placed inside and outside the interface, advected concurrently with it, and used to correct the zero level set location. The success of the method consists in its high accuracy in high-curvature regions and good mass-preservation properties in simple tests. Potentially even more successful is the update of the PLS by Losasso et al. [10, 11] by using adaptive (octree) techniques. Similarly with the PLS, our markers provide subpixel information that the Eulerian grid loses due to discretization errors. Such information is not directly available in methods like the one proposed by Bargteil et al. [12], where Lagrangian information is given by the surface triangles obtained by isocontouring the (Eulerian) level set. The main two differences between their and our framework are that, first, their (Lagrangian) triangle

vertices are reinitialized every time step, based on the Eulerian reconstruction of the zero level set, thus are prone to reinitialization errors, while we do not reinitialize the markers, and thus do not introduce grid-based errors. Second, the number of markers used by our method is user-defined, and can be as large as the computational resources allow it to be, whereas they are constrained to use a number of Lagrangian markers between 3 and 12 in three dimensions (the number of the zero level set crossing points in a cubic cell). [1] proposed recently a Lagrangian PLS (LPLS), in which they also use a level set and particles placed within a tube about the interface. Their particles do not carry along spheres that define the interface, but rather Gaussian maps that spread their influence locally. Their method is more efficient than the PLS, but their 3D implementation is less accurate. Our second Marker Level Set method resembles the LPLS in the use of the Gaussian kernels but we place our particles directly on the interface, unlike the PLS type of methods, resulting in more accurate interface tracking.

2.2 Tangential information. Texture advection.

Another undesirable feature of the level set method and its Eulerian updates, cured by our MLS method, is the loss of tangential information along the interface in the level set setting. This is due to the fact that the tangential component of the advection velocity in the level set equation (1) is not taken into account - the advection velocity $\mathbf{u}$ is projected onto the local gradient of the level set, which is normal to the interface. The only works we are aware of that address this problem are Pons et al. [13] and Xu and Zhao [14]. Their respective formulations are based on the use of two coupled level sets. Although they manage to show successful results for simple cases of tangential advection, their purely Eulerian method still displays the aforementioned problems that mar the original level set approach. In particular, their method cannot be used for accurate computational fluid dynamics numerical simulations (which is one of the goals of our method). Our method recovers the tangential information, lost by Eulerian techniques that use the level set or the volume of fluid method, through the use of an unstructured set of surface markers. As an important note, for both the PLS and the LPLS methods

one may imagine a conceptual attempt to transport interfacial tangential information by using the particles near the zero level set and interpolating/extrapolating techniques. However, for highly deforming velocity fields, the nonzero level sets may suffer high distortions. As a consequence, the particles would not anymore be a reliable source for transporting information tangential to the interface. This does not happen in our MLS framework, because the markers are always situated *on* the interface, they define the interface! Here is a good place to mention that the advection of tangential information is especially important in graphics, specifically in the form of texture advection on surfaces. There are several methods that concentrate on advecting the texture through the flow field, e.g. [15–17], that work well for some fluid simulations, but not for our specific case of liquid surface (rather than volume) texture advection. Stam [18] showed beautiful fluid texture advection on arbitrary surfaces of fixed shapes, in contrast MLS can handle textured dynamic liquid shapes. To address the particular context of liquids, Rassmusen et al. [19] proposed a method that advects texture particles, initialized near the interface, through the fluid flow field. Their method, while closest to MLS among all the current advection methods, differs from MLS in an essential way: the markers they use are "passive" color carriers, whereas our method recognizes their essential capacity of carrying both texture and motion characteristic information. We also provide a full mechanism for deletion and addition of markers, including local color inheritance. Wiebe and Houston [20] and Houston et al. [21] stored three-dimensional texture coordinates in a grid structure and advected them as scalar fields. To avoid artifacts resulting from volumetric advection, the authors used extrapolation techniques to force the gradient of the texture field to be perpendicular to the interface normal. Kwatra et al. [22] and Bargteil et al. [23] present two similar approaches to generate liquid flows with user specified textures. In their work they use texture synthesis methods to create the illusion that the liquid has a volumetric patterned texture similar to the one initially visible on the surface. Our results can be viewed rather as simulating a liquid carrying a very shallow strip of surface paint, which is moved together with the surface and changes color due to mixing. Another difference is that in our approach the textures suffer less grid-based diffusion per advection step than in their work, due to the colors being

carried by the surface markers, in a Lagrangian rather than an Eulerian method. The method of Bargteil et al. [12] allows for advection of texture coordinates (or other surface properties) on the actual surface, even though no mechanism for texture generation in the case of topological break-ups was offered. We note that the MLS has a built-in marker addition routine for such cases, and along with it local interpolation of texture from neighboring markers.

2.3 Lagrangian methods. SPH based methods.

Another class of interface advection methods is that of the (Lagrangian) front tracking methods, which track the interface directly by placing particles/markers along the interface and advecting them directly, such as proposed by Unverdi amd Tryggvason [24] or Shin and Juric [25]. The usual problem with these types of methods is that three-dimensional formulations become complicated when dealing with topological changes and the subsequent manifold surgery. The method presented in [25] is a somewhat unique case in front tracking in that it manages to treat both two and three dimensional cases without using logical connectivity between elements. However, their method would have problems resolving flows featuring long and thin strands or many small droplets/bubbles because they use a global volume conservation technique. Their specific volume conservation method biases small curvature regions against large curvature regions in a nonphysical manner, and may lead to shrinking and ultimate disappearance of fine features like the ones mentioned above. For example, regions where the $\mathbf{u} \cdot \nabla\phi = 0$ (e.g. if the velocity equals zero) will be mistakenly adjusted. They may also lose mass as every once in a while they remove all markers and reconstruct them based on a possibly inaccurate indicator function. Our method does not perform such reconstruction which is ultimately destructive of the characteristics, but rather tries to carefully preserve them and, as a consequence, our volume is also very well preserved. The work of Raad and Bidoae [26] is similar to ours in the spirit of using surface markers to designate the position of the interface. However, their approach doesn't use a consistently closed surface representation of the interface (the zero level set component

of the MLS fulfills that function for us) and their results suffer as a consequence, displaying artifacts such as holes in the surface. For the same reason, their approach to increasing local surface marker density, based purely on interpolating marker positions, is likely to introduce extra artifacts in the surface location. Aullisa et al. [27] propose a useful method for accurately and efficiently tracking interfaces with the help of surface markers, showing better test results than PLS or any other approach based on a single scalar function, such as a volume fraction or a level set function. Their method is not formulated to deal with topological changes though, which is an important case that MLS does address successfully. Our method can be also compared to the various particle methods used in graphics, for example [28–30]. These methods are customarily variants of the smoothed particle hydrodynamics method of Monaghan [31], and as such they place particles throughout the body of the liquid; MLS is more efficient in this respect, placing particles only along the interface. Also, their surface reconstruction often uses an implicit method that is employed at every time step to recover a smooth surface from the particles. Because there is no dynamic link between the implicit surface and the particles (in MLS this link exists), these methods are prone to lacking temporal coherence. Adams et al. [32] is a nice recent attempt that improves a lot on the interface resolution by the SPH particles, proving that the SPH methods are reaching maturity and are a good candidate for attempting both CFD simulations and animations for computer graphics.

Chapter 3

The MLS method. General framework and implementation. Numerical tests.

In this chapter we will present in detail the theoretical idea of the MLS method, its implementation and various test results. In the next chapter we will also introduce a new method for solving the level set reinitialization equation, including a preliminary analysis of the cases in which the Russo and Smereka [3] reinitialization method may fail, but where our method does not. Before getting into the meat of the problem, we will make a short preamble concerning the advection equation in general and its treatment by the level set method.

3.1 Preamble: the level set method and the advection equation.

The standard level set setting is the following: in a domain $\Omega \subset R^n$ we are given a closed hypersurface Σ (which may be thought of as the interface between two different materials, e.g. water and air), which we represent as the zero level of a differentiable function ϕ (the level set), defined on the whole domain: $\Sigma = \{x \in \Omega : \phi(x) = 0\}$. Σ thus separates the positive region of the level set (water) from the negative one (air). The hypersurface moves inside Ω following a velocity vector field $\mathbf{u}$ defined on the whole domain (or extended to the whole domain from a velocity vector field defined along the interface). The equation of motion is the *transport* (or *advection*) equation

$$\frac{D\phi}{Dt} = 0 \tag{3.1.1}$$

written with the use of the material derivative $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$. The standard interpretation of this equation is that ϕ stays constant along the characteristics of motion. From a mathematical point of view, this is a first order (linear) PDE, and one of the preferred

methods for its resolution is the *method of characteristics*. Intuitively, a characteristic curve can be understood as the trajectory of a massless marker following the path of motion. A characteristic curve has the nice mathematical property that the original PDE to be solved reduces to an ODE along each characteristic curve. For example, if we work in one dimension and the velocity field is constant, the characteristics can be computed to be parallel straight lines. A similar situation occurs if one solves the inviscid Burgers'equation, although in this case the straight lines may intersect (see for example chapter 3 in Chorin and Marsden [33]). The characteristics in our case of a variable smooth external velocity field are non-intersecting curves. The method of characteristics has an immediate implication in choosing the correct spatial discretization method for 3.1.1, namely the so-called upwind methods. These methods bias the discretization stencil in order to conform to the local flow of information which follows the characteristics. For example, in one dimension, if the velocity is positive at a node, the values of ϕ are moving from left to right, and one looks to the left of the node to determine what value of ϕ will land on the node at the end of the time step. In practice, we discretize the equation 3.1.1 on an Eulerian grid and we solve it using first order upwind methods of advection for the level set (specifically, we use an implicit semi-Lagrangian method, as in Strain [34]). Although higher-order discretization methods like Hamilton-Jacobi ENO or WENO (see [4, 5, 7]) have been an important part of the level set methodology from its birth, they are less important for us (in a first stage at least), due to the subgrid accuracy induced by the markers, that alleviates the need of using such high-order methods. The interface can be reconstructed at any step by using a contouring routine to extract the zero level set, for example, marching squares/cubes in 2D/3D. More often than not, authors prefer to keep the level set as a signed-distance function with respect to the interface, in order to ensure more robust and accurate computation of gradients and other constant level sets. With this in mind, Chopp [35] introduced the idea of *reinitialization*, whose purpose is to maintain the signed-distance property. Sussman et al. [36] formalized it into solving the reinitialization equation

$$\frac{\partial \phi}{\partial t} = sgn(\phi^0)(1 - |\nabla \phi|) \tag{3.1.2}$$

and proposed upwind methods for solving it. Related important contributions are Peng et al. [37], who discuss improvements to the numerical solution of 3.1.1 and use them in the development of fast level set methods, and Adalsteinsson and Sethian [38], who obtain the signed distance function as a bi-product of the fast marching method. Russo and Smereka [3] discuss various artifacts associated with certain numerical solutions to 3.1.1 and offer solutions to them. In the next chapter we show how their work can be improved in the MLS framework.

Equations 3.1.1 and 3.1.2 need to be solved only in a neighborhood of the interface (see e.g. [37,38]), and we apply the same philosophy. One of the advantages of the level set method is that many important geometrical quantities can be readily calculated from the level set function, for example the unit normal $\mathbf{n} = \nabla\phi/|\nabla\phi|$ and the curvature $\kappa = \nabla \cdot (\nabla\phi/|\nabla\phi|)$.

3.2 General framework of the MLS.

Our method uses a set of markers placed along the zero level set. They are used for several purposes: 1) to track characteristic information, 2) to help reconstruct the interface in regions where the level set method has failed to accurately preserve mass, and 3) to provide tangential dynamics information that the level set method discards. Conceptually, our approach is more efficient than the PLS or other standard methods such as Harlow and Welch [39] or Rider and Kothe [40], which place particles either throughout the domain or in a narrow band about the zero level set. We first place marker particles in a regular fashion along a linear reconstruction of the zero level set, obtained with marching squares/cubes. After this **marker initialization process** we advect the particles using, at each time step, the evolution equation

$$\frac{d\mathbf{x}^p}{dt} = \mathbf{u}(\mathbf{x}^p) \tag{3.2.1}$$

where x^p is the position of the particle and $\mathbf{u}$ is its velocity. Particle velocities are interpolated from the velocities on the underlying grid using either trilinear or tricubic interpolation. Similarly to the proponents of the PLS, we found that we need the numerical solution of 3.2.1 to be at least second-order accurate for good results, and we

used the second-order Runge-Kutta (Heun's version) in order to accomplish this.

We limit the total number of markers that lie inside a cell, which the PLS does as well. In our experiments, we initialize up to 20 markers per interface cell and allow a maximum of 50 markers (this maximum is rarely reached or needed though; in fact, as we will show in the examples section, the average number of markers per cell was in the range of 6-9). If more markers enter a **full** cell we delete the new comers. When dealing with stretching/compressing of velocity fields, markers may clump together; limiting the maximum number of markers per interface cell is one of the methods that helps keep the markers relatively well distributed. Other methods consist in the addition and deletion of markers. We use two routines for these processes, with the following logic.

Add Markers If the number of markers within an interface cell drops below a certain threshold (1 in our case) we consider adding markers to the cell. If the length (area in 3D) of the interface is above a certain threshold τ (for example $\tau = \Delta x/2$), then we add markers along the linear reconstruction of the interface (in a number proportional with its length); this ensures that we don't add markers all the time the interface enters a new cell, and makes the method more robust. If the length (area in 3D) of the interface is below τ, we add only two markers (in 2D), at the intersection with the cell edges. For all the results presented in this thesis we do not apply any extra regularization techniques to ensure a uniform marker distribution. The addition/deletion mechanism already provides good results for practical purposes, although there is still room for further improvement (for more considerations on this please see the convergence section in this chapter).

Delete Markers Our approach to deleting surface markers is inspired from physical reality: we consider that interfacial markers carry properties of both phases. Let us illustrate this concept with two examples. If a thin sheet of water stretches more and more, it becomes a spray of fine liquid particles due to surface tension. In such a case the Lagrangian surface markers can be used to capture the spray that would be missed by a coarse Eulerian grid, hence they should not be deleted. Similarly, if an air bubble "dissolves" inside water, leading to formation of an underwater cloud of micro bubbles, the surface markers could be used to capture this phenomenon. Again, this

suggests that they should not be deleted. In both these examples markers represent *subcell information lost by the Eulerian grid.* While this seems to indicate that the most realistic philosophy would be not to delete surface markers, our experiments show that when implementing MLS one has to take into consideration the specific correction methods for the level set values based on the marker placement. In particular, marker deletion contributes to a more robust overall MLS method. Thus, for the examples presented in this work we do delete markers, but only if they lie in "positive" cells (with all four corners positive). Such markers would customarily appear when reconnection takes place, and are deleted to mark the topological reconnection of two bodies of liquid. In contrast, markers that lie in "negative" cells are not deleted, and this allows the MLS method to capture thin sheets of liquid, similarly with the PLS method.

The MLS method starts with the initialization procedure already described, and then repeats the following procedures every time step:

1. Advect the level set
2. Advect the markers
3. Correct the level set values based on marker positions
4. Add and then delete markers

3.3 Implementation.

We covered all the steps with the exception of the fourth one in the previous two sections. Next, we present two versions of the MLS that differ from one another in the method used to implement step 3. To get an idea in advance of our strategy, MLS-1 uses markers in two fashions to help with the update of the signed-distance. In a first step, it updates the local distance based on marker position, but preserves the sign. In a second step, it updates the sign using a linear reconstruction/approximation of the $\pm\Delta x$ level sets. MLS-2 uses markers in some neighborhood of each grid point to apply a correction to the level set value. The correction is performed so that it forces the level set to take (approximately) zero values at the marker positions. The two versions have

similar accuracy. The first version seems to be slightly more robust while the second one is more efficient and easier to implement in three dimensions.

3.3.1 MLS-1, a first version of the MLS

The first thing that one naturally tries to do in order to carry information from surface markers to the level set is to update the level set value by keeping its sign but modifying its value, based on the exact distance to local markers. We accomplish that with the following routine:

Update Distance

for each marker mk

for any grid point (i,j) in a small neighborhood of mk

$$|\phi_{i,j}| = \min(|\phi_{i,j}|, dist(node_{i,j}, mk)$$

Here *dist* is the exact Euclidean distance. The whole update takes O(M*k) operations, where M is the total number of markers and k is the dimension of the zero level set (e.g. 2 in three dimension). It is important to note that, at the end of this procedure, all the values in a narrow band about the interface have correct absolute values. Furthermore, all the nodes farther than $\pm\Delta x$ from the interface have the correct sign as well, hence we can consider them fully updated.

However, updating the distance is not enough - a subtle local error mechanism requires more work to be done. This error can cause the level set to "hang" over grid nodes (see figure 3.1 for a 2D illustration of the effect).

The "hanging effect" is present only in more than one dimension. It simply consists of a node receiving wrong sign information from the level set computation at some instant in time, and being unable to recover its correct sign at subsequent time steps even after the distance is updated from local markers. In order to eliminate this effect we need to somehow send the correct sign information to the level set. Figure 3.1 indicates that the $\pm\Delta x$ level sets are correctly constructed (these are obtained using the same marching squares algorithm as for determining the zero level) and do not suffer from

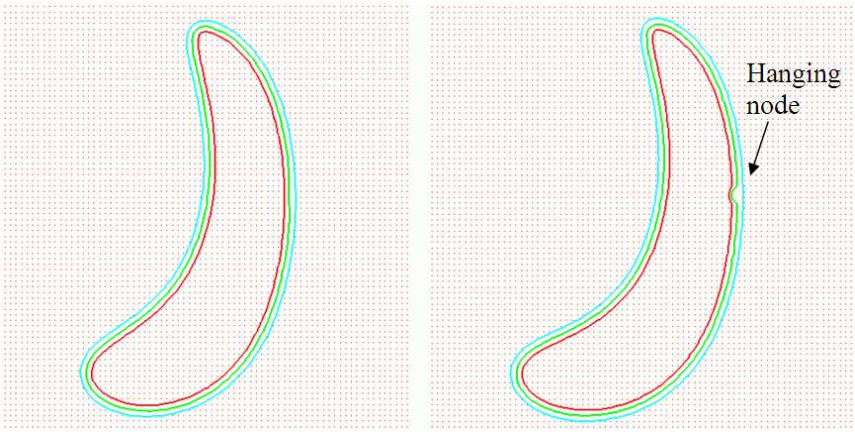

Figure 3.1: **The "hanging" effect in 2D: before and after hanging occurs. The grid point that "hangs" has the wrong sign information, even though updated with the correct distance information. The interface is figured in green, the $\pm\Delta x$ level sets in light blue and red.**

any artifacts, unlike the zero level set. This is the core idea for MLS-1: to use the $\pm\Delta x$ level information in order to reconstruct the correct sign within the narrow band. This is the second (and last) step in which we use (indirectly) the marker information to reconstruct the level set about the zero interface:

Update Sign

for each "interface node" (i,j)

$d1 = dist((i,j), -\Delta x$ level set)

$d2 = dist((i,j), +\Delta x$ level set)

if $d1 < d2$

$$sgn(\phi_{i,j}) = sgn(d1 - \Delta x)$$

else

$$sgn(\phi_{i,j}) = sgn(\Delta x - d2)$$

This second procedure updates the sign of the level set in the immediate vicinity of

the interface, essentially selecting the closest characteristic information of the reinitialization equation 3.1.2 in making its decision on which sign to choose. For computing the distances $d1$ and $d2$ we can either use level set values, or compute exact distances to the reconstructed linear approximations to the $\pm\Delta x$ level sets.

Our implemented algorithm seems to indicate that more accurate results are obtained when an extra Update Distance procedure is run after all was said and done, i.e. after step 5 (as a note, the original PLS works the same way). This way the signed distances are updated using the correct sign, thus the hanging effect does not occur anymore. Note that, because we compute in this method exact distances to the interface in a tubular neighborhood, there is no need to solve the reinitialization equation 3.1.2.

3.3.2 MLS-2, a second version of the MLS

Our second version of the MLS computes correction terms for the level set values at each of the nodes located in a tubular neighborhood $\Sigma_{\Delta x}$ of the interface. Namely, for each node $(i,j) \in \Sigma_{\Delta x}$ we compute $\phi_{i,j}^{new} = \phi_{i,j} - \lambda_{i,j}$, with $\lambda_{i,j}$ the local corrections of the level set. These corrections are computed as an average of the current level set values at the closest local markers:

$$\lambda_{i,j} = \sum_k w_k \phi(x_k) / \sum_k w_k \tag{3.3.1}$$

where x_k are the positions of the markers from a small neighborhood of (i,j) and w_k are weights associated to these markers and the node (i,j). $\phi(x_k)$ is the interpolated value of the level set function at the marker location. By performing the local correction of the level set we effectively force its zero level to align with the markers. Thus, our procedure is simply the following:

Update LS Value

for each node (i,j) close enough to the set of markers

$\phi_{i,j}^{new} = \phi_{i,j} - \sum_k w_k \phi(x_k) / \sum_k w_k$

We used Gaussian weights of the form

$$w(q) = \begin{cases} \frac{e^{-(q/c)^2} - e^{-1/c^2}}{1 - e^{-1/c^2}}, & 0 \le q \le 1; \\ 0, & \text{otherwise.} \end{cases} \tag{3.3.2}$$

where $q = q_{i,j}(x) = dist(x, (i,j))/\rho$, $c \in [0,1]$ is a constant and ρ is the kernel radius (the distance beyond which the weight vanishes).

We also need to perform periodic reinitialization of the level set in this setup. To that end, we solve 3.1.2 with our update of the Russo-Smereka reinitialization procedure (see [3]) in the context of the MLS, developed in last section of the next chapter. The scheme is the following:

Reinitialize

for each node (i,j)

$$\phi_{i,j}^{n+1} = \begin{cases} d_{i,j}, & (i,j) \in \Sigma_{\Delta x} \\ \phi_{i,j}^{n} - \Delta t \cdot sgn(\phi_{i,j}^{0})(G(\phi)_{i,j} - 1), & \text{otherwise.} \end{cases}$$

where $d_{i,j}$ is the local distance as computed by the MLS2 update. We use an upwind scheme to compute $G(\phi)_{i,j}$, the approximation to the gradient of ϕ.

3.4 Numerical tests.

3.4.1 Rigid Body Rotation of Zalesak's Disk

We first test the rigid-body rotation of Zalesak's disk in a constant-vorticity velocity field, as in [41]. The initial data is a slotted circle centered at (50,75) with a radius of 15, a width of 5, and a slot length of 25. The velocity field is given by

$$\begin{aligned} u(x,y) &= (\pi/314)(50 - y) \\ v(x,y) &= (\pi/314)(x - 50) \end{aligned}$$

so that the disk completes one revolution every 628 time units. We use a 100×100 grid, and the time step equals the grid spacing.

Figures 3.2 and 3.3 illustrate the high quality solutions obtained with MLS-1 and MLS-2, after one or two rotations. The final solution is plotted overlapped with the

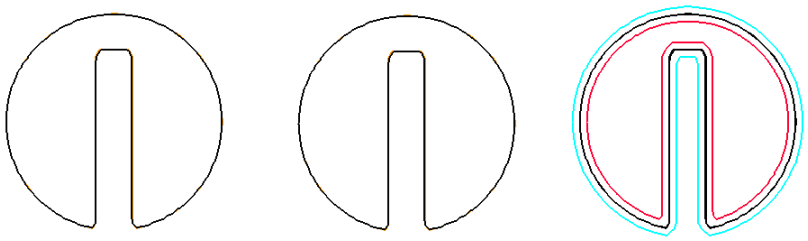

Figure 3.2: **Zalesak's problem using MLS-1. The MLS-1 solution is in black, theory in orange. From left to right: after one rotation, after two rotations, after one rotation showing also the $\pm\Delta x$ level sets.**

Table 3.1: **Comparison of initial particle counts for MLS, PLS and LPLS. The results for PLS and LPLS come from [1].**

Spacing	MLS-1	MLS-2	PLS	LPLS
1/50	770	413	3328	208
1/100	1552	840	12864	804

theoretical solution for the sake of comparison, and the position of the $\pm\Delta x$ level sets after one rotation is also shown. These figures indicate that both MLS versions do well in rigid motion velocity fields and are very good at preserving the corners of the original geometry. Level set reinitialization was used for MLS-2 only. For the sake of comparison we show also in figure 3.4 the result of PLS and simple level set method in the Zalesak test case. Our result is comparable to the PLS and superior to the level set solution.

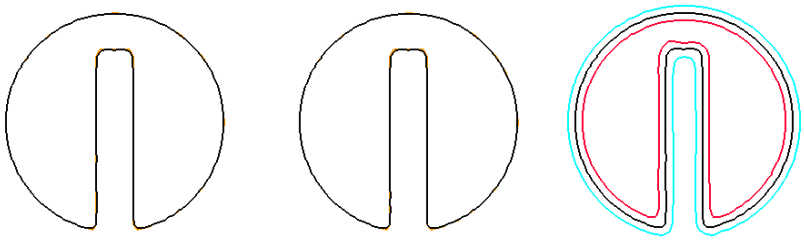

Figure 3.3: **Zalesak's problem using MLS-2. The MLS-2 solution is in black, theory in orange. From left to right: after one rotation, after two rotations, after one rotation showing also the $\pm\Delta x$ level sets.**

Table 3.2: **Area preservation properties for Zalesak's problem**

		MLS-1		MLS-2		PLS (from [2])	
	Grid Cells	Area	Area Loss	Area	Area Loss	Area	Area Loss
One rev.	exact	582.2	0%	582.2	0%	582.2	0%
	50	576.14	1.04%	577.38	0.82%	495.7	14.9%
	100	581.04	0.20%	580.96	0.21%	580.4	0.31%
	200	581.73	0.08%	581.78	0.07%	581.0	0.20%
Two rev.	exact	582.2	0%	582.2	0%	582.2	0%
	50	576.12	1.04%	576.54	0.97%	487.6	16.2%
	100	581.05	0.20%	581.04	0.20%	578.0	0.72%
	200	581.81	0.07%	581.80	0.07%	580.0	0.38%

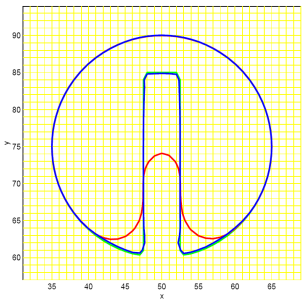

Figure 3.4: **Comparison of the level set solution (red), particle level set solution (blue) and theory (green) after one revolution. See also [2]. Our result is slightly better than the PLS one. This is visible in our solution having more overlapping between the initial and final curves compared to PLS, and also in the better area conservation properties reported in table 3.2.**

For each of the tests we ran, we performed a comparison between the area loss that each of the MLS methods incurs, and the results for PLS and LS from [2]. Overall, we obtained better area preservation than PLS and much better than the LS. Here are the results for the Zalesak problem for the two MLS methods and the PLS (from Enright et al. [2]).

The total number of markers in the Zalesak problem stays fairly constant (in the range of the initial number of markers 1544, which resulted over the whole period in an 8.63 average/cell for MLS-1 and an 8.6 average/cell for MLS-2). We present in Figure 3.5 the variation in time of the average marker-per-cell count. In Table 3.1 we show the initial particle counts for MLS-1 and MLS-2, and how they compare to PLS and LPLS.

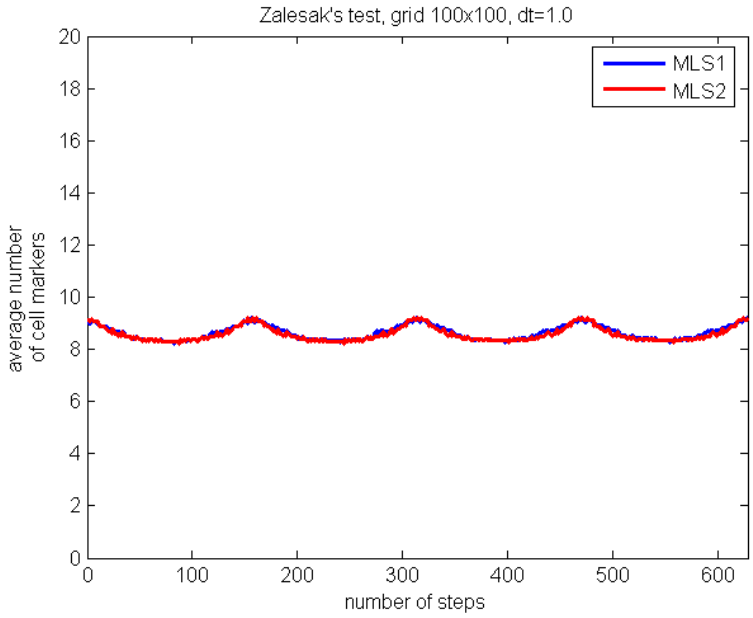

Figure 3.5: **Variation in time of the average marker-per-cell count in interfacial cells for the Zalesak test.**

In fact, MLS-2 was able to sustain good runs with fewer particles than MLS-1, and both were using much fewer particles than PLS and slightly more than LPLS. The data for PLS and LPLS come from [1]. In section 3.4.4 we will show and discuss numerical convergence results for Zalesak and the other 2D tests we ran.

3.4.2 Single Vortex

In order to check how the MLS does when dealing with stretching and tearing flows we use the "vortex-in-a-box" problem proposed by Bell et al. [42]. The velocity field is defined by the stream function

$$\Psi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y) \tag{3.4.1}$$

The computational domain is the unit square with a disk of radius 0.15 placed at $(0.5, 0.75)$, and we use a 128×128 grid. The time step used for this grid was 0.01.

The velocity field stretches the disk into a long thin spiral, testing the capacity of the interface advection method to preserve long thin strands. PLS is known to be one of the few methods that can pass this test quite successfully. We can see that the MLS methods also do well. Figure 3.6 shows the solution obtained with MLS-1 at time t=5. In green we show the initial green markers (2334 altogether at the beginning, or 20 per cell) and in brown the markers during the simulation. The total number of markers at the end of the run is 9417.

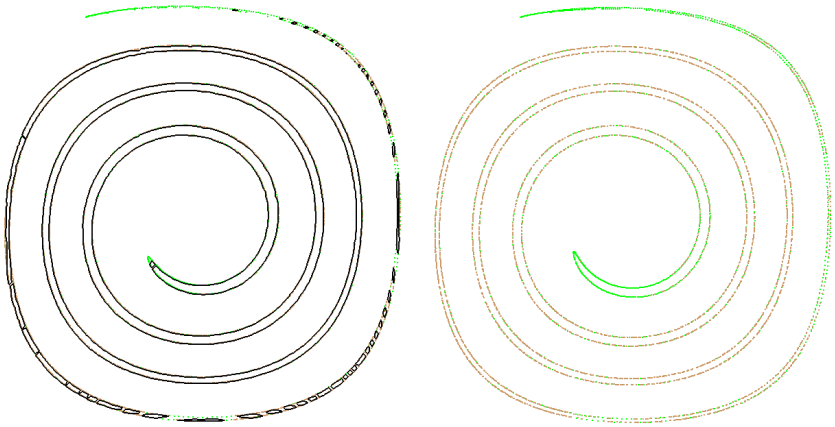

Figure 3.6: **The MLS-1 solution to the vortex flow (black) show here together with initial particles (green) and added particles (brown) at time $t = 5$. On the right we show only the particles.**

Figure 3.7 illustrates how MLS-1 performs in the same experiment if we modulate the velocity field so that it is time reversed until the geometry returns to the initial configuration. This is achieved by multiplying the velocity field by , where T is the time at which the flow returns to its initial state (see [43]). The reversal period used in the error analysis of the vortex problem is T = 8, producing a maximal stretching at T = 4. Figure 3.7 shows the configuration at time 4 on the left, and the usual initial (green) and added (brown) particles in the middle, while on the right we show superimposed

the initial configuration (in orange) and the final solution after one period (in black). The overlapping is almost perfect. Our solution is slightly more resolved that the PLS one (see for example [44], fig.4). The total particle count is 4880 at T = 4 and 3480 at T = 8.

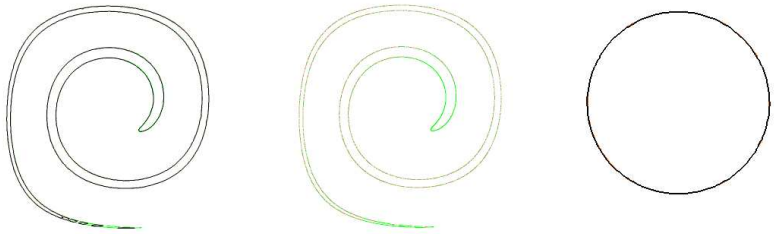

Figure 3.7: **MLS-1 results for the periodic vortex flow. Left and middle show the interface (black) and the particles (initial = green, added = brown) at maximum stretching time T=4. On the right we see overlapped the original (orange) and the final (black) configuration of the interface.**

In figure 3.8 we illustrate the MLS-2 simulation results. The total marker count at T = 5 is 5076. We used trilinear interpolation of the level set function at the markers positions in the Update LS Value procedure (section 3.3.2) - this ensured the presence of the interface almost everywhere a marker was present. For the periodic vertex flow problem (figure 3.9), the marker count was 3045 at T = 4 and 2827 at T = 8. We performed reinitialization every tenth step. Again, the solution returns to an almost perfect circle and, at T = 4, the interface was even better resolved than for MLS-1 (again, an effect of the trilinear interpolation). In Table 3.3 we display the percentage of lost area and we notice improvement even over the PLS for both MLS implementations with full particle addition/deletion.

For the periodic single vortex problem we measured over the whole period a 7.73 marker average/cell for MLS-1 and a 6.74 marker average/cell for MLS-2. The variation in time of the average marker-per-cell count is presented below. Further down, in Table 3.4, we compare again the initial particle count for successful runs using MLS, PLS and LPLS. The MLS methods use 3-6 times fewer particles than PLS and 2-5 times

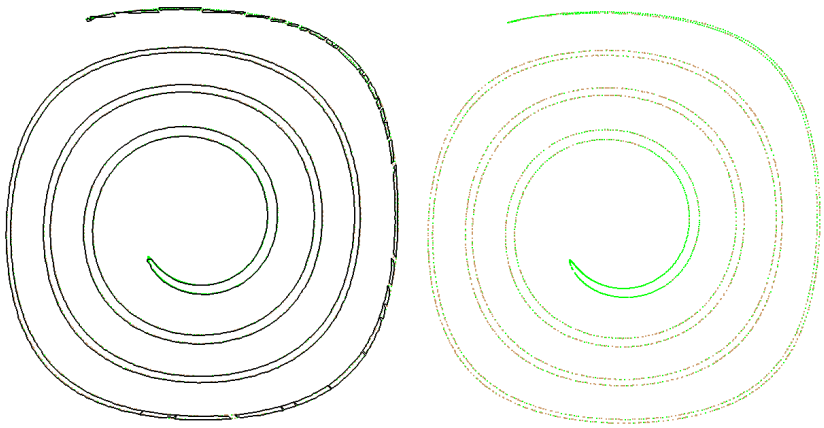

Figure 3.8: **The MLS-2 solution to the vortex flow (black) show here together with initial particles (green) and added particles (brown) at time t=5. On the right we show only the particles.**

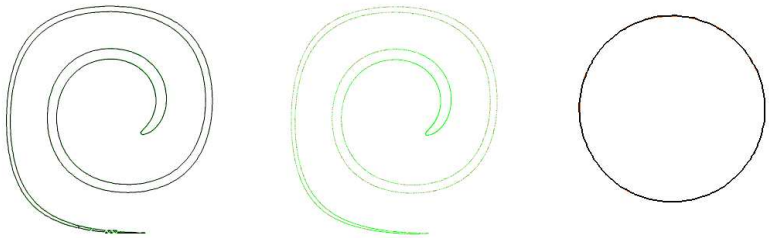

Figure 3.9: **MLS-2 results for the periodic vortex flow. Left and middle show the interface (black) and the particles (initial = green, added = brown) at maximum stretching time T=4. On the right we see overlapped the original (orange) and the final (black) configuration of the interface.**

more than LPLS. Compared to a second order implementation of the standard level set method, MLS2 was twice as slow, for and initial marker density of 6 markers/interfacial cell. For numerical convergence results please see section 3.4.4.

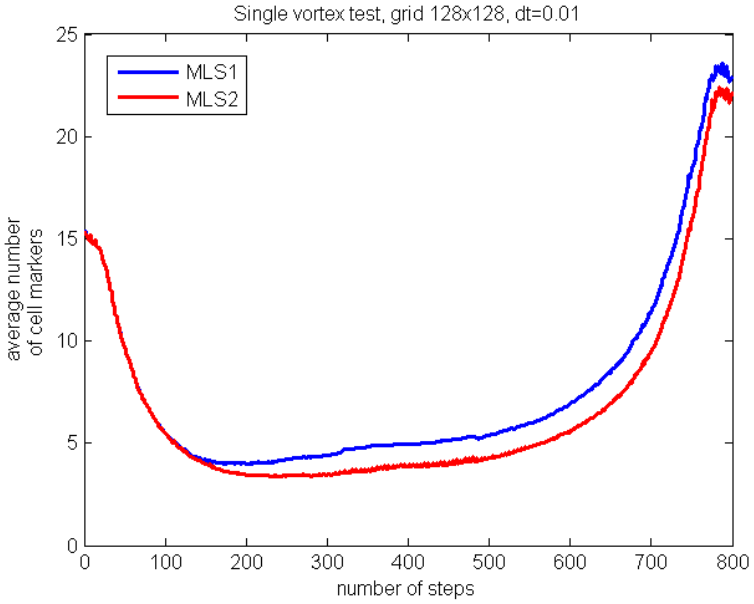

Figure 3.10: **Variation in time of the average marker-per-cell count in interfacial cells for the 2D single-vortex problem.**

Table 3.3: **Area preservation properties for the single vortex problem after one period**

	MLS-1		MLS-2		PLS	
Grid Cells	Area	Area Loss	Area	Area Loss	Area	Area Loss
exact	0.07069	0%	0.07069	0%	0.07069	0%
64	0.0706	0.12%	0.07036	0.46%	0.0694	1.81%
128	0.07061	0.11%	0.07062	0.09%	0.0702	0.71%
256	0.07066	0.04%	0.07063	0.08%	0.0704	0.35%

Table 3.4: **Comparison of initial particle counts for MLS, PLS and LPLS in the single-vortex problem. The results for PLS and LPLS come from [1].**

Spacing	MLS	PLS	LPLS
1/64	1164	3776	236
1/128	2334	15040	940

3.4.3 2D Deformation Field

The final test is the most difficult so far, with more dramatic stretching and tearing involved inside a unit box. It involves the entrainment of a circle of radius 0.15 in a deformation field defined by 16 vortices (as introduced by Smolarkiewicz [45]). The velocity field is periodic in space and given by the stream function

$$\Psi = \frac{1}{4\pi} \sin(4\pi(x + 0.5)) \cos(4\pi(y + 0.5)) \tag{3.4.2}$$

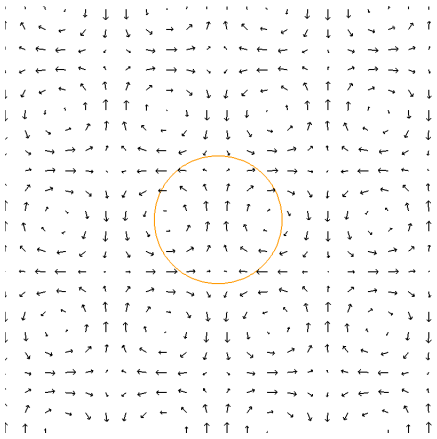

Figure 3.11: **The deformation field setup: 16 vortices treading on a central circle.**

Table 3.5: **Area preservation properties for the deformation problem after one period**

	MLS-1		MLS-2		PLS	
Grid Cells	Area	Area Loss	Area	Area Loss	Area	Area Loss
exact	0.07069	0%	0.07069	0%	0.07069	0%
64	0.0756	-6.95%	0.0703	0.54%	0.0696	1.59%
128	0.0706	0.12%	0.07064	0.06%	0.0705	0.26%
256	0.0706	0.05%	0.07067	0.02%	0.0705	0.26%

The field is made also periodic in time (of period 2) using the same procedure outlined in the single vortex section. We prefer to use (as in Aulisa et al. [46]) a slightly modified variant of the original test (figure 3.11), which places the initial circle at $(0.5, 0.5)$, rather than at $(0.5, 0.75)$, as was originally proposed. This is just for convenience and the final

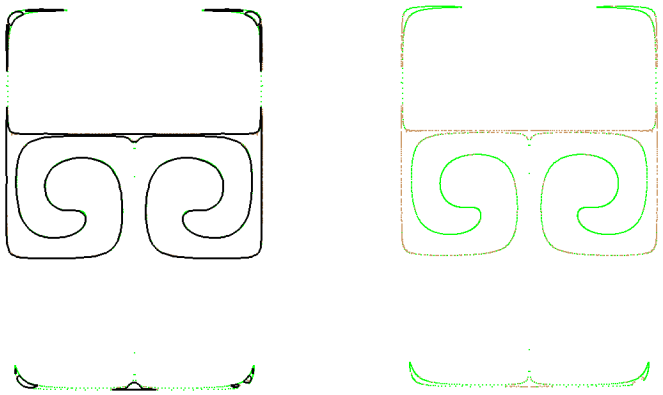

Figure 3.12: **Deformation field simulation using MLS-1 at $t = 1$. Interface in black, particles in green (initial) and brown (added).**

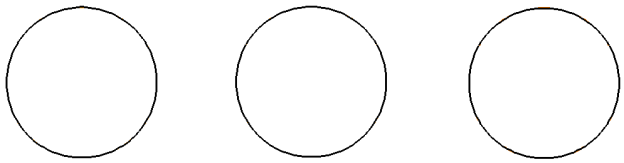

Figure 3.13: **Deformation field simulations at time $t = 2$. Final solution (black) superimposed onto theoretical solution (orange). From left to right: MLS-1, MLS-2 without marker addition/deletion, MLS-2 with marker addition/deletion.**

results are symmetric to the original procedure (compare [2] and [46] for example). The initial marker count for the 128×128 test was 2368. The results for MLS-1 at time $t = 1$, when the maximum stretching occurs, are presented in figure 3.12. The marker count at that time is 4089, while at time $t = 2$ (when the periodic flow reverses the circle to the initial position), the marker count is 2569. At time $t = 2$ the flow returns to a perfect circle (figure 3.13). MLS-2 performed similarly with MLS-1 (fig. 3.13 and 3.14) and we tried both a version that does not perform marker addition/deletion and one that does. Both versions performed similarly at the moment of maximum stretching ($t = 1$), but we noticed that the version that uses marker addition and deletion may introduce extra

markers that would be considered flotsam and jetsam. This happens if one performs too frequent addition of markers (we found that adding markers every tenth step is good enough, and the results obtained prove this). We show again excellent area preservation properties for both MLS (superior to PLS in almost all cases) in Table 3.5.

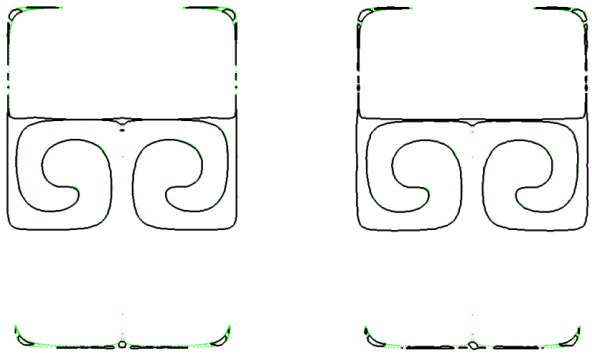

Figure 3.14: **Deformation field simulations using MLS-2 at $t = 1$. No particle addition/deletion on the left, with marker addition and deletion on the right.**

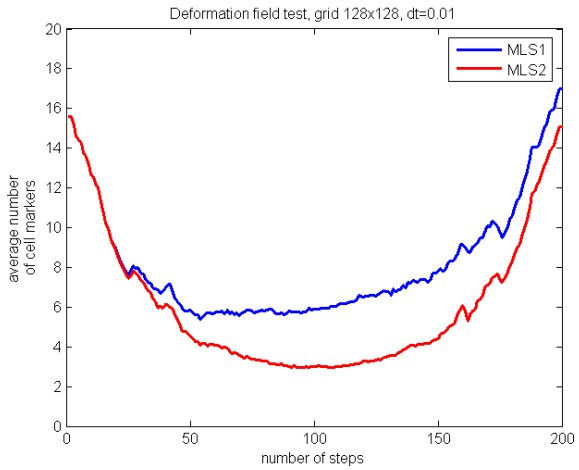

Figure 3.15: **Variation in time of the average marker-per-cell count in interfacial cells for the 2D deformation problem.**

Finally, for the periodic deformation field problem, we measured over the whole period an 8.27 marker average/cell for MLS-1 and a 6.18 marker average/cell for MLS-2. Figure 3.15 displays the variation in time of the average marker-per-cell count.

3.4.4 Convergence issues

A proof of convergence for either MLS1 or MLS2 would be a rather involved matter, in which one would have to account for the global effect that local operations like the level set error correction, deletion and addition of markers would have on the numerical solution. This would need to also take into account variables like the gaussian width and slope of the weight functions and the order of the local interpolation. A simpler case one can look at is the one in which there is no marker addition, and one either enables marker deletion or not. In such a case let us first note that the MLS algorithm consists of modifying the level set solution (known to be convergent, given that one enforces stability) in a set of points in the neighborhood of the interface. The modification is based on marker positions that are easily seen to converge to the exact solution in the linear case (externally generated smooth velocity field) as they give (stable) approximations to the motion characteristics with order of accuracy equal to the temporal order of accuracy of the markers (equal to 2 in our tests). If the local error correction of the level set based on marker position is stable and does not increase the error order of accuracy then the MLS would converge. While we have no proof yet that this is the case, the numerical convergence tests performed for MLS2 without marker addition or deletion indicate that our error correction is indeed stable and the convergence reaches second order of accuracy. We should however underline the fact that the method has an order of convergence that depends on several factors: the order of convergence of the markers, the order of convergence of the level set, the frequency of deletion and the frequency of addition of markers.

The numerical tests refer to the three problems presented before for a fixed resolution, the Zalesak, vortex in a box and deformation problems. We present the associated error and order of accuracy tables in 3.6, 3.7 and 3.8. The tests were performed without enabling the addition of markers, and this lead to obtaining convergence rates approaching

two - the marker advection order of accuracy. We expect that *the order of accuracy of marker advection will be the one dictating the MLS order of accuracy*, at least in linear cases like the ones tested. The error was computed similarly to [8] for example, by measuring how much the computed interface differs from the expected interface:

$$\int_{\Omega} \frac{1}{L} |H(\phi_{expected}) - H(\phi_{computed})| dxdy \tag{3.4.3}$$

where L is the perimeter size of the expected interface. The error is computed by

1. partitioning the domain into many tiny pieces (e.g., 1000×1000)
2. interpolating the values of $\phi_{expected}$ and $\phi_{computed}$ onto the newly created pieces
3. numerically integrating 3.4.3, where H is the already familiar Heaviside function.

We should mention that performing the MLS marker addition may decrease the accuracy of the overall method. This is at the moment due to our choice of a low accuracy marker addition algorithm, that uses the level set itself to introduce the new markers. The appeal of such an algorithm is that it is simple to implement, but one should be aware of its shortcomings - the main one being that it uses the possibly inaccurate level set information to generate the new markers. One can increase its accuracy in several ways: one can implement a method to generate the new markers at the roots of a cubic (rather than linear) reconstruction of the level set in the interfacial cells. One can use for example Newton's method to quickly generate such markers on the edges of interfacial cells. Another natural way to increase the accuracy is to increase the accuracy of the level set itself. Related to this, one may want to place the newly added markers in regions in which the curvature does not exceed a chosen threshold, as the level set solution's error scales proportionally with the curvature. Finally, a fourth method that may increase the accuracy of new marker placement is trying to make use of the local marker distribution, and we are currently investigating such an approach.

Table 3.6: **Absolute area interface error computations for Zalesak's problem using MLS2.**

Spacing	Error	Order
1/50	10.9	NA
1/100	1.1	3.3
1/200	0.33	1.7
1/400	0.1	1.7
1/800	0.028	1.8

Table 3.7: **Absolute area interface error computations for vortex problem using MLS2.**

Spacing	Error	Order
1/50	5.4e-4	NA
1/100	9.1e-5	2.5
1/200	2.8e-5	1.7
1/400	7.3e-6	1.9
1/800	1.9e-6	1.9

Table 3.8: **Absolute area interface error computations for deformation problem using MLS2.**

Spacing	Error	Order
1/50	19.5e-4	NA
1/100	3.4e-4	2.5
1/200	4.9e-5	2.7
1/400	7.5e-6	2.7
1/800	1.9e-6	2.0

3.4.5 Tangential dynamics

Our method features an extra quality not immediately shared by PLS or other interface capturing methods of comparable accuracy: it handles naturally movement tangential to the interface. The level set approach focuses only on motion normal to the interface and all tangential dynamics is lost. This effect is easy to illustrate if we consider a color field associated with the interface and monitor its dynamics in time. In figure 3.16 we show that nothing changes in time in the appearance of a circle placed into a rigid rotation field similar to the one from Zalesak's test (the grid size is 40×40), as there is no explicit connection between the color field and the zero level set. An attempt to associate color to vertices and advect the color as a vector field would show that diffusion problems appear very soon and the color field moves slower that the zero interface. The

PLS may also have some difficulty capturing accurately the tangential motion of the interface because its particles lie away from the zero interface. Consequently, under the influence of highly deforming velocity fields, the particles would lose accuracy in approximating tangential motion along the zero level set.

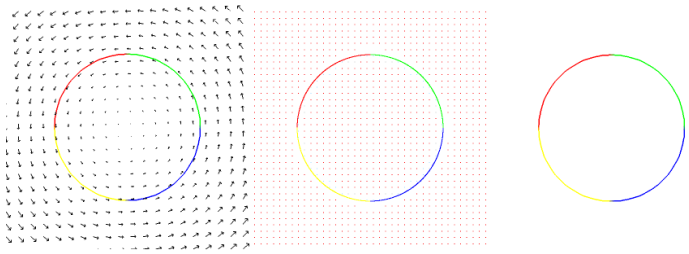

Figure 3.16: **Inside a rigid rotational field a circle represented as a level set doesn't transport its tangential information (interface color) in time. From left to right: (left) the velocity field and initial level set, (middle) the grid and initial particle placement and (right) the level set at times 0.66 and 1.32 seconds (and any subsequent times) when marker-level set coupling is not used.**

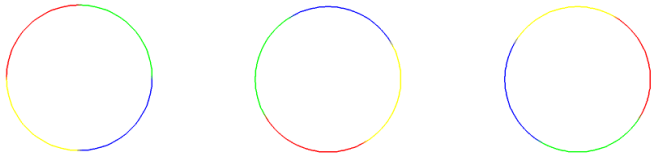

Figure 3.17: **The MLS method transports naturally the tangential information (interface color) in time. From left to right we show the level set at times 0, 0.66 and 1.32 seconds.**

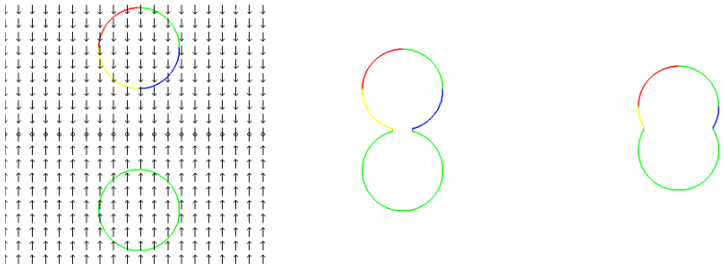

Figure 3.18: **A contact test: two circles with different colors connect. MLS takes care of the color change naturally.**

In contrast, fig. 3.17 illustrates the changes that occur as time passes, from the initial moment to times 0.66 and 1.32. All these are handled naturally by the MLS, by reading color information from the markers and using it to keep the node colors close to the zero interface always updated. As with every other detail of the MLS, this is a process that works painlessly in three dimensions as well.
As a final two dimensional test we present (in fig. 3.18) the contact of two circles of radius 0.15 placed on a unit grid inside a unit velocity field pointing up in the lower half and down in the upper half. The MLS handles both the contact and the color change of the interface.

3.4.6 3D examples: the Zalesak sphere test

Implementing a three dimensional version of the MLS-2 method requires minimal effort, as all the steps carry on easily from 2D. The only real novelty is that the marker initialization and addition is now being done along a surface rather than a curve. For this we reconstruct the surface at every time step using marching cubes and place markers regularly along each of the surface triangles.

The Zalesak sphere test consists (as in [2]) in a rigid rotation of a notched sphere of radius 15 placed at $(50, 75, 50)$ in a $[0, 100] \times [0, 100] \times [0, 100]$ domain. The slot is 5 grid cells wide and 12.5 grid cells deep on a $100 \times 100 \times 100$ grid cell domain. The constant vorticity velocity field is given by

$$\begin{aligned} u(x, y, z) &= (\pi/314)(50 - y) \\ v(x, y, z) &= (\pi/314)(x - 50) \\ w(x, y, z) &= 0 \end{aligned}$$

so that the sphere completes one revolution every 628 time units. Figure 3.19 shows the MLS solution at approximately equally spaced time intervals from $t = 0$ to $t = 628$. One may appreciate the non-diffusive quality of our MLS advection method, with the notch features being nicely preserved, unlike the diffusive performance of the level set method (see for example [2], fig.27 and 28).

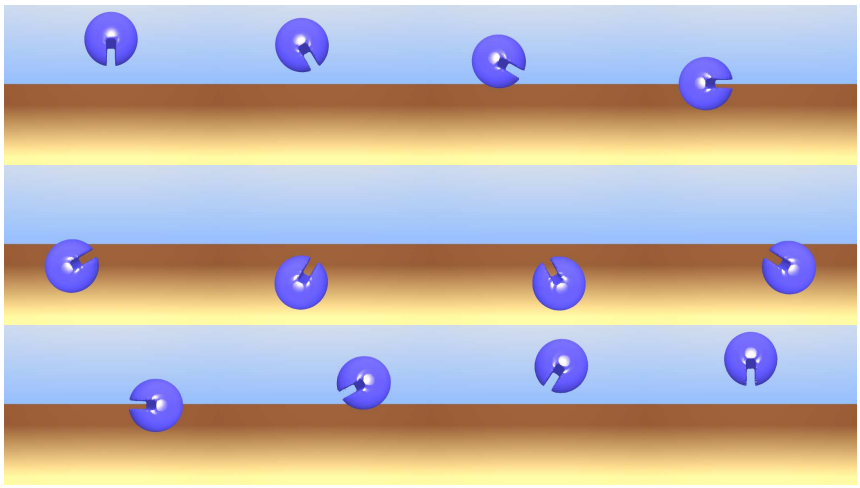

Figure 3.19: **3D Zalesak sphere test using the MLS method.**

3.4.7 3D examples: The Enright test with tangential dynamics

Our last test is the Enright test from [2], which places a sphere of radius 0.15 within a unit computational domain at $(0.35, 0.35, 0.35)$. The sphere is deformed by the three dimensional incompressible flow field proposed by [43], which combines a deformation in the $x-y$ plane with one in the $x-z$ plane. The velocity field is

$$u(x,y,z) = 2\cos(\pi T/2)\sin^2(\pi x)\sin(\pi y)\sin(\pi z) \quad (3.4.4)$$

$$v(x,y,z) = -\cos(\pi T/2)\sin(\pi x)\sin^2(\pi y)\sin(\pi z) \quad (3.4.5)$$

$$w(x,y,z) = -\cos(\pi T/2)\sin(\pi x)\sin(\pi y)\sin^2(\pi z) \quad (3.4.6)$$

T is the period of the flow, after which the sphere returns to its initial configuration. We used a domain of $100\times100\times100$ grid cells. In figure 3.20 we present the final results for a period $T=3$ test, at times $t=0$, $10\Delta s$, $20\Delta s$, $30\Delta s$, $40\Delta s$, $50\Delta s$, $60\Delta s$, $70\Delta s$, $90\Delta s$, $110\Delta s$, $130\Delta s$ and $150\Delta s$. Our results are comparable to the ones obtained by the PLS and much superior to the simple level set method ([2], fig. 29 and 30). The extreme stretching at the time $t=75\Delta s$ thins the interface to less than a grid cell, and several surface break-ups appear as a consequence. This is similar to the performance of the PLS, and an octree-based extension of the MLS would certainly improve the results,

as proved by [11].

Finally, we show in figure 3.21 the results of an Enright test with period $T = 2.5$, for which we applied a surface texture to the initial sphere. Each of the surface markers receives the initial texture information as an (R, G, B) triplet, and carries it along, thus showing in a dramatic fashion the tangential dynamics of the interface. Even after severe stretching and compression, the texture returns to its initial configuration with no visible diffusion. We also implemented successful routines that handle texture extrapolation to newly added markers (reported elsewhere). This functionality that offers immediate visual information about the tangential motion of the interface goes beyond present methods reported in CFD literature, as far as these authors know. Note that the MLS method works in such a manner that it handles automatically tangential motion/texture advection even for topology changing cases. We are certain that this will prove very useful for many numerical experiments involving interfacial dynamics.

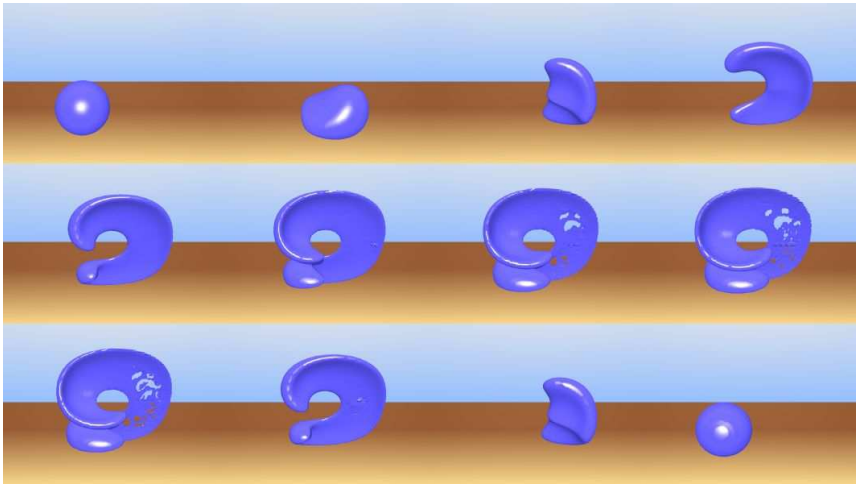

Figure 3.20: **The Enright test using the MLS method.**

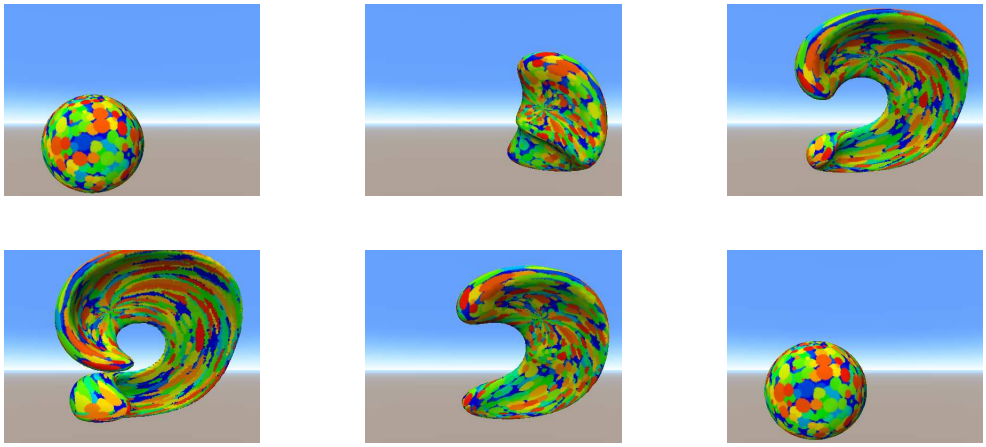

Figure 3.21: **Textured Enright test with period T = 2.5. First row times: 0, 0.16T, and 0.32T. Second row times: 0.5T, 0.75T, and T.**

Chapter 4

Analysis of the reinitialization equation. MLS reinitialization.

This chapter contains our new approach to reinitializing the level set based on the Marker Level Set approach. We will present several analytical and numerical insights that lead to the final solution. We start with a preamble concerning the reinitialization philosophy.

Many authors prefer to keep the level set as a signed-distance function in a tubular neighborhood of the interface, in order to avoid numerical errors in the gradient and the non-zero level sets due to the possible distortions that the level set may incur during its movement. Such distortions may be spatially unevenly distributed along the interface, and using the same stencil for the level set everywhere along the interface would not take into account this lack of uniformity. Maintaining the level set a signed-distance function amounts to keeping the length of the gradient equal to 1 throughout the domain. The process that performs this was called reinitialization by Chopp [35] where it was used in the context of minimal surfaces. Sussman et al. [36] and [47] used it in the context of free boundary problems in two-phase flow, Chen et al. [48] for crystal growth, Merriman et al. [49] for dynamics of multiple junctions. Other important contributions are Peng et al. [37], who used it in the development of fast level set methods, and Adalsteinsson and Sethian [38] who obtain the signed distance function as a bi-product of the fast marching method.

There are several methods of performing the reinitialization. The conceptually simplest one would be to measure the distance from each grid point to the interface itself and multiply it with the sign at the grid point. This is considered usually to be too expensive for practical purposes, but [38] proposed efficient methods of implementing this

idea locally, in the context of the *narrow band* methods, while Strain [50] has developed fast tree-based methods.

In the following we focus on the approach introduced in Sussman et al. [36] and discussed and updated in Russo and Smereka [3]. It is based on solving the following *reinitialization PDE*:

$$\frac{\partial \phi}{\partial t} = sgn(\phi^0)(1 - |\nabla \phi|) \tag{4.0.1}$$

$\phi^0(x) = \phi(x, 0)$ is the initial level set, which has to be changed into a signed-distance function, such that the zero level set is preserved. When this equation is solved up to time T, then $\phi(x, T)$ is the signed-distance function for all the points within distance T from the interface. This is apparent once we rewrite the equation in a scalar convection form: $\frac{\partial \phi}{\partial t} + sgn(\phi^0)n \cdot \nabla \phi = sgn(\phi^0)$ and observe that the transport speed is 1. Here $n = \nabla\phi/|\nabla\phi|$ is the unit field normal to the interface and the sign function is the usual one:

$$sgn(x) = \begin{cases} -1, & \text{if } x < 0; \\ 0, & \text{if } x = 0; \\ 1, & \text{if } x > 0. \end{cases}$$

The signal propagates away from the interface, as illustrated in figure 4.1.

In the following we take a close analytical look to the various artifacts associated with various numerical solutions to the reinitialization equation 4.0.1, and present an approach that does not move the zero interface even upon repeated application, unlike previous schemes.

4.1 The problem in one dimension and its solution

We examine now the Sussman Smereka Osher [36] reinitialization approach. The first-order 1D version used in [36] is given by

$$\phi_i^{n+1} = \phi_i^n - \triangle t S(\phi_i^0) G(\phi)_i \tag{4.1.1}$$

where

$$G(\phi)_i = \begin{cases} max(|a_+|, |b_-|) - 1, & \text{if } \phi_i^0 > 0; \\ max(|a_-|, |b_+|) - 1, & \text{if } \phi_i^0 < 0. \end{cases}$$

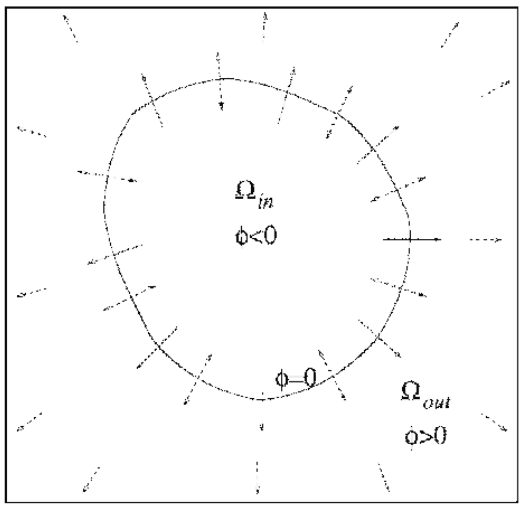

Figure 4.1: **Propagation of the signal off the zero level set. The arrows represent the unit normal to the zero level set. The dashed arrows represent the direction of propagation of the signal. See also [3].**

and

$$\begin{aligned} a \equiv D_x^- \phi_i &= (\phi_i - \phi_{i-1})/\triangle x \\ b \equiv D_x^+ \phi_i &= (\phi_{i+1} - \phi_i)/\triangle x \end{aligned}$$

and, for any real number h, we define $h_+ = max(h, 0)$ and $h_- = min(h, 0)$. The smoothed sign function S is given by

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + \triangle x^2}}$$

[3] pointed out that, even though the above scheme has been used apparently successfully in several contexts, it suffers from the artifact illustrated in Fig. 4.2: after repeated iterations, even though the local gradient converges to 1 in absolute value, the zero of the level set is being moved and may converge towards the closest grid point. In the following we analyze theoretically why and when this happens.

Let us assume without loss of generality that the level set function is increasing in the interval $[i-1, i+1]$ and has a zero between $i-1$ and i. In this case, a and b are

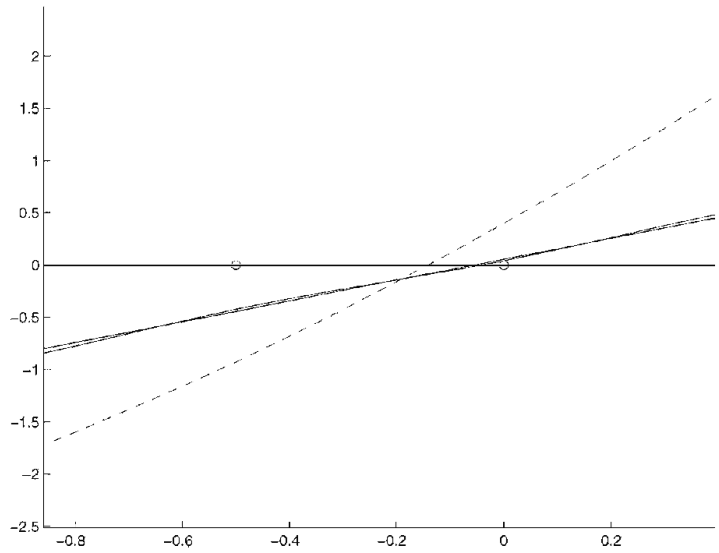

Figure 4.2: **The figure shows how the zero of the initial level set function (dashed) moves towards the closest grid point. See also [3].**

positive numbers, and we have

$$G(\phi)_i = \begin{cases} (\phi_i - \phi_{i-1})/\triangle x - 1, & \text{if } \phi_i^0 > 0; \\ (\phi_{i+1} - \phi_i)/\triangle x - 1, & \text{if } \phi_i^0 < 0. \end{cases}$$

Consequently, we obtain the following two equations for the nodes $i-1$ and i:

$$\begin{aligned} \phi_i^{n+1} &= \phi_i^n(1-h) - \phi_{i-1}^n h + \triangle t \\ \phi_{i-1}^{n+1} &= \phi_{i-1}^n(1-h) - \phi_i^n h - \triangle t \end{aligned}$$

where we used the notation $h = \triangle t/\triangle x$. We also assume for the moment that the sign of the level set at the nodes i and $i-1$ does not change when solving the above numerically, hence the zero doesn't exit the interval $[i-1, i]$. We will see that this is not necessarily true. These two equations form a coupled system of the form

$$\Phi^{n+1} = A\Phi^n + b \tag{4.1.2}$$

where $\Phi^n = \begin{bmatrix} \phi_{i-1}^n \\ \phi_i^n \end{bmatrix}$, $A = \begin{pmatrix} 1-h & h \\ h & 1-h \end{pmatrix}$, and $b = \triangle t \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. The iteration 4.1.2 converges if and only if the eigenvalues of the matrix A (1 and $1-2h$) are

less than 1 in absolute value, in other words we need $h \in [0,1]$, which is just the expected CFL condition for unit speed. Let us note that $A^n b = \triangle t(1-2h)^n b$ and that A^n converges (in the conditions above) to the matrix $\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$. Consequently we can rewrite 4.1.2, after computations, as

$$\Phi^{n+1} = A^{n+1}\Phi^0 + \triangle t \frac{1-(1-2h)^{n+1}}{2h} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{4.1.3}$$

It is now visible that one obtains at the limit

$$\begin{aligned} \phi_{i-1}^{\infty} &= \phi_{avg}^0 - \triangle x/2 \\ \phi_{i}^{\infty} &= \phi_{avg}^0 + \triangle x/2 \end{aligned} \tag{4.1.4}$$

to be the converged solutions at the nodes $i-1$ and i, where we denoted $\phi_{avg}^0 = (\phi_{i-1}^0 + \phi_i^0)/2$. One readily observes that the numerical gradient between the level set values at the two nodes equals one. One can also formulate the following:

Proposition 4.1.1. *If $|\phi_{avg}^0| < \triangle x/2$ and $0 < h < 1$ the system 4.1.2 converges, and the zeros of the level set don't move outside their initial intervals.*

Indeed, if $|\phi_{avg}^0| < \triangle x/2$ then, clearly, $sgn(\phi_{i-1}^{\infty}) = sgn(\phi_{i-1}^0)$ and the same for the index i. Furthermore, it is visible that, if $|\phi_{avg}^0| > \triangle x/2$, the level set value at the node i *will* change sign and become negative for a large enough n, at which point the system 4.1.2 will (have to) be solved on the interval $[i, i+1]$. Based on the above analysis but without getting into a rigorous proof, we can deduce that, depending on the value of the level set at the node $i+1$, the level set will either:

1. cross back to the initial interval $[i-1, i]$, using a similar mechanism with the one that pushed it out of it to begin with, and oscillate about the node i, or
2. stay within the new interval, or
3. travel further on to the next interval

As a note regarding the example used in [3], in figures 2 and 3 of that paper, in that case one has $\phi_{avg}^0 = -0.2625$ and $\triangle x/2 = 0.25$ hence, by Proposition 4.1.1, the zero

will move *out of the interval*, and converge to $x = 0$. Following the explanation given in [3] one may get the wrong impression that the convergence is one sided, whereas the mechanism must be of an oscillatory nature, as outlined in the first of the three options above.

To conclude this discussion, the zero of the level set may move during the reinitialization process as outlined above. Furthermore, using our results 4.1.4 one can see, after a few computations, that the zero will *always* move upon reinitialization, unless the numerical gradient $(\phi_i^0 - \phi_{i-1}^0)/\triangle x$ equals either 0 or 1. Russo and Smereka provided in their paper [3] a method to fix this. They correctly identified the problem as stemming from an incorrect use of upwinding, namely from evaluating the gradient *across* the interface. Their fix consists in modifying the computation of the discrete gradient at the nodes of an interval containing a zero of the level set. Their computation is based on the better approximation of the G function in 4.0.1 with

$$G(\phi)_i = |D_x^{up}\phi_i|,$$

where the upwind derivative $|D_x^{up}\phi|$ of the function ϕ near the interface is given by

$$|D_x^{up}\phi| = \begin{cases} \phi_i/|D_i|, & \text{if } \phi_i^0\phi_{i-1}^0 < 0; \\ -\phi_i/|D_i|, & \phi_i^0\phi_{i+1}^0 < 0. \end{cases}$$

where D_i is an approximation of the signed distance function from the interface to the ith node. To get a picture of the situation, note that the left derivative at point 4 in fig. 4.3 is given by ϕ_4/D_4, where D_4 is the approximation of the distance function computed using the original level set function ϕ^0 (the length of the thick segment in the figure). These considerations lead to their final (RS) scheme:

$$\phi_i^{n+1} = \phi_i^n - \triangle t \begin{cases} \frac{sgn(\phi_i^0)|\phi_i^n| - D_i}{\triangle x}, & \text{if } \phi_i^0\phi_{i+1}^0 < 0 \text{ or } \phi_i^0\phi_{i-1}^0 < 0; \\ sgn(\phi_i^0)G(\phi_i), & \text{otherwise.} \end{cases} \tag{4.1.5}$$

[3] proposes as a possible choice for D_i

$$D_i = \triangle x \frac{2\phi_i^0}{|\phi_{i+1}^0 - \phi_{i-1}^0|} \tag{4.1.6}$$

and shows simple examples in 1D and 2D that work well with this choice (we will see that in 2D the similar computation of D_i based only on level set values is not accurate

enough for practical purposes). Let us see if the scheme 4.1.5 moves the interface in 1D.

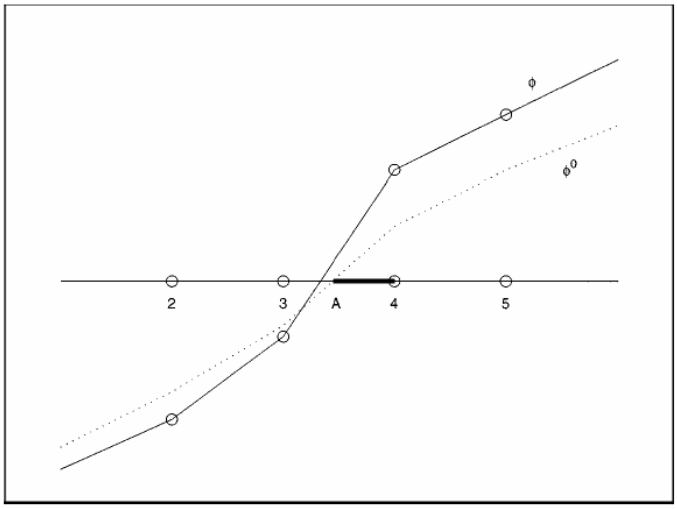

Figure 4.3: **Example that shows why the original Sussman Smereka Osher scheme is not truly upwinding. The dashed line represents the piecewise linear reconstruction of the original level set function. Point A represents the intersection of the latter with the x axis, and the length of the thick line is the approximation of the distance function at point 4. See also [3].**

Assume, again without loss of generality, that $\phi_i^0 > 0 > \phi_{i-1}^0$, so that we have that $\phi_i^{n+1} = \phi_i^n - \frac{\triangle t}{\triangle x}(|\phi_i^n| - D_i)$ and $\phi_{i-1}^{n+1} = \phi_{i-1}^n - \frac{\triangle t}{\triangle x}(|\phi_{i-1}^n| - D_{i-1})$. Furthermore, assume that $\phi_i^n > 0 > \phi_{i-1}^n$ also holds up to n. In this case, both the equations above become (we write down the one for i): $\phi_i^{n+1} = \phi_i^n(1-h) + hD_i$ with $h = \frac{\triangle t}{\triangle x}$. By induction we obtain

$$\phi_i^{n+1} = \phi_i^0(1-h)^n + \frac{(1-h)^n - 1}{1-h-1}hD_i = \phi_i^0(1-h)^n + (1-(1-h)^n)D_i \qquad (4.1.7)$$

But, by the definition of D_i, ϕ_i^0 and D_i (this also holds for index $i-1$) have the same sign, hence any affine combination of them (we assume again that $0 < h < 1$) has that same sign, consequently ϕ_i^{n+1} has the same sign and, by induction, the sign is preserved for all n. Hence we can pass to the limit and we obtain $\phi_i^\infty = D_i$ and $\phi_{i-1}^\infty = D_{i-1}$, hence *the initial signed distances are preserved*! The above considerations prove then the following:

Proposition 4.1.2. *If $0 < h < 1$ then, for any node i neighboring a zero of the level set, the numerical scheme 4.1.5 converges and the limit verifies $\phi_i^\infty = D_i$.*

Of course, the local gradient on a "zero interval" is $D_i - D_{i-1} = \pm 1$, as wished. Also, as an immediate corollary, we observe that the zero of the level set does not move, indeed. Let us notice however, that we essentially reached the conclusion that the Russo-Smereka algorithm updates the level set values ϕ_i in the zero intervals with the signed-distance values D_i, and nothing more. This can be done, of course, in 0 iterations! In section 5.2.2 we will look at a fast implementation for redistancing that makes use of this observation in two dimensions.

4.2 The problem in two or more dimensions

4.2.1 Discussion of the Russo-Smereka scheme

As noted by Russo and Smereka, in two or more dimensions, the redistancing errors are more severe, and marching in fictitious time in order to find the solution to the reinitialization PDE will lead to area loss (fig. 4.4). Their solution to the problem consists in naturally extending their 1D scheme to two dimensions:

$$\phi_{i,j}^{n+1} = \begin{cases} \phi_{i,j}^n - \frac{\triangle t}{\triangle x}(sgn(\phi_{i,j}^0)|\phi_{i,j}^n| - D_{i,j}), & \text{if } (i,j) \in \Sigma_{\Delta x}; \\ \phi_{i,j}^n - \triangle t sgn(\phi_{i,j}^0) G(\phi_{i,j}), & \text{otherwise.} \end{cases} \tag{4.2.1}$$

where the set $\Sigma_{\Delta x}$ defines the points which are within one grid point from the zero level set, namely the grid points (i,j) which verify

$$\phi_{i,j}^0 \phi_{i-1,j}^0 < 0 \text{ or } \phi_{i,j}^0 \phi_{i+1,j}^0 < 0 \text{ or } \phi_{i,j}^0 \phi_{i,j-1}^0 < 0 \text{ or } \phi_{i,j}^0 \phi_{i,j+1}^0 < 0$$

One can see easily that the discussion we did in the one dimensional case holds as well for this one, and, if the reasonable assumption $0 < \frac{\triangle t}{\triangle x} < 1$ holds, we have:

Proposition 4.2.1. *For any node $(i,j) \in \Sigma_{\Delta x}$ the numerical scheme 4.2.1 converges, and the limit verifies $\phi_{i,j}^\infty = D_{i,j}$.*

This method solves indeed problems like the one visible in figure 4.4, but the choice they suggest

$$D_{i,j} = \frac{2\triangle x \phi_{i,j}^0}{[(\phi_{i+1,j}^0 - \phi_{i-1,j}^0)^2 + (\phi_{i,j+1}^0 - \phi_{i,j-1}^0)^2]^{1/2}} \tag{4.2.2}$$

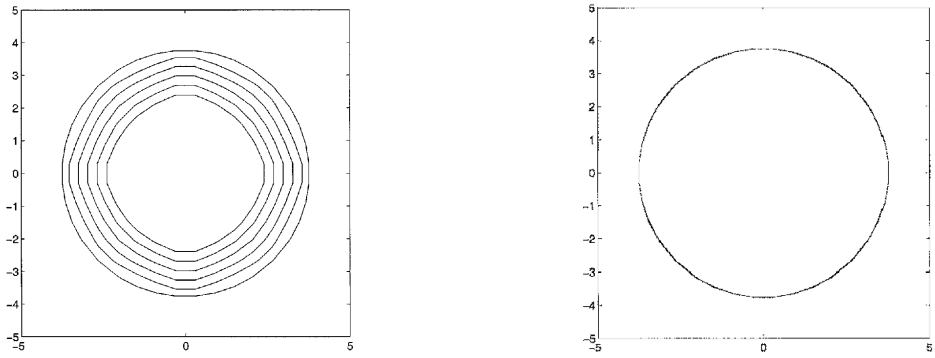

Figure 4.4: **In 2D the interface moves considerably and has spatial anisotropy. In the left figure we see how the zero level set of ϕ shrinks when the number of iterations is 0, 160, 320, 480, 640, 800. In the right figure the artifact is fixed using the Russo-Smereka fix. The domain is $[-5,5]\times[-5,5]$, and we took $\triangle t = \triangle x/2$ and $\triangle x = 10/16$.**

is not good enough for all practical purposes. Namely, the choice of the stencil isn't careful enough to stay close to the interface (again we should use upwinding), and errors can creep in. To illustrate what we mean, we run repeatedly the redistancing procedure 25, 50 and 75 times using the suggested choice for $D_{i,j}$, with 20 iterations for each redistancing (for this specific problem 20 iterations are enough to ensure a reasonably converged solution, namely a 10^{-6} relative error for ϕ). We also perform similar repeated applications of the redistancing algorithm 25 and 50 times for an upwind computation of $D_{i,j}$:

$$D_{i,j} = \phi^0_{i,j} / \begin{cases} \sqrt{max(|a_+|^2, |b_-|^2) + max(|c_+|^2, |d_-|^2)}, & \text{if } \phi^0_{i,j} > 0; \\ \sqrt{max(|a_-|^2, |b_+|^2) + max(|c_-|^2, |d_+|^2)}, & \text{if } \phi^0_{i,j} < 0. \end{cases} \tag{4.2.3}$$

where $a = D^-_x \phi_{i,j} = (\phi_{i,j} - \phi_{i-1,j})/\triangle x$, $b = D^+_x \phi_{i,j} = (\phi_{i+1,j} - \phi_{i,j})/\triangle x$, $c = D^-_y \phi_{i,j} = (\phi_{i,j} - \phi_{i,j-1})/\triangle y$ and $d = D^+_y \phi_{i,j} = (\phi_{i,j+1} - \phi_{i,j})/\triangle y$, and show the results in the last two pictures of figure 4.5 (the first one uses unsmoothed sign function, while the last one uses the smoothed one). (As a note, we also performed similar computations for $D_{i,j}$ being the exact distance to the linear reconstruction of the level set, and the results are qualitatively the same with the upwind simulation). We see that the choice 4.2.2 for the gradient approximation is unstable in such a case, while for the upwind choice 4.2.3

the zero level set does not leave $\Sigma_{\Delta x}$ and converges to adjacent nodes. Depending on the smoothing of the sign function (last picture) or lack of it (middle picture) we obtain dispersive or dissipative error.

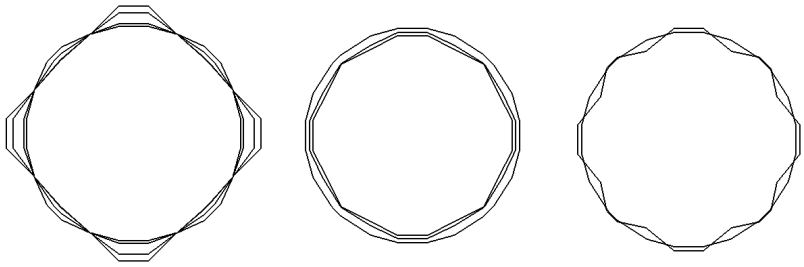

Figure 4.5: **Upon repeated application of the Russo-Smereka redistancing algorithm one can see how the error accumulates in time and again moves the interface. The first picture uses the $D_{i,j}$ from 4.2.2, the last two from 4.2.3. The number of repetitions is respectively 25, 50, and 75 for the first picture, 25 and 50 for the second one and 50 for the last one (which uses the smoothed sign function).**

Obviously the use of the upwind scheme is advantageous, even though still not perfect. What is the source of these errors anyway? This is easy to see if we consider the result of Proposition 4.2.1. In the Russo-Smereka scheme, the final value of the level set is replaced after each redistancing with the computed value of $D_{i,j}$. Unless the scheme ensures that, upon its application, the discrete approximation of the gradient norm of the new level set equals one, the interface will move. In a Navier-Stokes or simply in a moving interface simulation this artifact may become very visible when it dominates the dynamics, namely when the advection velocity is very small.

Let us look at a quick **example** that will shed some light on one of the mechanism that draws the interface towards grid points (refer to fig. 4.6). The initial value of the level set at the center point (i, j) is $\phi^0_{i,j} = 3$, and the other nodal values are as in the left picture. Using the upwinding algorithm we calculate up to two decimals (assuming $\triangle x = 10$): $D_{i,j} = 3/\sqrt{9^2 + 9^2} * 10 = 2.36$ and $D_{i-1,j} = D_{i,j-1} = -6/\sqrt{9^2 + 6^2} * 10 = -5.54$. These are in fact the new values of the level set at those nodes, and they

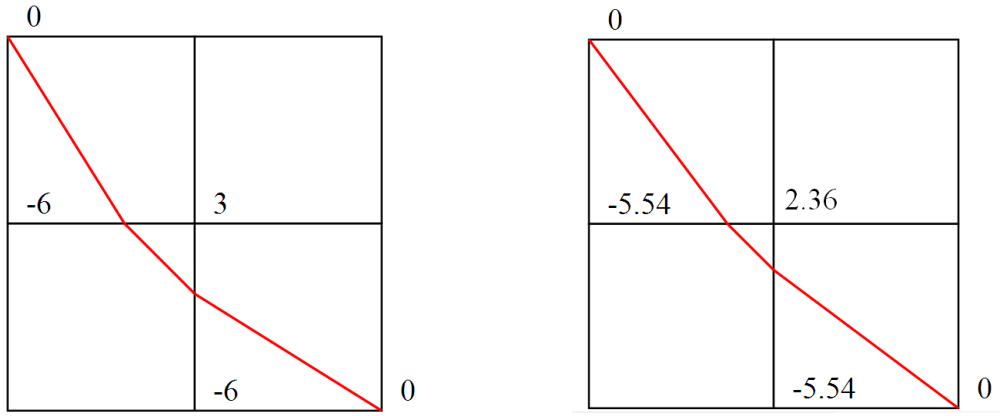

Figure 4.6: **Level set values before and after the reinitialization procedure is applied. This showcases one of the possible mechanisms that move the zero interface towards grid points.**

effectively pull the interface toward the center node. If one repeats the procedure, the new values would be 2.1 respectively -5.74, and the small value would get closer and closer to 0 as we repeat the iterations. If we look at the general case and replace the values 0,3 and -6 with some arbitrary values, then it becomes readily visible that, unless some magical equalities involving those values happen (namely, unless the cross-interface gradients along the grid directions are equal $|\phi^0_{i-1,j} - \phi^0_{i-1,j+1}| = |\phi^0_{i,j-1} - \phi^0_{i,j}|$ and/or $|\phi^0_{i-1,j} - \phi^0_{i-1,j}| = |\phi^0_{i,j-1} - \phi^0_{i+1,j-1}|$) the interface bordering (i, j) *will* move towards grid points (this is similar to what we observed in 1D about the Sussman [36] reinitialization). Repeated application of the scheme will increase the errors and will steadily displace the interface towards nodes.

In conclusion, the Russo-Smereka reinitialization scheme in two dimensions (or more) must be carefully supplied with upwind computations of the distances $D_{i,j}$, in order to provide stable results upon repeated use. Except for a few symmetric situations (read: zero probability for all practical purposes) the scheme *will* move the zero interface upon repeated application.

4.2.2 A fast alternative scheme

Before we look at a possible solution for these issues we should again note that Proposition 4.2.1 enables us to perform the RS-scheme in a fast manner, given also that we mainly are interested in maintaining unit gradients along the interface. Namely, we can simply set the values $\phi_{i,j}$ inside $\Sigma_{\Delta x}$ to be equal to $D_{i,j}$ (computed as in 4.2.3) and use those as boundary conditions for the rest of the iteration. This is similar to the high order fast marching method proposed by Chopp [51].The number of iterations need not be large at all, given that we iterate in the "less interesting" region outside $\Sigma_{\Delta x}$. The new scheme will be

$$\phi_{i,j}^{n+1} = \begin{cases} D_{i,j}, & \text{if } (i,j) \in \Sigma_{\Delta x} \\ \phi_{i,j}^{n} - \triangle t sgn(\phi_{i,j}^{0}) G(\phi)_{i,j}, & \text{otherwise.} \end{cases} \tag{4.2.4}$$

This scheme achieved similar results with the RS scheme 4.2.1 for the ellipse redistancing problem (see [3]) in less than half of the iterations (6 versus 16), and less than half the time (0.02sec vs. 0.045sec). Please refer to figure 4.7.

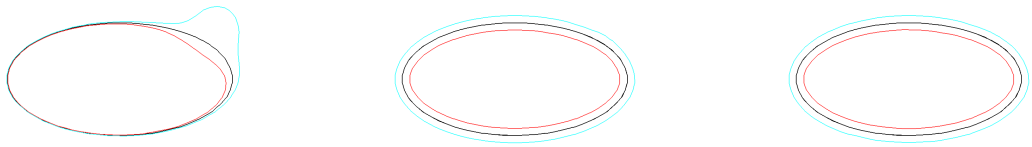

Figure 4.7: **In the ellipse redistancing test ([3]) the distorted level sets (left) are recovered after 16 iterations of the RS scheme (middle) and only 6 iterations using our fast scheme (right). The grid size is** 40×40**.**

4.2.3 A possible solution: reinitialization in the Marker Level Set framework

The second version of the Marker Level Set method introduced in section 2.3.3 takes advantage of marker placement along the interface to update the level set values in the interface neighborhood. This enables us to use the markers to compute the level set

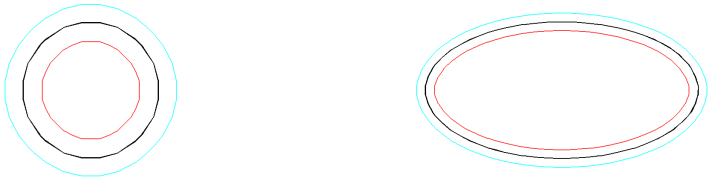

Figure 4.8: **Our scheme does a very good job at preserving the initial interface location while making the gradient a unit vector field throughout the domain. The pictures above show in black the initial level set and, superimposed, its final position after 1,2,3...500 applications of the redistancing scheme 4.2.5. In blue and red we pictured the $\pm\Delta x$ level sets. We used two markers per cell.**

values within $\Sigma_{\Delta x}$ and use those values as boundary conditions for solving iteratively outside $\Sigma_{\Delta x}$, similarly to what we do in equation 4.2.4. The new scheme becomes

$$\phi_{i,j}^{n+1} = \begin{cases} d_{i,j}, & \text{if } (i,j) \in \Sigma_{\Delta x} \\ \phi_{i,j}^{n} - \triangle t sgn(\phi_{i,j}^{0}) G(\phi)_{i,j}, & \text{otherwise.} \end{cases} \tag{4.2.5}$$

where $d_{i,j}$ are the level set values computed with the aid of markers. Before we explain how this is done, we present the graphic results in fig. 4.8. It is obvious that our scheme cures the previous artifacts and does the job. Now let us explain how we use the markers in order to determine the values of $d_{i,j}$. Assume that there are markers present in each of the interface cells (one can easily ensure this by placing them at zero crossings along each grid edge). Then, for each node $(i,j) \in \Sigma_{\Delta x}$ we compute $d_{i,j} = \phi_{i,j} - \lambda_{i,j}$, where $\lambda_{i,j}$ are the local corrections of the level set. These corrections are computed as a weighted average of the current level set values at the closest local markers:

$$\lambda_{i,j} = \sum_k w_k \phi(x_k) / \sum_k w_k \tag{4.2.6}$$

where x_k are the positions of the markers from a small neighborhood of (i,j) and w_k are weights associated to these markers and the node (i,j). $\phi(x_k)$ is the interpolated value of the level set function at the marker location. By performing the local correction of

the level set we effectively force its zero level to align with the markers. This provides excellent values for the level set in the neighborhood of the interface, which in their turn provide very accurate boundary conditions for the iteration outside $\Sigma_{\Delta x}$. The specific solution to the iteration problem is solved, as we saw earlier, along characteristics that move away from the interface, so that it does not modify the good values inside $\Sigma_{\Delta x}$ and ensures that our scheme 4.2.5 works quickly and accurately.

Chapter 5

Applications to simulation and animation of liquids.

In this chapter we will look at some direct applications of the MLS method to simulation of liquids. Firstly, as an interface tracker MLS can be combined with a Navier-Stokes solver in the same way the level set was used in Sussman et al. [36], and use as a free surface fluid simulator. Based on the two dimensional tests from chapter 3 we can expect that immediate advantages may ensue, for example one can get more detail than with the level set at even coarse resolutions, better mass conservation, and thin sheets and threads can be tracked better. Secondly, the surface markers have the capability of carrying tangential information, like texture. This feature can be regarded as the simulation of a liquid carrying a very shallow strip of surface paint, which is moved together with the surface and can change color due to mixing. We demonstrate how this works in several simulations below.

5.1 The fluid simulator

The fluid simulator uses the technology proposed by Sussman [9] and introduced to graphics by Mihalef et al. [52], but instead of CLSVOF (the coupled level set and volume of fluid method) we use MLS for interface advection. The MLS method's excellent motion characteristic tracking properties ensure that its mass preservation is also very good. We already saw in chapter 3 that in several standard 2D tests for surface trackers MLS preserved mass to a fraction of a percent, while in the physically realistic 3D simulations presented in this chapter, the mass loss was less than two percent for averagely sized computations (e.g. $64 \times 64 \times 64$ grids), and mass conservation is expected to improve even more when more resolved grids are used. Moreover, due to the same motion characteristic tracking built into MLS, its preservation of high-gradient or

high-curvature quantities (corners, for example) is better than CLSVOF's for any given resolution.

The fluid simulator consists of the Navier-Stokes simulator which advances the velocity, coupled with the MLS, which advances the level set. It consists of solving, at every time step:

- conservation of momentum equations
- conservation of mass equation
- MLS equations

We have already discussed the MLS equations in chapter 2. For the rest of the solver we use the inviscid incompressible form of the Navier-Stokes equations:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \rho \mathbf{g} \tag{5.1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5.1.2}$$

Here $\mathbf{u}$ denotes the velocity, p the pressure, ρ is the fluid density and $\mathbf{g}$ is the gravity vector. $\frac{D}{Dt}$ is the material (advective) derivative:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{5.1.3}$$

A second order upwind finite difference scheme is used to discretize the nonlinear advective terms. The vapor pressure is considered constant (one-phase formulation). The density ρ is controlled at any time step by the level set:

$$\rho = \begin{cases} \rho_{liquid}, & \phi \geq 0; \\ 0, & \phi < 0. \end{cases} \tag{5.1.4}$$

The Navier-Stokes equations are solved using the projection approach of Bell et al. [42]. One can find more information on the general setup of the solver, including possible addition of viscosity and surface tension terms, in [9].

5.2 Simulation results

We tested the power of the Marker Level Set method by running several 3D simulations on a 2.4GHz Core 2 Duo machine. The results are very encouraging judging from their

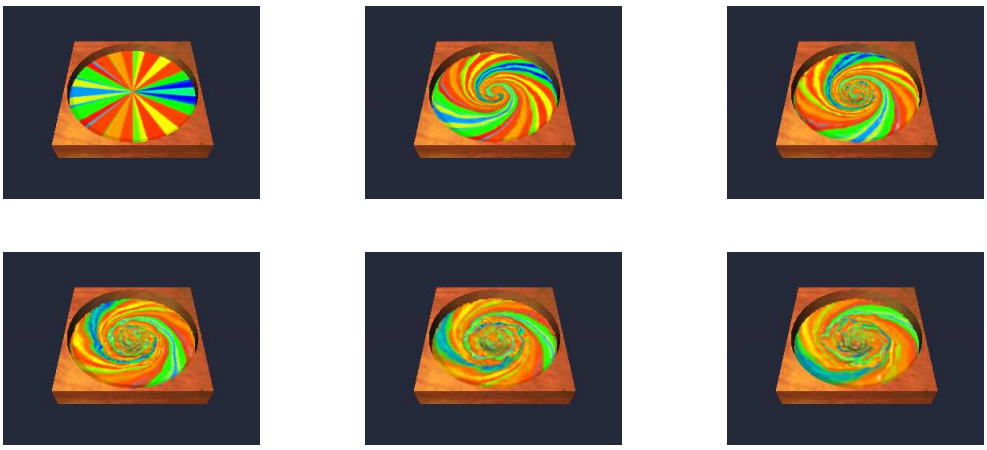

Figure 5.1: **Swirly water simulation. From left to right and top to bottom, time t = 0,1,2,3,4,5.**

accuracy and the display of the color advection capabilities of the MLS. Several of these results have already been published in our paper [53].

5.2.1 The swirl

In this simulation (Figure 5.1) we set up a small-depth body of liquid with a radial stripe texture on the surface and a radially-decreasing rotational velocity. This setup forces the liquid to twist and produces nice color mixing on the surface. Due to the higher center velocity and gravitational pull, the surface suffers a central depression. This forms a wavefront that is bounced back and forth to the circular walls several times, while the surface color continues to mix. The grid was fairly coarse, $64 \times 64 \times 16$, and one simulation step took about 45 seconds. The presence of the subpixel color information stored by the markers is essential for providing robust local color information, even in the presence of such a strongly distorting flow. In this simulation the average number of markers was 140 thousands, with a fairly small time variance.

5.2.2 A splashing liquid simulation

In this experiment a body of water is placed on top of a solid object. Due to gravity, waterfall-type dynamics ensues and the top water falls over the static water at the

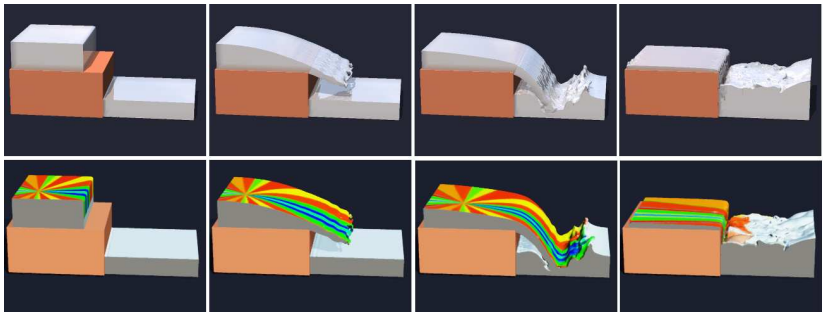

Figure 5.2: **Splashing liquid simulation with milk shader (first row) and marker color based shader (second row). From left to right, time t = 0, 0.8, 1.6, 6.4 seconds.**

bottom of the solid object, forming local overturning waves with various splashing effects, after which the dynamics tones down. The top surface markers are colored while the bottom ones are white. The experiment showcases successful dynamics with many topological changes, in which the colors vary continuously in time and space. The domain was discretized with a $96 \times 48 \times 48$ grid and one simulation step took about 120 seconds. The average number of markers used was 180K. The grid size is average, but one can already see that the waterfall sheet receives a good interfacial treatment, due to the excellent characteristics tracking of the markers. Upon reconnection, the surface markers are deleted by the MLS, due to becoming trapped inside the liquid, and this leads to the disappearance of several of the initial colors by the end of the simulation. We should note however that this is essentially a characteristic of the one-phase flow fluid simulator we use in this work and of the numerical discretization errors, rather than of the MLS. In the continuum limit, if the air bubbles would be safely restituted to the surface by the Navier-Stokes simulator, the colors would be also carried along to the surface by each bubble's surface markers.

5.2.3 A dam breaking problem

We set up a dam breaking problem, with the top half multi-colored and the bottom half using just one color. The water falls due to gravity and leads to the formation

of several overturning waves, entraining the surface texture as well (Figure 5.3). The power of the MLS is quite obvious: its level set component handles the normal dynamics of the interface, while its surface marker component also handles dynamics *tangential to* the interface (disregarded by the level set formulation). The standard attempt to advect surface textures volumetrically on the 3D Eulerian grid would lead to strong color diffusion very quickly, even for very simple flows. The MLS markers instead may introduce color diffusion only in the case of interfacial stretching that necessitates marker addition, and are thus much better suited to carry robustly surface information, for example color or transparency. In fact, if one has a very dense initial distribution of markers one is likely to suffer very little diffusion, or almost none (as the color Enright test shows). The average number of markers varied between 135-165K in this simulation.

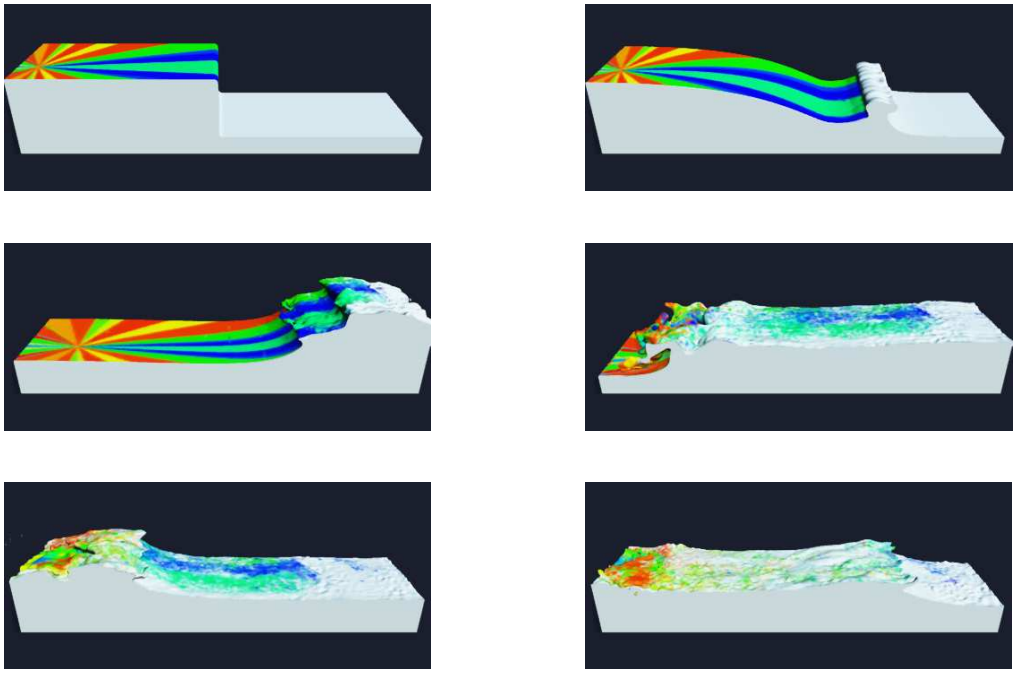

Figure 5.3: **Dam breaking simulation with matte rendering. From left to right and top to bottom, time t = 0, 1, 3.6, 6, 8.6, and 10.2 seconds.**

Regarding transparency, we note that one can start from the fully colored animations (fig. 5.3) and obtain for free new ones featuring partially transparent surfaces, by defining a chosen color to be transparent (fig. 5.4). Surely, partially transparent animations could be also obtained by advecting an extra transparency field with the markers, but this

would involve an extra variable. The domain had the physical dimensions $20m \times 5m \times 5m$, was discretized with a $128 \times 32 \times 32$ grid and one simulation step took about 75 seconds. It is apparent that even for such a coarse grid the results are quite satisfactory.

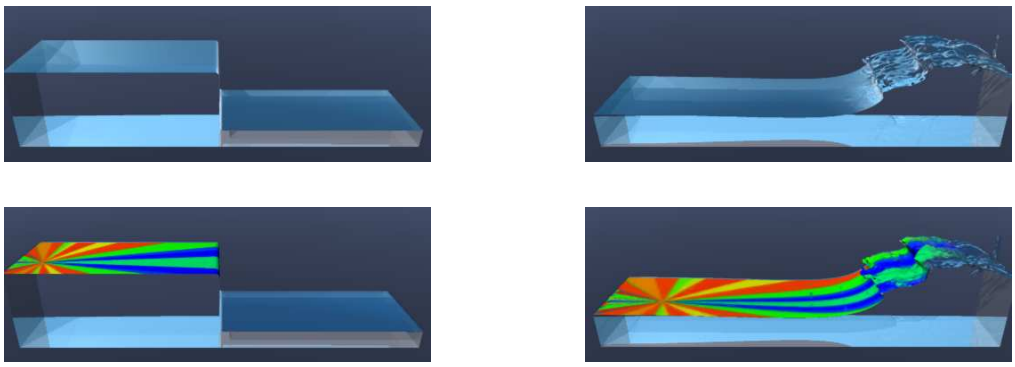

Figure 5.4: **Dam breaking simulation with transparent rendering. From left to right, time t = 0 and 3.6 seconds. First row: no color from markers. Second row: with surface color interpolated from markers.**

5.2.4 Small droplet and bubble generation in the context of MLS

One of the immediate advantages of the MLS method over Eulerian advection approaches is that its formulation lends itself naturally and easily to devising methods for spray and small bubble generation. This is due to the fact that the deleted markers are essentially an expression of the subgrid information lost by the Eulerian level set. As such, they may be treated, up to a certain approximation, as escaped liquid droplets (for markers deleted in the gas region) or gas bubbles (for markers deleted in the liquid region). In the following we will show an example that implemented such a treatment, and discuss choices for dealing with the dynamics of the escaped markers. Before we describe our simulations, let us note that Eulerian methods like the volume of fluid method could theoretically generate droplet and bubbles based on small volume fractions that are otherwise deleted as flotsam, but finding their location may be hampered (especially in on coarse grids) by the fact that the characteristic information is "hidden" by such methods, and consequently defining the position of the flotsam piece inside a cell is an unsolved (and possibly hard) problem.

We used a version of our dam breaking example as a basis for this experiment. After each time step of the original simulation we built a list of the deleted markers, using a boolean indicator to specify their nature (air bubbles or water droplets). After the initial Navier-Stokes simulation of the dam breaking was finished, we defined new dynamics for the generated droplets and bubbles.

Droplet dynamics

Once the droplets break free from the main body of liquid their motion may be approximated with ballistic dynamics (this is the method that we used). Namely, their initial velocity is defined to be their original velocity as interface markers, then the global position is updated by solving the inertial equation $\mathbf{x}^{new} = \mathbf{x}^{old} + \mathbf{u}(\mathbf{x})dt + \alpha \mathbf{u} dt + m\mathbf{g}\frac{dt^2}{2}$, where $\mathbf{x}^{old,new}$ are the old respectively the updated marker position, $\mathbf{u}(\mathbf{x})$ is the old marker velocity, α is the drag coefficient, $\mathbf{u}$ is the fluid velocity at the $\mathbf{x}^{old}$ marker position, m is the particle mass, $\mathbf{g}$ is the gravity vector and dt is the time step. The droplets are deleted once they reenter the body of liquid, which is easily detected by checking the sign of the level set at the droplet location. The drag coefficient may be defined to depend on the radius of the droplet, conforming to a Stokes like law, as described in the bubble dynamics below. Also, the use of drag requires that the simulation is performed in the whole domain, so that one has a valid (i.e. nonzero) air velocity. Surely, this is only a first approximation dynamics that may work in the air region for animation purposes, but one would need a more complex model for more realistic simulations. In particular, one needs to address the transfer of momentum from the water reentering droplets. One way to do this is to assign initial weight values to each droplet marker, and use them to modify the local density in the Navier-Stokes simulator.

Small bubble dynamics

We modeled the small bubble dynamics by regarding them as spherical bubbles in Stokes flow. Without getting into too much detail, bubble hydrodynamics theory says that after its formation, a bubble rapidly accelerates to its terminal velocity. This terminal velocity is determined by the balance between the buoyant rise force, and the drag force.This can be calculated for small, spherical bubbles, yielding Stokes' Law: $v_{Stokes} = \frac{2}{9}\frac{\mathbf{g}r^2}{\nu}$, where $\mathbf{g}$ is the gravity vector, r is the bubble radius and ν is the kinematic viscosity.

The Stokes velocity was added to the local fluid velocity (obtained by interpolation) to give the final marker velocity. In our simulation we used a (uniformly randomly generated) base radius of $1 \pm 0.5mm$ for the bubbles, and a viscosity coefficient of $10^{-6}cs$. There was no extra routine to handle bubble collision, yet the results of rising bubble cloud look quite realistic (figure 5.5).

5.3 Ray tracing with particles

All the 3D results presented in this thesis are rendered in the commercial software Vue D'Esprit, using the following routine. First, a cubic interpolated isosurface is extracted from each level set (the level set is discretized on the grid nodes). The isosurface is rendered using either a standard ray-tracer, without considering color properties, or an MLS-augmented ray-tracer, that uses the markers local to each ray intersection point as the basis for a kernel-based color integration. The kernels are similar to the ones used by the MLS itself for defining the color of the added markers. For rendering the partly transparent colored water without advecting extra transparency information we use the color info from the original animations. We choose a base color (very close to white, for our multicolored simulations) to be the "transparent" color, and resort again to local kernel interpolation for smoothing out the transparency regions (namely, the closer a color is to white, the more transparent it becomes). The values of the kernel coefficients are the same with the ones used by the MLS (see section 3.2.4).

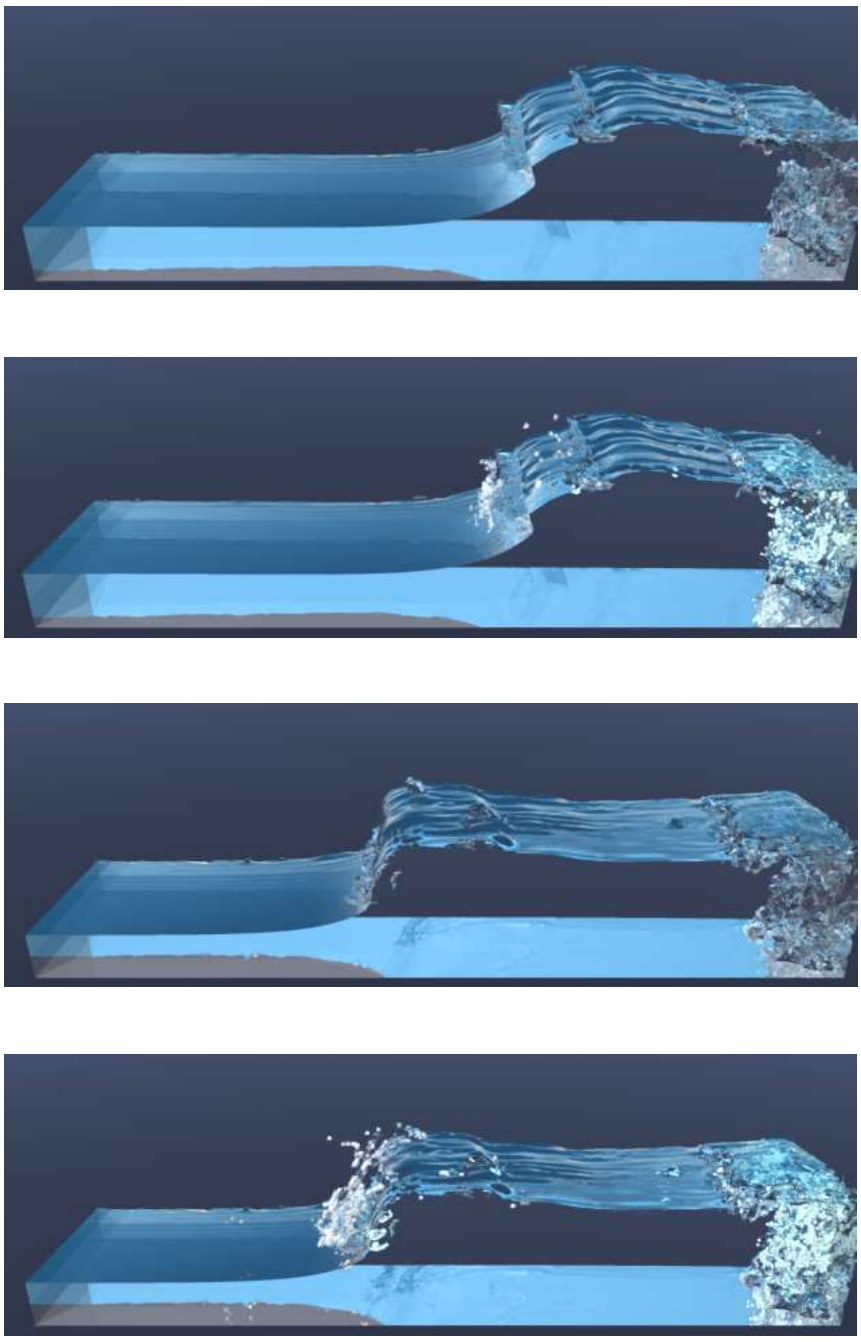

Figure 5.5: **Addition of spray and bubbles to a dam breaking simulation. Each original simulation image (first and third images) is followed by an image with spray and bubble effects added. First two images, time t = 2.6; next two at t = 3.2 seconds.**

Chapter 6

Conclusion and directions for future work

We presented in this thesis a novel method for surface advection which has the additional ability to handle surface texture (or any other interfacial tensor field) advection very accurately. This method, the Marker Level Set, combines in a unique way a level set and a set of markers placed on the interface, so that, for example, color information is carried along by the markers and texture advection is thus possible. From the two proposed MLS methods we chose the second one for full blown convergence tests that showed its numerical convergence with an order of accuracy equal to that of the marker advection, and also (because of its implementation simplicity) for 3D simulation of free surface flows. We interpreted physically the generated textured flows as liquid carrying a thin layer of paint, and we also showed an easy way to generate spray and small bubbles by identifying them with markers deleted in insufficient grid resolution regions.

We have discussed in the text several of the technical limitations of the MLS method. At a higher level, we should also mention that theoretically the MLS method may have difficulties in handling surface tension dominated effects, as instabilities may occur due to the markers having microscale resolution, whereas the fluid equations would be solved on a macroscale. However, the nice bubble and droplet simulation results obtained using the PLS method by Hong and Kim [54] provide a nice reference and the hope that MLS may succeed as well in attempting such simulations. Another issue that future extensions to the MLS method will need to solve is how to treat the impact between the surface markers describing a free surface and solid bodies.

Possible directions for future work may come from several directions. First, while

MLS is successful as both a sharp interface tracker and a texture advecting tool, we feel that there is room for improvement in the way the textures are rendered and initialized. Our marker-based rendering method may be updated with mesh-based functions that smoothly integrate the marker color to vertices, prior to rendering.

Another nice graphics/ocean engineering project could be to simulate realistically foam dynamics on ocean or sea surface. While MLS offers immediate support for the tangential dynamics, it must be complemented with a realistic mechanism for foam generation and disappearance. Figure 6.1 for example shows that an initial foam texture loses its fractal characteristics in the regions of high stretching soon after being advected, appearing diffused on the back of the wave. A possible solution to this problem may be offered by the MLS itself, by using its small droplet and bubble generation capability and modeling the dynamics of droplets and bubbles once they intersect the wave surface.

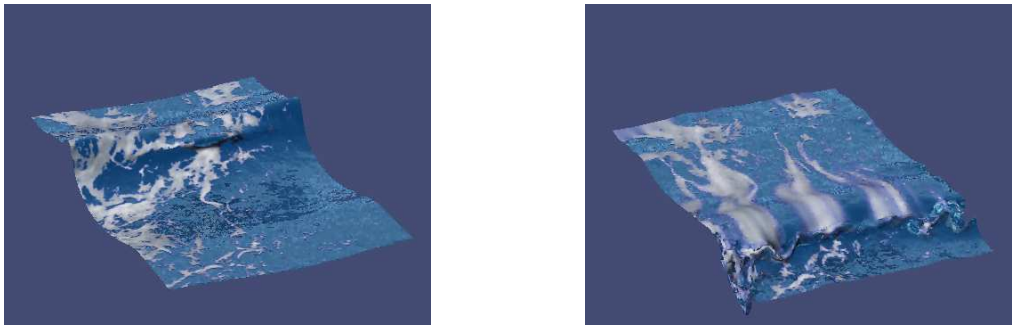

Figure 6.1: **Simulation of a breaking wave with a textured surface. Without adequate treatment the initial foam texture (left) would only stretch along the surface (right).**

Directly related to the droplet and bubble generation is an exciting computationally fluid dynamics project that would attempt to model the transition of fluid flow from a breakup region to spray region, and the subsequent dynamics using ballistics. Another CFD project should take advantage of the surface information carried by the markers, for simulations in which surfactant effects are important. The markers could track the

dynamics of a quantity of surfactant along the surface in a way implicit methods would not be able to. While front tracking methods would also perform well in such a task, the capability of the MLS of handling topology changes gives it an advantage over most front tracking methods.

Exciting directions for further research may come from the field of computer vision. Due to its accuracy, MLS could be used to improve the results obtained with the standard level set method, for problems in which corner or thin structure resolution is an important matter. The method can be also used to find skeletons of two or three dimensional closed regions. For example, one can make good use of the fact that the markers travel with the same velocity as the zero level set. If one starts with a closed 2D contour and advects it inwards with constant velocity, the moment a portion of the zero level set disappears, the associated markers land exactly on the skeleton of the contour. One can then stop the local velocity of the level set in the cells that contain markers but do not contain a portion of the zero contour, and thus obtain a very good approximation of the skeleton given by the markers. Note that the fact that in the MLS framework the markers *define* the zero level is crucial here, and a method that advects surface markers uncoupled with the zero of the level set would not give the true skeleton. This is because the markers would be advanced with the wrong velocity in the regions where they do not coincide with the zero level set. In figure 6.2 we show several encouraging preliminary results of 2D skeletons obtained with the approach described above. In particular our method handles simple (row 1 through 3) or multiple component (row 4) objects, regardless of genus. The method is easily extendable to three dimensions.

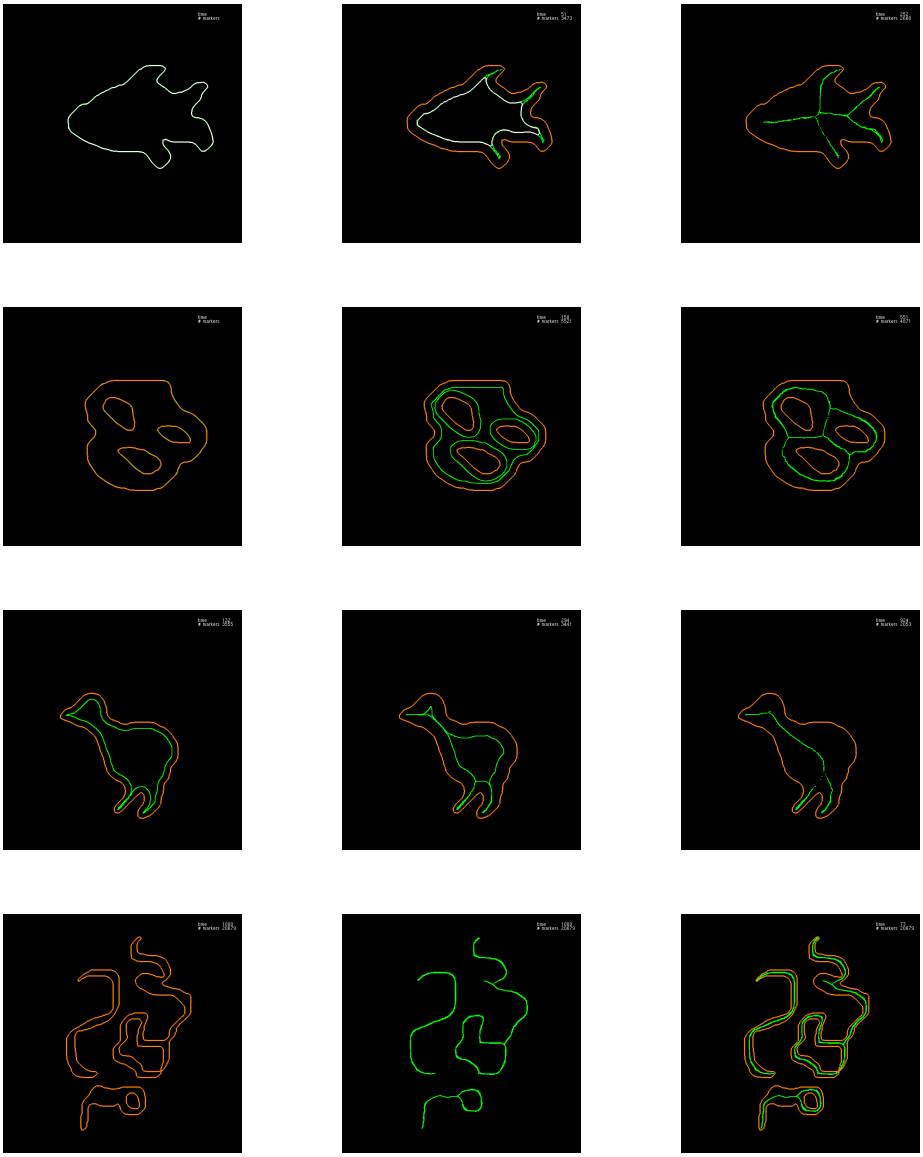

Figure 6.2: **Skeletonization of binary images (positive inside the contour, negative outside). Original contour in orange. Level set in white (row 1). Markers in green. Last column shows the initial contour + final skeleton. Row 1: Fish. Row 2: Genus 3 object. Row 3: Kiwi. Row 4: Objects with various genuses and separate connected components.**

References

[1] S. Hieber and P. Koumoutsakos. A Lagrangian particle level set method. *Journal of Computational Physics*, 210:342–367, 2005.

[2] 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.

[3] G. Russo and P. Smereka. A remark on computing distance functions. *Journal of Computational Physics*, 163:51–67, 2000.

[4] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge Univ. Press., 1996.

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

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

[7] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[8] M. Sussman and E. G. Puckett. A coupled level set and volume of fluid method for computing 3D and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162:301–337, 2000.

[9] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics*, 187:110–136, 2003.

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

[11] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35(10):995–1010, 2006.

[12] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, and John A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1), 2006.

[13] J. P. Pons, G. Hermosillo, R. Keriven, and O. Faugeras. How to deal with point correspondences and tangential velocities in the level set framework. *INRIA Rapport de recherché*, 2003.

[14] J.-J. Xu and H.-K. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. *Journal of Scientific Computing*, 19:573–594, 2003.

[15] P. Witting. Computational fluid dynamics in a traditional animation environment. *Proceedings of ACM SIGGRAPH 1999*, 2:129–136, 1999.

[16] Jos Stam. Stable fluids. *ACM SIGGRAPH 1999*, pages 121–128, 1999.

[17] F. Neyret. Advected textures. *Proceedings of SIGGRAPH/Eurographics Symposium on Computer animation*, pages 147–153, 2003.

[18] J. Stam. Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG)*, 22(3):724–731, 2003.

[19] N. Rassmusen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. *Proceedings of SIGGRAPH/Eurographics Symposium on Computer animation*, 4:193–2002, 2004.

[20] M. Wiebe and B. Houston. The tar monster: Creating a character with fluid simulation. *Proceedings of ACM SIGGRAPH 2004 Sketches and Applications*, 2004.

[21] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics*, 25(1):151–175, 2006.

[22] V. Kwatra, D. Adalsteinsson, N. Kwatra, M. Carlson, and M. Lin. Texturing fluids. *Tehnical sketches program, ACM Siggraph*, 2006.

[23] A. Bargteil, F. Sin, J. E. Michaels, T.G. Goktekin, and J.F. O'Brien. A texture synthesis method for liquid animations. *Proceedings of SIGGRAPH/Eurographics Symposium on Computer animation*, pages 345–351, 2006.

[24] O. Unverdi, S. and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. *Journal of Computational Physics*, 100:25–37, 1992.

[25] S. Shin and D. Juric. Modeling three-dimensional multiphase using a level contour reconstruction for front tracking without connectivity. *Journal of Computational Physics*, 180:427–470, 2002.

[26] P. E. Raad and R. Bidoae. The three-dimensional eulerian-lagrangian marker and micro cell method for the simulation of free surface flows. *Journal of Computational Physics*, 203:668–699, 2005.

[27] E. Aulisa, S. Manservisi, and R. Scardovelli. A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking. *Journal of Computational Physics*, 197:555–584, 2004.

[28] M. Muller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 Symposium of Computer Animation*. ACM SIGGRAPH/Eurographics, ACM Press, 2005.

[29] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. *Eurographics*, 22(3):401–410, 2003.

[30] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutre, and M. Gross. A unified Lagrangian approach to solid-fluid animation. *Proceedings of Eurographics Symposium on Point-Based Graphics*, 2005.

[31] J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, 2005.

[32] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *Transactions On Graphics*, 2007.

[33] A. J. Chorin and J. E. Marsden. *A mathematical introduction to fluid mechanics*. Springer-Verlag, New York, 1979.

[34] J. Strain. Semi-lagrangian methods for level set equations. *Journal of Computational Physics*, 151:498–533, 1999.

[35] D. L. Chopp. Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics*, 106:77, 1993.

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

[37] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A pde based fast local level set method. *UCLACAM Report*, pages 98–125, 1998.

[38] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269, 1995.

[39] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *Physics of Fluids*, 8:212–218, 1965.

[40] W. Rider and D. Kothe. A marker particle method for interface tracking. In *Proceedings of the 6th International Symposium on Computational Fluid Dynamics*, pages 976–981, 1995.

[41] S. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31:335–362, 1979.

[42] J. Bell, P. Colella, and H. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.

[43] R. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33:627–665, 1996.

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

[45] P. Smolarkiewicz. The multi-dimensional crowley advection scheme. *Monthly Weather Review*, 110:1968–1983, 1982.

[46] E. Aulisa, S. Manservisi, and R. Scardovelli. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *Journal of Computational Physics*, 188:611, 2003.

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

[48] S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving Stefan problem. *Journal of Computational Physics*, 135:8, 1997.

[49] B. Merriman, J. Bence, and S. Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112:334, 1994.

[50] J. Strain. Tree methods for moving interfaces. *Journal of Computational Physics*, 151:616, 1999.

[51] D. L. Chopp. Some improvements of the fast marching method. *Journal of Scientific Computing*, 23(1):230–244, 2001.

[52] V. Mihalef, D. Metaxas, and M. Sussman. Simulation and control of breaking waves. In *Proceedings of the 2004 Symposium of Computer Animation*. ACM SIGGRAPH/Eurographics, ACM Press, 2004.

[53] V. Mihalef, D. Metaxas, and M. Sussman. Textured liquids based on the Marker Level Set. *Eurographics*, 26(3), 2007.

[54] J.-M. Hong and C.-H. Kim. Discontinuous fluids. In *Siggraph 2005*, ACM Proceeedings. ACM, ACM Press / ACM SIGGRAPH, 2005.

Vita

Viorel Mihalef

Education

1990-1995 Attended University of Iasi, Iasi, Romania. Bachelor of Science in Mathematics.

1997-2001 Graduate work at University of Pennsylvania, Philadelphia, PA. Master of Philosophy in Mathematics, Master of Arts in Engineering.

2001-2007 Graduate work at Rutgers University, New Brunswick NJ. Doctor of Philosophy in Computer Science.

Publications

2004 Viorel Mihalef, Dimitris Metaxas, Mark Sussman - Animation and control of breaking waves, Proceedings of the 2004 Symposium of Computer Animation, Grenoble, France.

2005 Greg Slabaugh, Viorel Mihalef, Gozde Unal - A contour based approach to 3D text labeling on triangulated surfaces, Proceedings of the 5th International Conference on 3-D Digital Imaging and Modeling (3DIM), Ottawa, Canada.

2006 Viorel Mihalef, Betul Unlusu, Dimitris Metaxas, Mark Sussman, Youssuf Hussaini - Physics-based boiling simulation, Proceedings of the 2006 Symposium of Computer Animation, Vienna, Austria

2007 Viorel Mihalef, Dimitris Metaxas, Mark Sussman - Textured Liquids based on the Marker Level Set, Proceedings of Eurographics 2007, Prague, Czech Republic.

2008 Viorel Mihalef, Samet Kadioglu, Mark Sussman, Dimitris Metaxas, Vassilios Hurmusiadis - Interaction of Multiphase Flow with Animated Models, accepted by Graphical Models.