A Dynamic Reconstruction Method for Interface Tracking

Trevor Caldwell, Noah Duncan — Harvey Mudd College
Tengyuan Liang — Peking University
Shirley Zheng — Cornell University

Advisors: Jinsun Sohn, Joseph Teran
UCLA

August 5, 2011

Abstract

In this paper, we present work towards a novel method for tracking a moving interface. The interface is represented using particles, which are sampled via a grid-based interpolation scheme. The particles are advected with a second order Runge-Kutta scheme, and a signed-distance representation of the interface is then reconstructed with particle information. This is done by first computing the distance to the interface with fast sweeping, and then calculating sign information using an edge detection function. We periodically reseed particles to stabilize particle density and population. Our algorithm is easier to implement than comparable methods, and gives second-order accuracy. Finally, we present some preliminary results in two spatial dimensions.

Contents

1 Introduction

Interface tracking methods have many applications in fields such as image processing, computer vision, fluid dynamics, and computer graphics. Traditional level set methods based on an implicit representation of the interface include HJ-(W)ENO methods for spatial discreization, and Runge-Kutta schemes for temporal discretization ([4]). Despite the accuracy provided by these high-order methods, they suffer from a stability-based CFL criterion and numerical diffusion in regions of high curvature or in long, thin filamentous regions. Explicit particle methods do not have this time step restriction, but there is no way to handle topological changes such as pinching or merging, and it becomes impossible to determine the inside and outside regions relative to the interface. Thus, we need a method capable of preserving small-scale features and accurately computing distance, as well as keeping the correct sign information and handling topological changes.

Several methods have been used to try to improve these basic schemes, but each has its own advantages and disadvantages. Reinitialization has been used to keep the implicit level set function as a signed distance function, but this still suffers from similar problems such as CFL stability restrictions ([4]). The semi-Lagrangian method ([5]) is able to skirt these time step restrictions, but suffers from numerical diffusion and produces first-order accuracy. Hybrid methods have been attempted as well, most notably particle level set methods ([1], [2]). These schemes give promising results, but are very difficult to implement and are not robust due to the need for problem-specific reseeding strategies.

Because of the difficulty inherent in simultaneously maintaing accurate distance and sign information, we look for another alternative to these hybrid methods. Grid-based particle methods ([3]) are capable of accurate distance computations but do not have an effective criterion for maintaing sign information. However, the fast sweeping methods in [6] are capable of paritioning a domain into inside and outside regions relative to the ineterface. Using this partition for sign information, accurate distance from grid-based particle methods, and reseeding strategies to handle topological changes, we devise an algorithm for tracking a moving interface with second order accuracy in distance and correct sign information. Below we detail the background information necessary to understand the existing methods, and then we detail our algorithm and some preliminary results.

2 Background

2.1 Basic Level Set Methods

For level set methods ([4]), an interface $\partial\Omega$ bounding a set $\Omega^- \subset \mathbb{R}^n$ is implicitly captured as the zero isocontour of a higher dimensional function $\phi(\vec{x}, t) : \Omega \to \mathbb{R}$. For simplicity, we define Ω^+ as $\mathbb{R}^n \setminus \overline{\Omega^-}$; this sign convention makes Ω^- the inside region and Ω^+ the outside region. In general, the implicit function ϕ

defined on all $\vec{x} \in \Omega$ has dimension n, while the isocontour defining the interface has dimension $n-1$. The level set function also satisfies the following conditions

$$\begin{aligned} \phi(\vec{x},t) &= 0 \text{ on } \partial\Omega \\ \phi(\vec{x},t) &< 0 \text{ in } \Omega^- \\ \phi(\vec{x},t) &> 0 \text{ in } \Omega^+, \end{aligned}$$

where $\vec{x} \in \Omega$ and $t \in \mathbb{R}^+$.

We can also use the implicit function ϕ in order to evolve the interface. We know that $\phi(\vec{x},t) = 0$. From the chain rule, we have

$$\phi_t + \vec{v} \cdot \nabla\phi = 0, \tag{1}$$

where $\vec{v}(\vec{x})$ is the velocity at a point $\vec{x}$ on the implicit interface - this provides an Eulerian formulation of the interface evolution. It is often convenient to initialize the zero isocontour to be a signed distance function with $|\nabla\phi| = 1$. This ensures that the level set is smoothly varying (i.e. no steep gradients), allowing for accurate spatial approximations. Reinitialization algorithms based on Eulerian advection (discussed below) or fast marching methods can be used to maintain a signed distance function by solving the reinitialization equation

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

Solving (2) periodically maintains the signed distance function property, which preserves a smoothly varying level set and allows for simple calculations of geometric quantities such as the unit normal

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|},$$

and the curvature

$$\kappa = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|} \right).$$

There are standard Eulerian advection algorithms that can be used to solve (1) and (2). Hamilton-Jacobi ENO schemes can be used to extend basic first-order upwinding schemes to higher-order spatial accuracy. This is done by using the smoothest possible polynomial interpolation of ϕ, constructed via a table of divided differences. It is common to use the third divided differences in order to construct a polynomial yielding a third-order approximation; this leaves three possible HJ ENO approximations for each spatial derivative based on how the stencil is chosen. One can also use the weighted ENO (WENO) method, which uses a convex combination of the three ENO approximations in order to obtain even higher-order accuracy in smooth regions. To obtain a higher-order temporal discretization, one can use Runge-Kutta (RK) schemes. These work by taking successive standard Euler steps and combining the averaged results with the initial data with a convex combination. It is most common to use a third-order accurate RK method, which uses three Euler steps and two averaging steps to produce the approximation to ϕ after one time step. Although these methods are accurate in general due to their high-order nature, they suffer from a stability-based CFL condition and require computationally costly adaptive schemes to resolve small scale features. With insufficient grid resolution, level set methods are prone to numerical diffusion, especially in regions with high curvature or in long, thin filaments.

2.2 Additional Level Set Methods and Particle Based Methods

There are several other level set methods and particle based methods that avoid some of the shortcomings of the standard level set methods. The simplest approach is to use an explicit parameterization of the interface and populate particles along the parameterized interface. If finding an explicit parameterization proves to be too difficult, then one could use a grid-based interpolation scheme based on the initial interface (this sort of seeding method will be described in detail in the description of our algorithm). Once the particles have been placed, they are advected via the evolution equation

$$\frac{d\vec{x}_p}{dt} = \vec{v}(\vec{x}_p),$$

where $\vec{x}_p$ denotes the particle position and $\vec{v}(\vec{x}_p)$ is its velocity. This can be solved with standard Runge-Kutta methods, where the size of the time step is based on the degree of accuracy desired. There is no restrictive stability criterion. However, purely particle-based methods cannot handle topological changes to the interface, as there is no convention for connecting the interface and determining the correct sign information.

One way to bridge the gap between Eulerian and Lagrangian perspectives is to employ semi-Lagrangian advection methods ([5]), which treat grid points in a particle-like manner. These work by tracing solutions backward along characteristic lines to find where a parcel originates from and interpolating to estimate the value of the interface at the grid points surrounding the origin point. This method relaxes the CFL restriction and allows arbitrarily large time steps because the backward trajectory calculation ensures that the numerical domain of dependence includes the solution's domain of dependence. Despite the efficiency of these methods, they still suffer from the numerical diffusion issues of traditional level set methods.

The Particle Level Set Method ([1], [2]) is one of the most popular ways to extract the positive attributes of both particle and level set methods. At each time step, the level set and and particles are simultaneously advected. Then, particles are used to correct the level set in under-resolved regions. This is done by determining if particles lie on the wrong side of the interface (positive particles inside the level set, or negative particles outside the level set) by more than a specified particle radius. Local level sets associated to the positive particles are used to rebuild ϕ outside of the interface, and the level sets for negative particles are used for the $\phi \leq 0$ region. After the initial particle correction, reinitialization is applied to maintain a smooth signed distance function; then, another particle correction procedure is used and the particle radii are adjusted according to the new level set values. However, the particle reseeding operation is problem dependent, and there is no general scheme that will work for all types of interfaces and flows. Also, the entire algorithm is notoriously difficult to implement, which hinders its use in outside applications.

The Grid Based Particle Method ([3]) uses Lagrangian particles to represent the interface with an underlying Eulerian grid. In this method, grid points are collected in a small computational tube about the interface, and the closest point on the interface is calculated for each grid point in the tube. The particles are advected, and the interface is reconstructed locally using a least square fitting procedure. Using this reconstruction, the closest points to the interface are re-computed, resampling the particles. Finally, the computational tube is updated, and the underlying mesh is locally refined or coarsened in an adaptive manner. The use of the least squares fitting procedure provides a very accurate measure of distance, but the method does not include a reliable means of retaining sign information, which can give misleading results in some cases.

One final method of interest is the vartiational method employed in [6], which is used to reconstruct surfaces from sets of unorganized points. In order to estimate the surface, they apply a convex image segmentation model to a two-value image with edges located along the points. They use fast sweeping methods (see [7]) in order to find a distance function, which is then used as an edge detector. With this distance-based edge detector, they can then compute a binary image via another eikonal equation. After this, they use a variational image segmentation model to compute an implicit representation of the surface, and use smoothing methods and Bregman iteration in order to remove outliers and produce a smooth and accurate reconstructed surface. In our implementation, we found the smoothing steps to be superfluous, so we adopted the initial step of creating a binary image to the problem of aligning a level set with particles and determining the inside and outside sign information.

3 Outline of the Method

First, a general overview of the method is presented, and later, each compenent is discussed in detail.

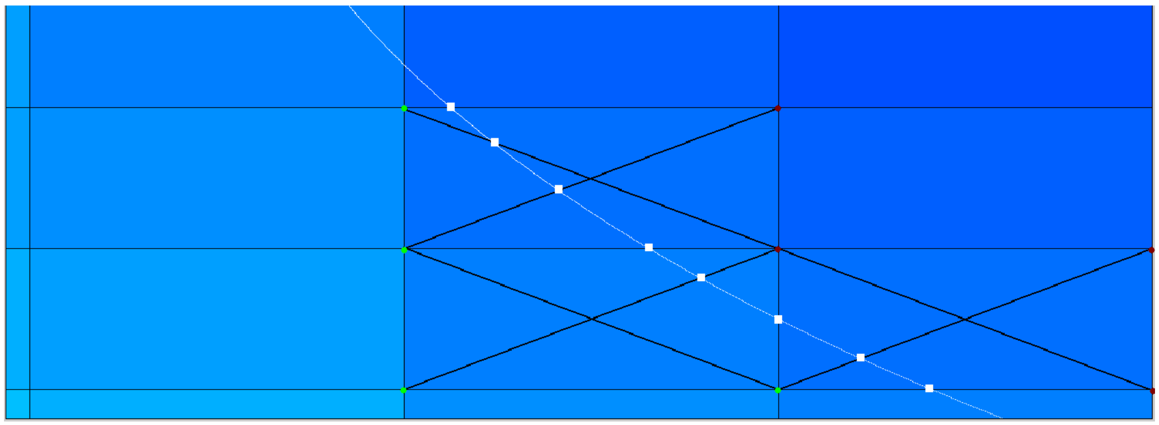


Figure 1: The green points mark exterior (positive) grid points, the red points mark interior (negative) grid points, and the white squares are the interpolated zero-crossings.

3.1 Initialization of the Particles

Our method begins with an initial level set function, ϕ, and seeds marker particles on the zero-isocontour before the interface is iteratively evolved. Conveniently, the reseeding and seeding schemes are identical, so the same function will applied periodically throughout the interface evolution steps. The (re)seeding procedure is explained below.

The (re)seeding scheme is a modified version of Marching Squares algorithm. Besides interpolating zero-crossings on the four edges of each cell, we also interpolate particles on the two diagonals.

A cut grid cell has two general configurations, that is, it has either an equal number of negative and positive vertices (2-2) or three vertices of one sign and one of another (1-3 and 3-1). For each cut grid cell, we linearly interpolate particles on the six edges (four edges and two diagonals) and mark them with particles. In the case that the level set information on a grid cell is zero, we apply a small perturbance term and interpolate as usual. Then, the interface is represented as the polygon obtained from the particles. Notice that this method yields two or three line segments in each cell compared to one in the marching squares.

The main advantage of this seeding scheme is the finer subcell detail achieved. This seeding scheme achieves finer subcell detail than the marching squares algorithm, as it resolves concave or convex information within each cell. The seeding and reseeding procedure are identical because both input level set information and output an explicit representation of the zero-isocontour. More regarding reseeding will be disgussed in section **3.4**.

3.2 Particle advection

We advect our particles in the velocity field using a second order TVD RK scheme. If the velocity field is only defined on grid points we obtain the velocity from bilinear interpolation between the 4 nearest grid points to the particle.

3.3 Reconstruction of the level set by wrapping

The key step of the method is the wrapping of the level set around the particles. This function takes particle locations as input and outputs a signed-distance representation of the interface. Once this process is complete, the zero isocontour of the level set should lie exactly on the particles. Thus, the Lagrangian information retained by the particles is transferred to the level set. In section **3.3.1**, a distance function is obtained by fast sweeping, and in section **3.3.2**, the signed distance function is computed from the distance function.

3.3.1 Compute Unsigned Distance Function Using the Fast Sweeping Algorithm

Compute distance on cut grid cells.

Given the new particle location after the advection, we want to get the exact distance on the grid points (grid points of the cut cell) near the zero isocontour. Quadratic fitting is used to get a higher order numerical approximation of distance near the new interface. The fitting is done locally at each cut grid cell.

The main procedure of computing distance on active cut grid cells is as following.

1. **Expand computational region.** For each cut grid cell, consider the extended 3×3 region consisting of the cell and eight neighboring cells.We refer to this as the fitting region.

2. **Change coordinates.** Compute the gradient, $(n_x, n_y) = \nabla\phi$ from the level set information. We transform the particles and corners into a new coordinate system where the normal vector of the corner defines the y-axis and the tangent vector defines the x-axis. We transform by multiplying all points by the linear transformation matrix:

$$\begin{pmatrix} n_y & -n_x \\ n_x & -n_y \end{pmatrix}$$

3. **Fit the curve.** Using all the particles in the fitting region, fit the quadratic curve according to the new coordinate system.

4. **Initialize distance values on cut grid cells.** Compute the distance function d_0 on the four corners of cut grid cell to the quadratic fit curve and update the distance on these points with the minimum of the initial distance and the computed distance.

Compute distance function across the domain by fast sweeping

Given an initial distance function d_0 defined in a narrow tube around the interface, we shall use the fast sweeping algorithm ([7]) to extend this Eulerian information to a distance function over the entire domain. We initialize d as follows: set $d(\bar{x}) = d_0(\bar{x})$ on all cut grid points. These values are fixed in later calculations. For all non-active grid points, set $d(\bar{x}) = C$, for any large $C \in \mathbb{R}^+$ greater than the maximum of the width and length of the domain. These values will be updated later.

In order to enforce the signed-distance property, set $f(\bar{x}) = 1$ for all $\bar{x}$,

$$|\nabla d| = 1. \tag{3}$$

Using a Godunov upwind difference scheme, discretize the domain:

$$[(d^h_{i,j} - d^h_{x\,\min})^+]^2 + [((d^h_{i,j} - d^h_{y\,\min})^+)]^2 = h^2 \tag{4}$$

for $d^h_{x\,\min} = \min(d^h_{i-1,j}, d^h_{i+1,j})$, $d^h_{y\,\min} = \min(d^h_{i,j-1}, d^h_{i,j+1})$, and $(x)^+ = \max(x, 0)$. For boundary points, approximate $|\nabla d|$ using one-sided difference.

As detailed in ([7]), iteratively sweep the domain, alternately sweep the domain:

$$(1) i = 1 : I, j = 1 : J \quad (2) i = I : 1, j = 1 : J,$$

$$(3) i = I : 1, j = J : 1 \quad (4) i = I : 1, j = J : 1,$$

where I and J are the horizonal and vertical resolutions, respectively. We then update d at each grid point with the minimum of its current value and the computed solution of (2).

After k applications of the fast sweeping algorithm, we take d to be the distance function over the entire computational domain. In our experiments, we use $k = 10$.

3.3.2 Compute sign informatin using fast sweeping

The interior is segmented from the exterior by a wrapping process. The wrapping begins with an initial level set that lies outside the particles, and then solves an eikonal equation to align the level set with the particles. If the particles lie in the positive region of the level set (i.e. the particles lie outside of the interior), the interior will deteriorate until it disappears. Hence we obtain a signed distance function ϕ_i whose zero-isocontour encloses the particles by extending the level set from the previous time step ϕ_p in the outward normal direction. Since the previous level set is a signed distance function we can extend it simply by subtracting the extension length, D_e.

$$\phi_i = \phi_p - D_e \tag{5}$$

To move the level set inwards to the particles we use the idea of edge detection, from image processing. In the edge detection problem, given a scalar image I, we find points on I that denote edges. To find these points we apply an edge detector function to I that is a function of ∇I. The function takes low values near an edge. A typical edge detector function is:

$$g(x) = \frac{1}{|\nabla I(x)|^p + \epsilon} \tag{6}$$

In our situation, we want to solve the inverse problem. That is, given a set of points that denotes edges we want to generate the corresponding image. So we write the edge detector equation in terms of the image gradient:

$$|\nabla I| = \frac{1}{g(x)^{1/p} + \epsilon} \tag{7}$$

For our edge detector function $g(x)$ we use $g(x) = d(x)$ where $d(x)$ is the distance to the nearest particle. This gave us the eikonal equation:

$$|\nabla I| = \frac{1}{\epsilon + d(x)^{1/p}} \tag{8}$$

We found that choosing $p = 1/4$, a pseudo-timestep of $dt^{1/p}$, and $\epsilon = dx^{1/p}$ obtained the best results across several test problems.

Solving this eikonal equation by the fast sweeping method ([7]) and segmenting at a critical value c, we obtain a two-valued function I such that $I(\bar{x}) = 1$ for $\bar{x} \in \Omega^+$ and $I(\bar{x}) = -1$ for $\bar{x} \in \Omega^-$. Finally, update the signed distance function ϕ:

$$\phi = d \cdot I \tag{9}$$

3.4 Reseeding of the particles

Assume that given the previous input level set function ϕ, we want to seed/reseed the particles around the zero isocontour. There are two advantages of reseeding. Firstly, with reseeding we can handle topological change and easily avoid the problem that the particles become so sparse which invalid the image-segmentation(or wrapping) algorithm. Secondly, it can judge the number of particles dynamically according to the shape of the surface, which keeps the density of particles the constant during the advection. A robust strategy for removing and adding particles throughout the simulation is necessary for any hybrid method. Simply

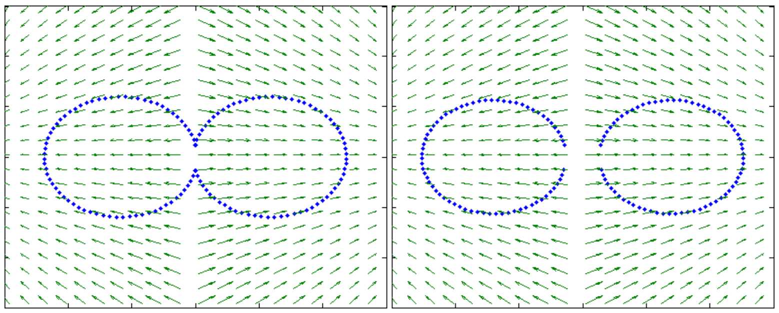


Figure 2: Strain will separate particles, which may cause the interior to be miscalculated as exterior.

placing a large amount of particles around the initial level set at the beginning of the simulation is not only computationally inefficient but unstable when a velocity field separates the particles and opens the interior to the exterior. Figure 2 shows such a scenario. In general it is impossible to know in advance that the dumbbell shape will be torn into two circles.

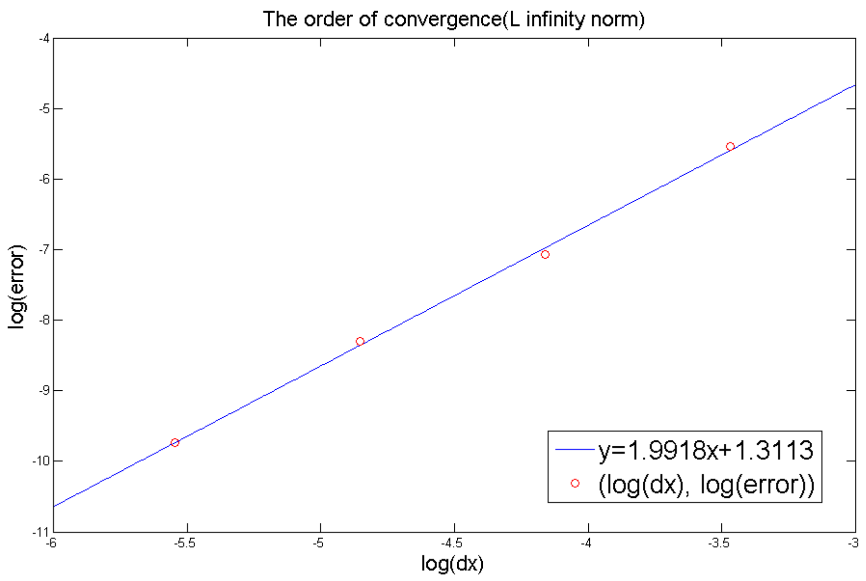
The reseeding algorithm is virtually identical to the seeding algorithm and takes as input the signed distance function computed by fast sweeping.

4 Test Problems

4.1 Convergence test for Translating Circle

We establish that our method is second order accurate for the test of translating a circle at constant unit velocity in two dimensions. We take the L_∞ norm of the error between the analytical signed distance function and the computed signed distance function across the interface cells.

Grid Size	Error (L_∞ norm)	Order
50	0.0018	N/A
100	5.1632e-004	1.8016
200	1.2485e-004	2.0480
400	3.0582e-005	2.0294



4.2 Deforming Vortex

A standard test for an interface tracking method's ability to preserve volume is the deformation of a circle into a thin spiral by a single vortex. In two dimensions, the velocity field for the vortex is given by:

$$v_x = -\cos(\pi y)\sin(\pi y)\sin(\pi x)^2$$

$$v_y = -\cos(\pi x)\ sin(\pi x)\sin(\pi y)^2$$

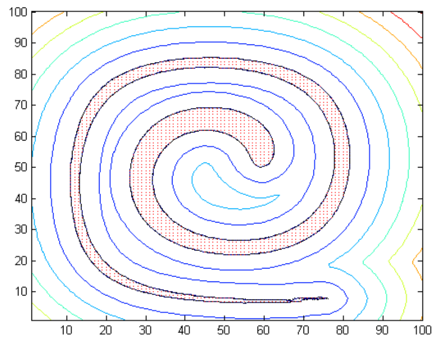
We start with a circle of radius 0.15 located at (0.5, 0.75) and run until T =0.5, then reverse the velocity and run until T = 1. The area losses for 3 grid sizes are

Grid Size	Area Lost	% Area Loss
50	0.0632	5.8
100	0.0669	2.55
200	0.0688	1.27

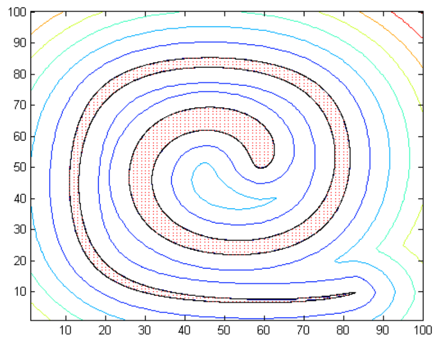
As shown in Figure 3, we see that the new algorithm keeps a good tail compared to the particle method, and far better than the 3rd ENO in space step and 3rd Runge−Kutta in time step with reinitialization. Furthermore, our method is much faster because the maximum number of particles during the advection is 2052, however the number for the pure particles method is 10000.

4.3 Merging Cirlces

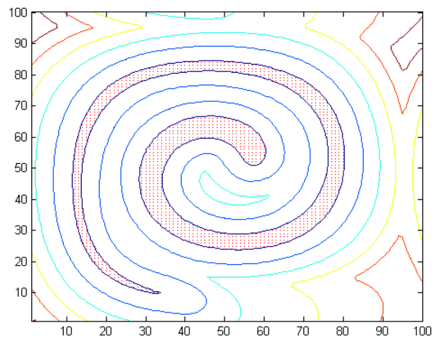
We confirm the ability of our method to handle topology change by testing the merging of two circles. Merging is handled naturally, as no parameters were adjusted to enable successful merging. Figure 4 shows



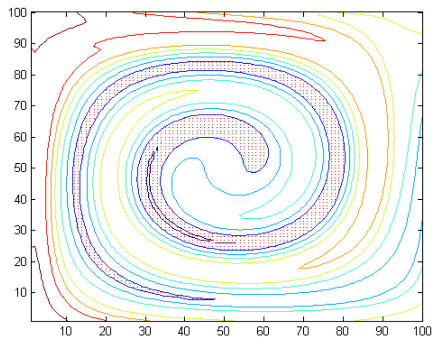
a



b



c



d

Figure 3: The spiral test is done in resoluation 100×100 and run until time $T = 2$. (a) is the dynamic reconstruction method with reseeding period equal to 10, the particle number at the end is 2052; (b) is the particle method, the initial number of particles is 10000; (c) is the 3rd order ENO in space step and 3rd order Runge−Kutta in time step with reinitialization; (d) is the 3rd order ENO in space step and 3rd order Runge−Kutta in time step

the performance on a simple merge test.

5 Conclusion

5.1 Summary

Dynamic Reconstruction provides a means to achieve more numerical accuracy by utilizing particle advection to track the interface. Unlike pure particle methods, however, the method reseeds periodically to stabilize particle density and population. Furthermore, a level set representation of the interface is constructed at each iteration, thus allowing for topological changes.The scheme produces second-order accuracy in the translation test.

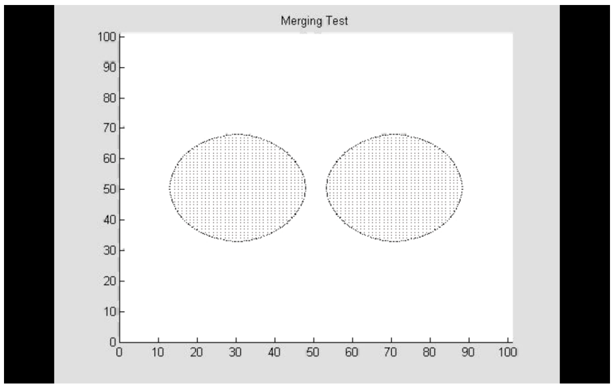


Figure 4.1

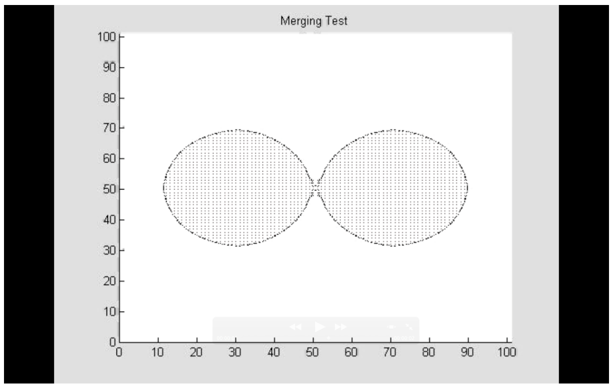


Figure 4.2

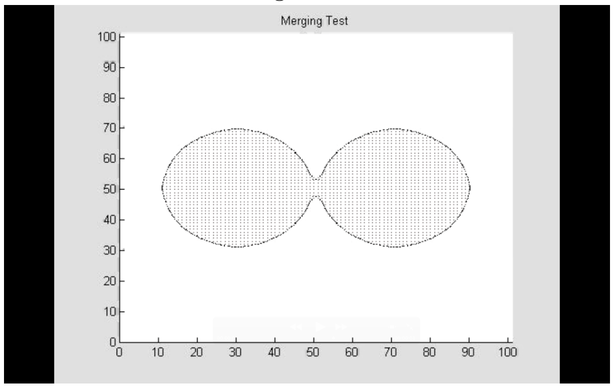


Figure 4.3

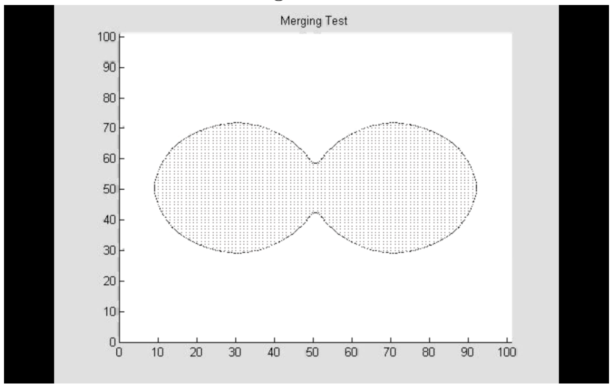


Figure 4.4

Figure 4: Circles expanding in the normal direction merge. No parameters were adjusted in advance.

5.2 Future Work

The volume in areas of high curvature is sensitive to the frequency with which we reseed. For instance, the period between reseedings is negatively correlated with the volume loss at the tail of the interface evolved by the vortex field. The frequency of reseeding thus is a parameter to be selected based on the velocity field. In future work, we hope to devise a consistent reseeding scheme that does not require a frequency parameter or depend on the velocity field.

In addition to reseeding, sign computation can be improved. Our scheme relies on fast sweeping to segment interior and exterior, however, the algorithm is not very accurate in concave regions, i.e. cusps/ dimples. The order of accuracy with respect to resolution is not uniform across the entire interface and varies depending on surface geometry, with higher accuracy in smoother regions. Therefore, improving sign will allow the method to more robustly handle pinching.

These improvements would make it sensible to implement our method in three dimensions, tested with a fluid velocity field.

6 Acknowledgements

We are extremely grateful to our advisors, Professors Jinsun Sohn and Joseph Teran for providing wonderful guidance and inspiration. Our project was made possible by Professor Andrea Bertozzi and the UCLA Applied Math REU. We plan to further develp the Dynamic Reconstruction Method and implement the method in 3D. Finally, big thanks to the reader, and we welcome all comments.

References

[1] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[2] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers & structures*, 83(6-7):479–490, 2005.

[3] S. Leung and H. Zhao. A grid based particle method for moving interface problems. *Journal of Computational Physics*, 228:2993–3024, 2009.

[4] S. Osher and R.P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153. Springer Verlag, 2003.

[5] J. Strain. Semi-lagrangian methods for level set equations. *Journal of Computational Physics*, 151(2):498–533, 1999.

[6] J. Ye, X. Bresson, T. Goldstein, and S. Osher. A fast variational method for surface reconstruction from sets of scattered points. UCLA CAM Report 10-01, 2010.

[7] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74(250):603–627, 2004.