A Grid Based Particle Method for Evolution of Open Curves and Surfaces

Shingyu Leung [a,*], Hongkai Zhao [a]

[a] *Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875.*

Abstract

We propose a new numerical method for modeling motion of open curves in two dimensions and open surfaces in three dimensions. Following the method we proposed in [15], we represent the open curve or the open surface by Lagrangian particles sampled according to the underlying fixed Eulerian mesh. The idea in the current paper is to represent and to track explicitly all end-points of the open curve and boundary points of the open surface. Then we extend the original algorithm in [15] to incorporate this extra condition on the boundary locations. As examples, we give applications to both spiral crystal growth modeling and image segmentation using active contours.

Key words: Open Curves, Open Surfaces, Eulerian underlying mesh, Lagrangian particles representation

1 Introduction

In a recent paper [15], we have proposed a novel approach to represent an interface and model its motion. The idea is to sample the interface by non-parametrized Lagrangian particles according to an underline uniform or an adaptive Eulerian mesh. This results in a quasi-uniform sampling of the interface. As the interface moves, we continuously update the location of these particles by solving ODE, rather than PDE. Using extra Lagrangian information defined on these sampling points, we can nicely capture topological changes such as merging or splitting of surfaces. To restore the quasi-uniform

[*] Corresponding author.
Email addresses: syleung@math.uci.edu (Shingyu Leung), zhao@math.uci.edu (Hongkai Zhao).

sampling of the interface, we have proposed a local least square fitting for local surface reconstruction.

We have successfully applied this technique to various velocity models, including the motion by an external velocity model, the motion in the normal direction, and the motion by mean curvature. Numerous test examples have been shown in [15] to demonstrate the flexibility of the approach.

The goal of this paper is to generalize the techniques in [15] to handle motions of an open curve in 2D or an open surface in 3D. The idea is to first keep track of the boundary (end-points of the open curve and a closed boundary curve of the open surface) and then incorporate this extra condition on the local reconstruction phase of the algorithm.

Various algorithms have been proposed for dealing with open curves/surfaces. One approach was the work of [20] for modeling spiral crystal growth. The idea was based on the level set method [18] where the author used the intersection of two level set functions to represent the codimension-two boundary of the open curve/surface. The curve/surface of interest was implicitly defined as the zero level set of one signed distance function at which the other one was positive, i.e. $S = \{x : \phi(x) = 0 \text{ and } \psi(x) > 0\}$. However, one had to define velocities for not only the level set function ϕ which represented the curve/surface of interest, but also the level set functions ψ which was used solely to define the codimension-two boundary. Moreover, the method proposed in [20] worked only for fixed-end curve and it is not clear how to define the velocity for ψ so that the evolution of the boundary satisfied certain given motion law. A generalization to this approach was proposed in [21] for constructing open surfaces from point cloud data. The method incorporated the method proposed in [4, 7] to allow motion of the boundary. Computationally, all these methods were not efficient since one has to solve not only one to get the implicit representation of the open curve/surface, but the number of PDEs, i.e. the number of the level set functions, equaled to the the codimension of the object.

Directly applying the level set method, [2] in the content of image analysis implicitly represented the open curves using the *centerline* of a level set function, i.e. the curve of interest is the zero level set. Numerically this is challenging since the level set function gives almost no zero value. To overcome this issue of numerical difficulties, the authors considered the μ-level set instead. But then the curve can never be recovered and there is always an $O(\mu)$ smoothed zone near the interface.

Another approach was the vector distance function method [11] which extended the signed distance function of the level set method. The vector distance function was defined as $\phi(\mathbf{x}) = \mathbf{x} - \mathbf{y}$ with $\mathbf{y}$ the closest point from

x to the interface. Therefore, this vector valued function embedded not only the distance and the inside/outside information, but also the normal vector. Unfortunately, this representation required solving the same number of PDE's to the dimension of the space where the object is living, independent of the codimension of the object. In particular, the method required solving three PDEs for modeling open/closed curves/surfaces in $\mathbb{R}^3$.

Similar to our representation, the vector level set method [22] modeled propagating crack using also closest points from an underlying fixed mesh. However, their motion law was imposed only at the tip of an open curve. No motion/reconstruction were considered elsewhere.

The advantages of our approach are multi-folded. First, the end-points/boundary points of the open curve/surface are accurately and explicit tracked. This is important for many physical applications. We will describe later in the example section two applications including the spiral crystal growth [5, 20] and image segmentation using active contours [13, 6]. Also, both the viscosity solution and the multivalued solution can be naturally handled in our algorithm using both the underlying Eulerian mesh together with the Lagrangian sampling particles. This gives a flexibility to control the change of the topology in the solution. Adaptivity can also be applied easily according to local dynamics and features. To resolve any fine structure in the interface, we might refine the underline Eulerian grid. The algorithm can be implemented easily since no PDE is involved on the underlying mesh. Our method will automatically put more sampling particles on these locations.

This paper is organized as follows. In Section 2, we will first summarize [15] and introduce important notions we will be using for the rest of the paper. In Section 3, we generalize this approach to motion of an closed curve in 3 dimensions. With that, we explain how to apply the method to model motions of an open interface in Section 4. Various examples from two dimensions and three dimensions are given in Section 5 to demonstrate the performance of our algorithm.

2 Grid Based Particle Method

In this section, we give a general recipe of the grid based particle method. For a complete description of the algorithm, we refer the readers to [15].

In [15] and also in this paper, we represent the interface by meshless particles which are associated to an underlying Eulerian mesh. In our current algorithm, each sampling particle on the interface is chosen to be the closest point from each underlying grid point in a small neighborhood of the interface. This one

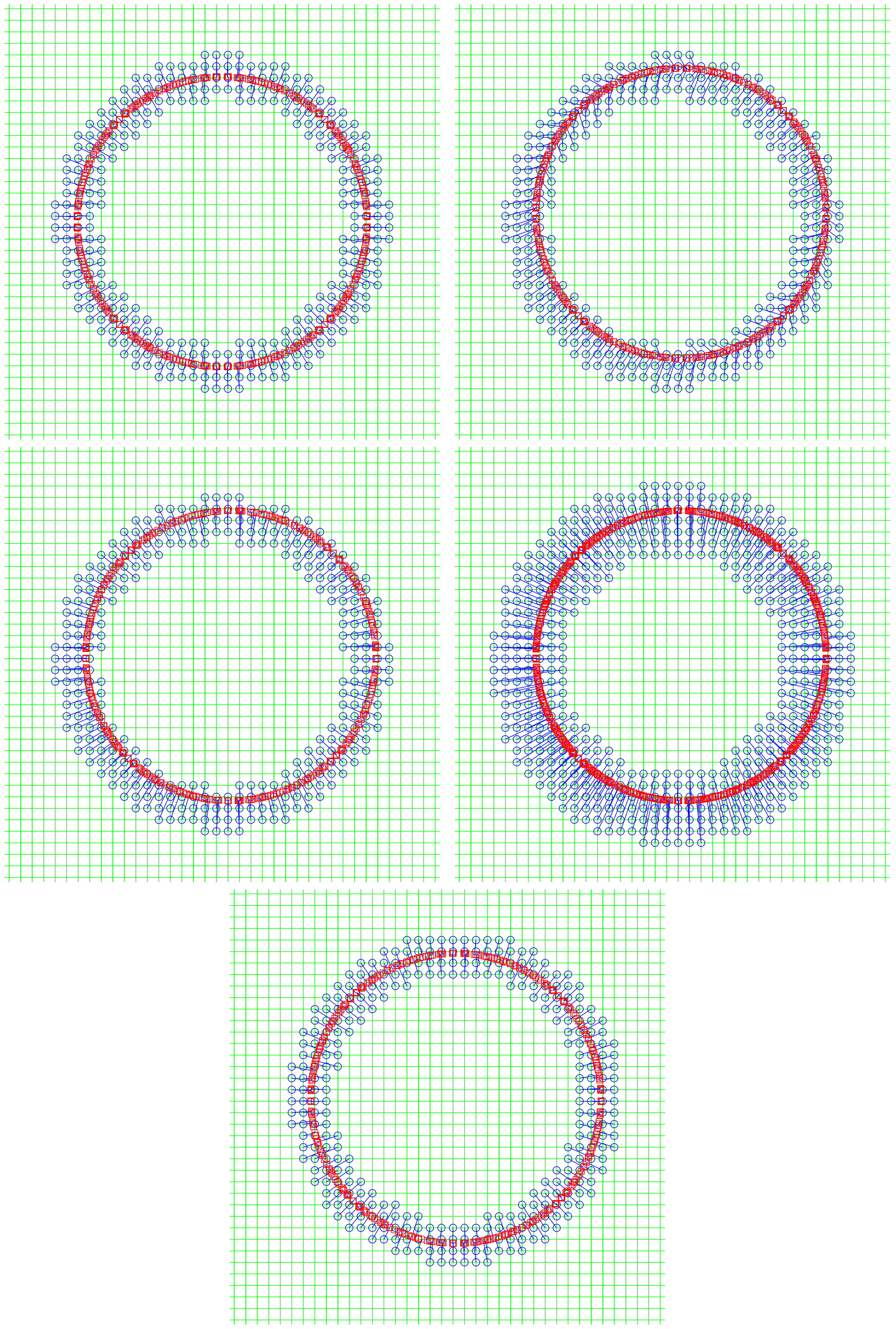

Fig. 1. Grid Based Particle Method. From left to right: (a) Initialization, (b) after motion, (c) after re-sampling, (d) after activating new grid points with their foot-points, (e) after inactivating grid points with their foot points.

to one correspondence gives each particle an Eulerian reference during the evolution. The closest point to a grid point, $\mathbf{x}$, and the corresponding shortest distance can be found in different ways depending on the form in which the interface is given.

At the first step, we define an initial computational tube for active grid points and use their corresponding closest points as the sampling particles for the interface. A grid point $\mathbf{p}$ is called **active** if its distance to the interface is smaller than a given **tube radius**, γ, and we label the set containing all active grids Γ. To each of these active grid points, we associate the corresponding closest point on the interface, and denote this point by $\mathbf{y}$. This particle is called the **foot-point** associated to this active grid point. This link between the active grid points and its foot-points is kept during the evolution. Furthermore, we can also compute and store certain Lagrangian information of the interface at the foot-points, including normal, curvature and parametrization, which will be useful in various applications.

As a result of the interface sampling, the density of particles on the interface will be roughly inversely proportional to the local grid size. This relation provides an easy adaptive approach in the current grid based particle method. In some regions where one wants to resolve the interface better by putting more marker particles, one might simply locally refine the underlying Eulerian grid and add the new foot-points accordingly.

This initial set-up is illustrated in figure 1. We plot the underlying mesh in solid line, all active grids using small circles and their associated foot-points using squares. On the left most sub-figure, we show the initial set-up after the initialization. To each grid point near the interface (blue circles), we associate a foot-point on the interface (red squares). The relationship of each of these pairs is shown by a solid line link.

To track the motion of the interface, we move all the sampling particles according to a given motion law. This motion law can be very general. Suppose the interface is moved under an external velocity field given by $\mathbf{u} = \mathbf{u}(\mathbf{y})$. Since we have a collection of particles on the interface, we simply move these points just like all other particle-based methods, which is simple and computationally efficient. One can simply solve a set of ordinary differential equations using high order schemes which gives very accurate location of the interface.

It should be noted that a foot-point $\mathbf{y}$ after motion may not be the closest point on the interface to its associated active grid point $\mathbf{p}$ anymore. For example, figure 1 (b) shows the location of all particles on the interface after the constant motion $\mathbf{u} = (1, 1)^T$ with a small time step. As we can see, these particles on the interface are not the closest point from these active grid points to the interface anymore. More importantly, the motion may cause those original foot-points

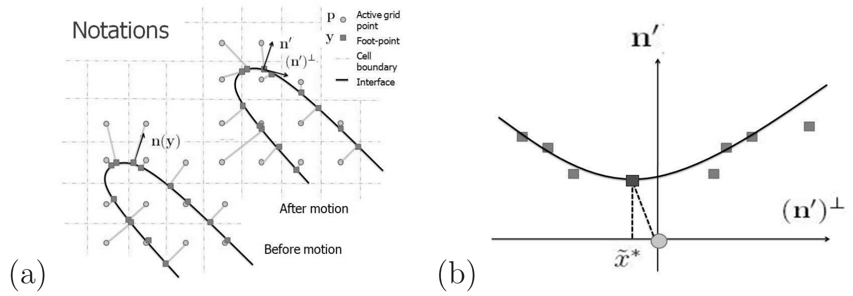

Fig. 2. (a) Definition of a local coordinates and (b) the way how we determine the new foot-point using a local least-square reconstruction of the interface.

to become unevenly distributed along the interface. This may introduce both stiffness, when particles are getting together, and large error, when particles are getting apart. To maintain a quasi-uniform distribution of particles, we need to resample the interface by recomputing the foot-points and updating the set of active grid points (Γ) during the evolution. During this resampling process, we locally reconstruct the interface, which involves communications among different particles on the interface. This local reconstruction also provides geometric and Lagrangian information at the recomputed foot-points on the interface.

The key step in this process is a least square approximation of the interface using polynomials at each particle in a local coordinate system, $\{(\mathbf{n}')^{\perp}, \mathbf{n}'\}$ with $\mathbf{y}$ as the origin, $\mathbf{n}'$ is the normal vector associated to $\mathbf{y}$ before motion, and $(\mathbf{n}')^{\perp}$ is the tangent vector, figure 2 (a). Using this local reconstruction, we find the closest point from this active grid point to the local approximation of the interface, figure 2 (b). This gives the new foot-point location. Further, we can also compute and update any necessary geometric and Lagrangian information, such as normal, curvature, and also possibly an updated parametrization of the interface at this new foot-point. For a detail description, we refer interested readers to [15] for each of the above procedures.

To end this section, we summarize the algorithm.

Algorithm:

(1) Initialization [Figure 1 (a)]. Collect all grid points in a small neighborhood (computational tube) of the interface. From each of these grid points, compute the closest point on the interface. We call these grid points **active** and their corresponding particles on the interface **foot-point**.
(2) Motion [Figure 1 (b)]. Move all foot-points according to a given motion law.
(3) Re-Sampling [Figure 1 (c)]. For each active grid point, re-compute the closest point to the interface reconstructed locally by those particles after the motion in step 2.

(4) Updating the computational tube [Figure 1 (d-e)]. Activate any grid point with an active neighboring grid point and find their corresponding foot-points. Then, inactivate grid points which are far away from the interface.
(5) Adaptation (Optional). Locally refine the underlying grid cell if necessary.
(6) Iteration. Repeat steps 2-5 until the final computational time.

---

3 Closed Curves in 3D

The original algorithm proposed in [15] can be applied to model motions of codimension two objects. In particular, we use it to model motions of a closed curve in 3D. In the original algorithm, we collect particles on the interface and use the normal/gloabl parametrization to classify them into group. The normal vectors are used to construct a local coordinate system to represent the curve (in 2D) or the surface (in 3D) as a graph. To model a curve in 3D, we can also repeat this procedure to locally reconstruct the curve and then determine the closest point from the grid point to this local reconstruction.

However, the local reconstruction procedure can be simplified by using the tangent vector, rather than the normal vector. For instance, we assume that we have already collected a set of points $\mathbf{y}_i$ and a tangent vector $\mathbf{t}$ for local reconstruction. Then we follow a similar procedure as in the three dimensional case to translate $\mathbf{y}_0$ t the original and to rotate the local coordinates such that $\mathbf{t}$ becomes the $(0, 0, 1)$-axis. By doing these, the curve in 3D can be expressed locally as a function of z, the third coordinate. Mathematically, we locally represent the curve using $(x(z), y(z), z)$. Next, we least square fitting $x(z)$ and $y(z)$, respectively, using quadratic polynomials. Finally, we compute the new foot-point associated to the active grid point $\mathbf{p} = (p_1, p_2, p_3)$ expressed with respect to the local coordinates by minimizing

$$\min_z \left\{ [x(z) - p_1]^2 + [y(z) - p_2]^2 + [z - p_3]^2 \right\} . \tag{1}$$

In the current implementation, we are using local quadratic polynomials for both $x(z)$ and $y(z)$ so that the minimizer z^* can be found explicitly by solving a cubic equation. Other Lagrangian information including the tangent vector, the normal vector, the global parametrization, and etc. can be obtained using the local reconstruction $(x(z), y(z), z)$ and z^* accordingly.

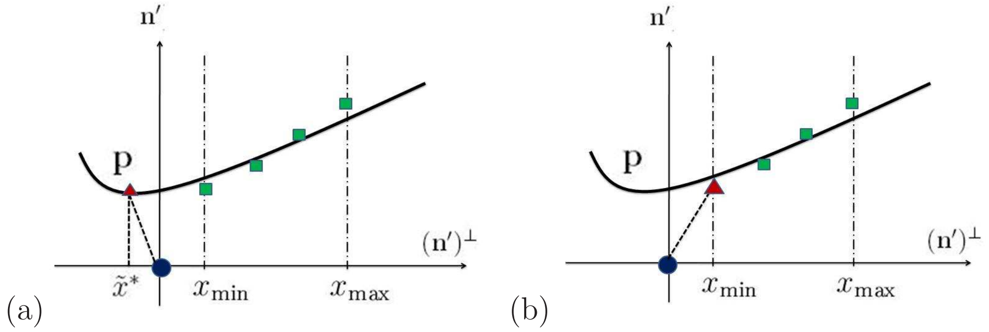

Fig. 3. (a) In the original algorithm, this potential new foot-point (red triangle) will be rejected since $\tilde{x}^* \notin [x_{\min}, x_{\max}] = [\min(x_i), \max(x_i)]$. (b) If $x_{\min}$ corresponds to an end-point of the interface, however, we associate the active grid point (blue circle) to the particle corresponding to $x_{\min}$ (red triangle).

4 Open Curves and Open Surface

In this section, we will discuss how our algorithm can model the motion of an open curve (in 2 dimensions) or an open surface (in 3 dimensions). The main idea is to explicitly keep track of the motion of the end-points of the open curve or the closed boundary of an open surface, and then to enforce this condition on the boundary locations in the local reconstruction step.

It is straight-forward in our algorithm to track the end-points of an open curve. We simply explicitly store and move these particles like usual Lagrangian type methods. There is no reconstruction or resampling associated to these end-point locations. For 3 dimensional problems, we explicitly represent the boundary of an open surface by applying the algorithm we described in the previous section. This gives a quasi-uniform sampling of the closed curved.

A more important step is how our algorithm incorporates this boundary information. Since in the evolution step the motion of each particle is obtained by solving an ODE independent of each other, this condition on the boundary location will have no effect on the motion step. The local reconstruction phase of the algorithm is also exactly the same as before. It does not matter whether an end-point or a boundary point is contained in the particles we have collected for local reconstructing the interface, we repeat the same procedure as before to find a local polynomial representation of the interface.

It is the resampling phase we need to enforce the condition on the boundary locations. In 2 dimensions, assume that we have already obtained a set of foot-points (for a more general situation we have a set of particles on the interface) and also a quadratic fitting from the local reconstruction step. In the local coordinates system $\{(\mathbf{n}')^{\perp}, \mathbf{n}'\}$, we denoted these particles by (x_i, y_i) for $i = 1, \cdots, m$ and the local reconstruction by $f(x)$. To update the new foot-

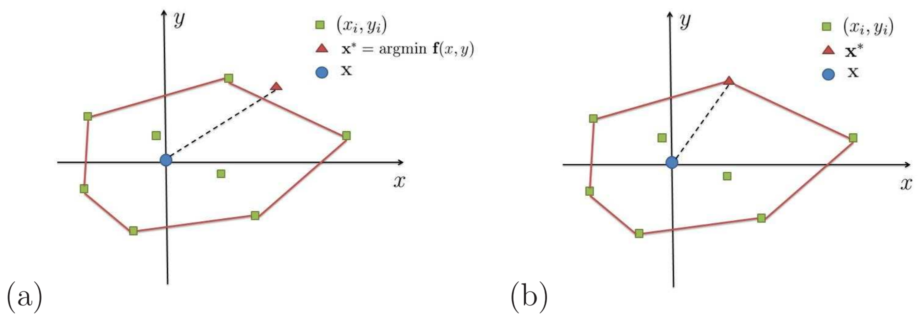

Fig. 4. (a) In the original algorithm, this potential new foot-point (red triangle) will be rejected since $\tilde{\mathbf{x}}^* \notin \tilde{\Omega}$. (b) If one of $\mathbf{y}$ corresponds to an boundary point, however, we associate the active grid point (blue circle) to the particle corresponding to $x_{\min}$ (red triangle).

point, we find the closest point from the grid point $\mathbf{p}$ to the function $f(x)$. In the previous algorithm, we have deactivated any grid point if this new foot-point comes from extrapolation, i.e. $\tilde{x}^* \notin [\min(x_i), \max(x_i)]$, figure 3 (a). In this new situation, on the other hand, extrapolation might appear when we reconstruct near the end-point of the open curve. For instance, in figure 3 (a), assuming that the associated particle corresponding to $x_{\min}$ is an end-point, the potential foot-point (red triangle) is indeed obtained by extrapolation. In this case, instead of deactivating this active grid point as before, we will associate this grid point to this particular end-point location, i.e. $\tilde{x}^* = x_{\min}$, as shown in figure 3 (b). The slightly change in the algorithm will allow accurate tracking of the end-point of the open interface.

In this resampling phase, all other steps will be similar as before. To update the Lagrangian information associated to this particle. For example, the normal vector at this new foot-point is taken to be the normal vector of the local reconstruction at $x = \tilde{x}^*$. The curvature and the global parametrization are both updated accordingly using the local interface reconstruction.

For 3 dimensional cases, we follow the same procedure. We first determine if the potential foot-point $\mathbf{x}^*$ is obtained by extrapolation. To check this, we compute the angle $(x_i, y_i) - \mathbf{x}^*$ made with the positive x-axis, denoted by θ_i. If any two adjacent θ_i is great than π, it indicates that the point $\mathbf{x}^*$ lies outside the convex hull $\tilde{\Omega}$ for the set (x_i, y_i), figure 4 (a) and Appendix A. Then we conclude that $\mathbf{x}^*$ is obtained by extrapolation. The computational complexity of this checking algorithm is only $O(m \log m)$, due to the sorting of θ_i.

If this potential foot-point is in $\tilde{\Omega}$, we follow the same algorithm as before. Otherwise, we check if any of (x_i, y_i) corresponds to a boundary point. If so, instead of deactivating the grid point, we will associate it to the closest point on the boundary of the open surface, figure 4 (b). Otherwise, we will remove this particular active grid point from the computational tube and will also delete the corresponding associating particle.

5 Example

Unless otherwise specified, we will be using the computational tube with radius $\gamma = 1.1h$, where h is the local grid size. We use quadratic polynomials for the least square fitting in local reconstruction, which uses 4 particles in two dimensions and 8 particles three dimensions. The time integration is done using the TVD-RK2 scheme with time step equals $0.75h_{\min}$, where $h_{\min}$ is the smallest grid size of the underlying mesh.

Since our sampling particles are unconnected on the interface, we are simply plotting the solution (interface location) using unconnected dots which shows the foot-points locations. For some examples in 3D, to better visualize the solution, we convert our computed solution to an implicit representation, i.e. a level set representation using

$$\phi(\mathbf{x}) = \mathbf{n} \cdot (\mathbf{y} - \mathbf{x}) \,, \tag{2}$$

and then plot the zero level set $\{\phi = 0\}$ using the MATLAB function isosurface.

5.1 *Open Curve*

5.1.1 *Rigid Body Rotation*

In the first example, we consider an upper hemisphere of a circle of radius 0.15 moving under the rigid body rotation given by

$$\begin{aligned} u &= 1 - 2y \\ v &= 2x - 1 \,. \end{aligned} \tag{3}$$

The curve will rotate around the point (0.5,0.5) in a period of π. Solutions at $t = m\pi/6$ for $m = 1, \cdots, 6$ are shown on the top row in figure 5. On the second row, we consider the solution at the final time $t = \pi$. The end-points of the open curve are tracked explicitly using the TVD-RK2 scheme and are plotted using a red circle. As we can see from this solution, end-points locations from our algorithm matches with the exact locations very well.

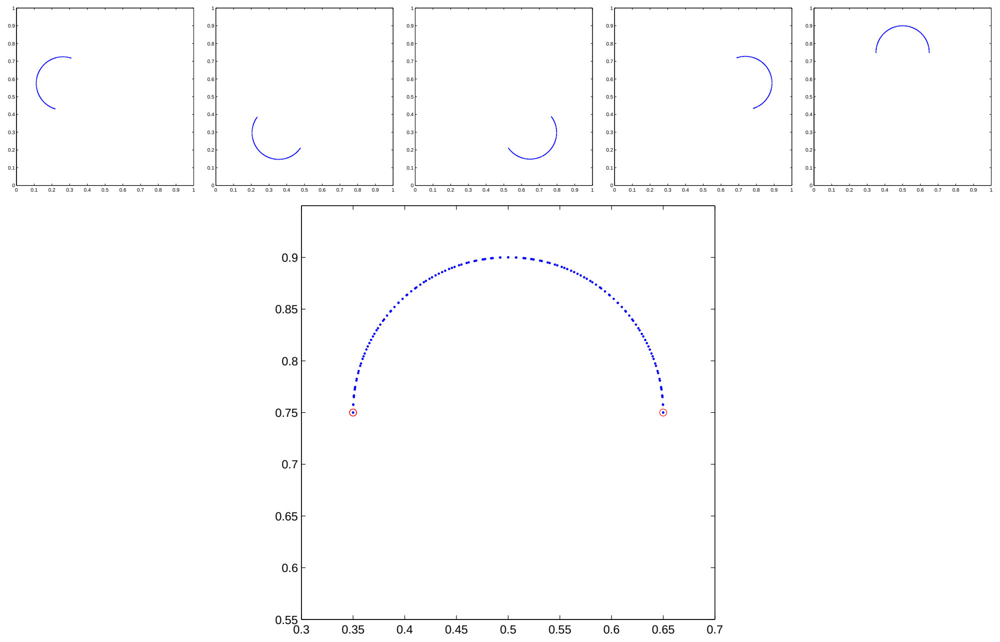

Fig. 5. Rotation of an upper half circle using an underlining uniform mesh of resolution 129^2. The second row shows the solution at the final time $t = \pi$ with the exact endpoint locations plotted in red circle.

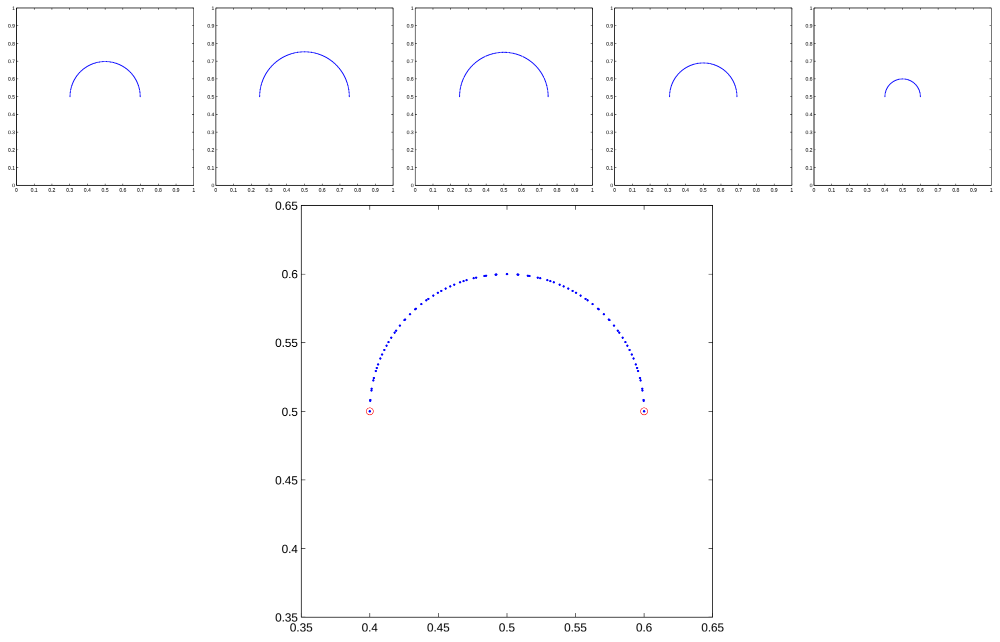

Fig. 6. Normal motion of an upper half circle using an underlining uniform mesh of resolution 129^2. The half circle first expands in the outward normal direction for $t = 0.25$ and then collapses in the inward normal direction for $t = 0.25$. The second row shows the solution at the final time $t = 0.5$ with the exact endpoint locations plotted in red circle.

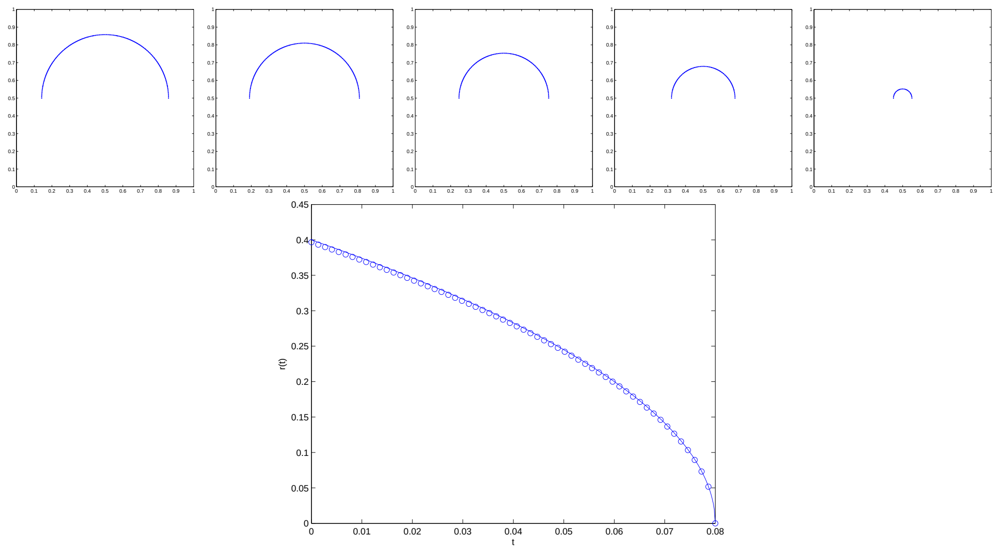

Fig. 7. Normal motion of an upper half circle using an underlining uniform mesh of resolution 129^2. The second row shows the change in the mean distance from the center $(0.5, 0.5)$. The solid line is the exact distance, the blue circle is the computed solution.

5.1.2 Motion in the Normal Direction

A slightly more complicated example is the motion in the normal direction. We consider the inward normal motion of an upper hemisphere of a circle with radius 0.35. Figure 6 shows the solutions at $t = 0.5m$ for $m = 1, \cdots, 5$. The solution at the final time are plotted on the second row. Like the previous example, we also show using red circles the location of the end-points computed by explicitly tracking them using an ODE solver.

5.1.3 Motion by Mean Curvature

We next consider the motion by mean curvature of an upper hemisphere of a circle of radius 0.4 centered at $(x_c, y_c) = (0.5, 0.5)$. We solve the evolutions of these two interfaces up to $t = 0.08$ using the time step restriction $\Delta t = 0.5\Delta x^2$. Away from the end-points of the open curve, the radius of the circle can be analytically calculated and it is given by $r(t) = \sqrt{0.4^2 - 2t}$. At the end-points, on the other hand, the curvature is not well-defined. Instead, we explicitly impose the motion at the two-ends by

$$\mathbf{v}(\mathbf{y}) = -\mathbf{n}(\mathbf{y})/|x - x_c| \,. \qquad (4)$$

The solutions at various times are shown in figures 7 and 8. Figure 7 uses an underlying mesh of 129 grids in each direction. We have doubled the number of grids in each direction in figure 8. On the second row of each figures, we plot the change in both the total number of particles and also the mean radius

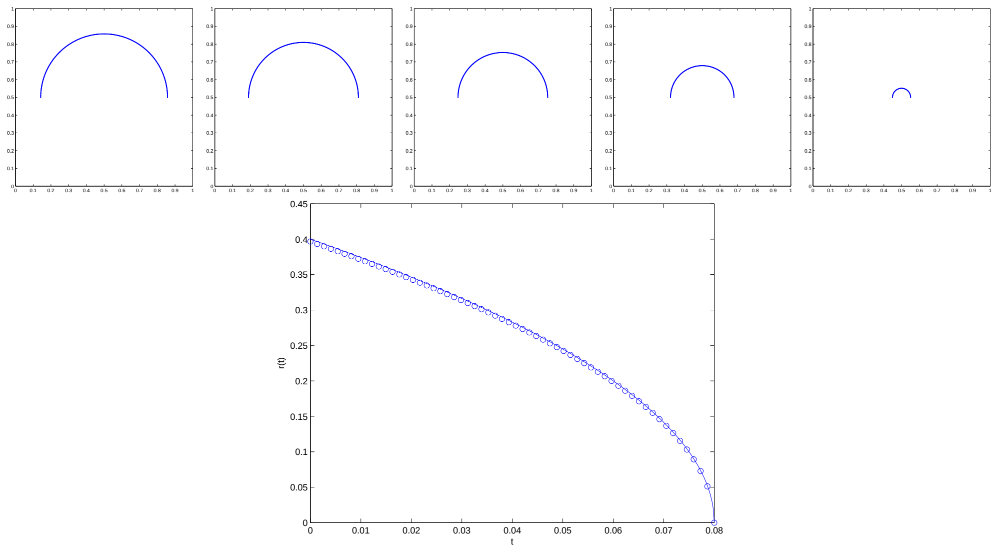

Fig. 8. Normal motion of an upper half circle using an underlining uniform mesh of resolution 257^2. The second row shows the change in the mean distance from the center $(0.5, 0.5)$. The solid line is the exact distance, the blue circle is the computed solution.

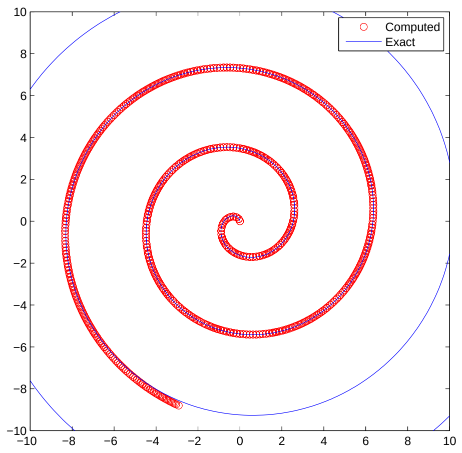

Fig. 9. Motion of a step-line at $t = 25$ with $\lambda = 0.2$ using an underlining uniform mesh of resolution 129^2. The solution by [5] is plotted in solid line, while our computed solution is shown by red circles.

of the circle. The computed mean radius (the circles) matches with the exact solution (solid line) very well.

5.1.4 Spiral Crystal Growth

As an interesting application of the proposed algorithm, we follow [20] and compute the motion of a slip-line in the spiral crystal growth. Proposed in [5], the slip-line is initially a straight segment and its motion is in the normal direction with the normal velocity related to the curvature κ given by

$$\mathbf{v} = (1 - \lambda\kappa)\mathbf{n}\,. \tag{5}$$

The end-points of the line segment are fixed for all time.

In figure 9, we consider the following initial set-up so that the exact solution can be analytically found. The initial slip-line is a straight line with a single screw dislocation, i.e. only one of the end-point is fixed while the other end-point is allowed to freely move. The initial slip-line is the straight line joining the origin and the point (10,0), with the origin is fixed. The critical radius λ equals 0.2. Figure 9 shows the computed solution (red circles)at $t = 25$ together with the analytical solution (blue solid line) [5, 20]

$$\theta = \frac{\sqrt{3}}{2(1+\sqrt{3})}\left[\frac{r}{\lambda} + \log\left(1 + \frac{r}{\lambda\sqrt{3}}\right)\right] + \frac{\pi}{2}\,. \tag{6}$$

The computed solution matches with the exact solution very well, except near the free-end where the curvature is not accurately approximated.

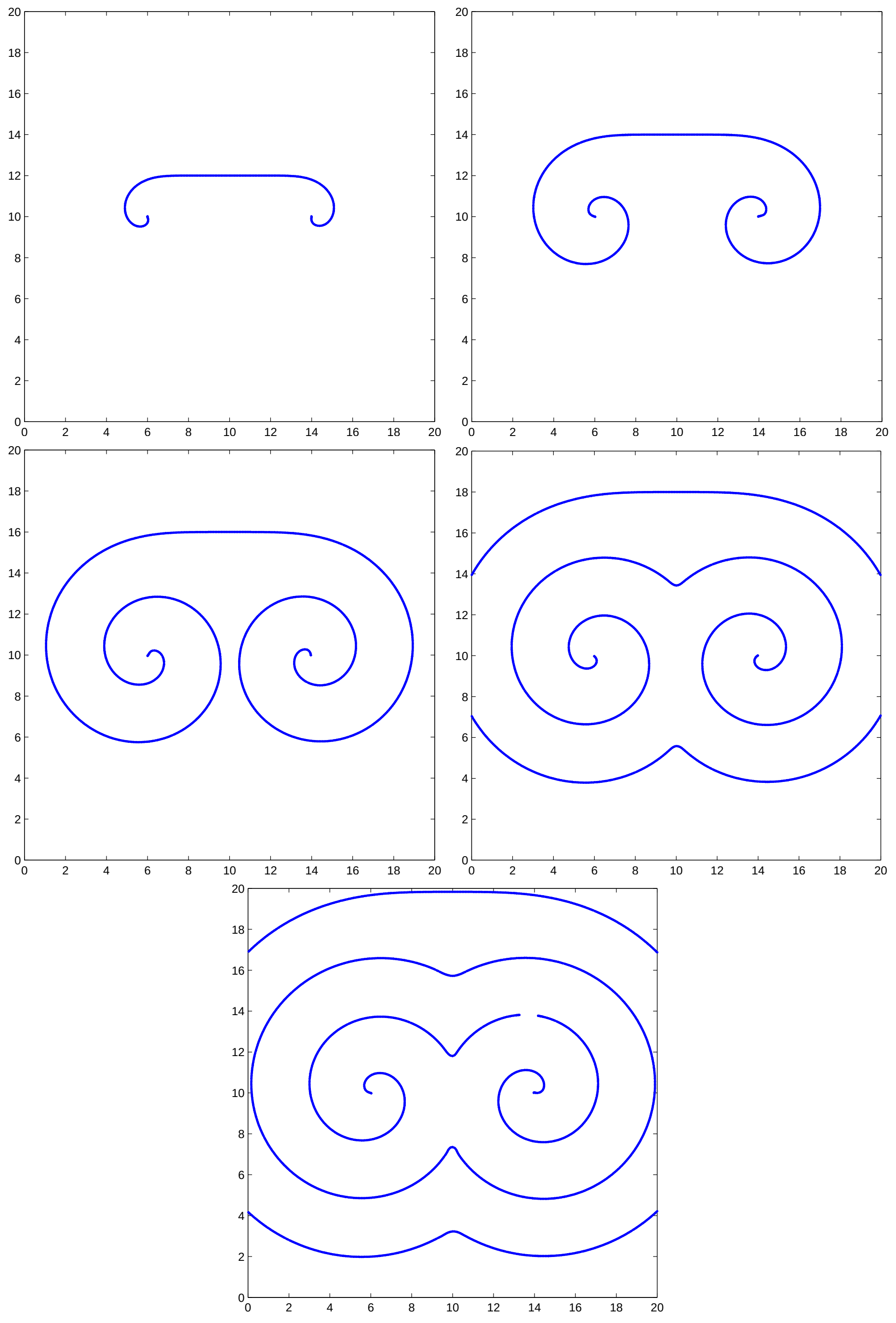

Fig. 10. Motion of a step-line with $\lambda = 0.1$ using an underlining uniform mesh of resolution 257^2.

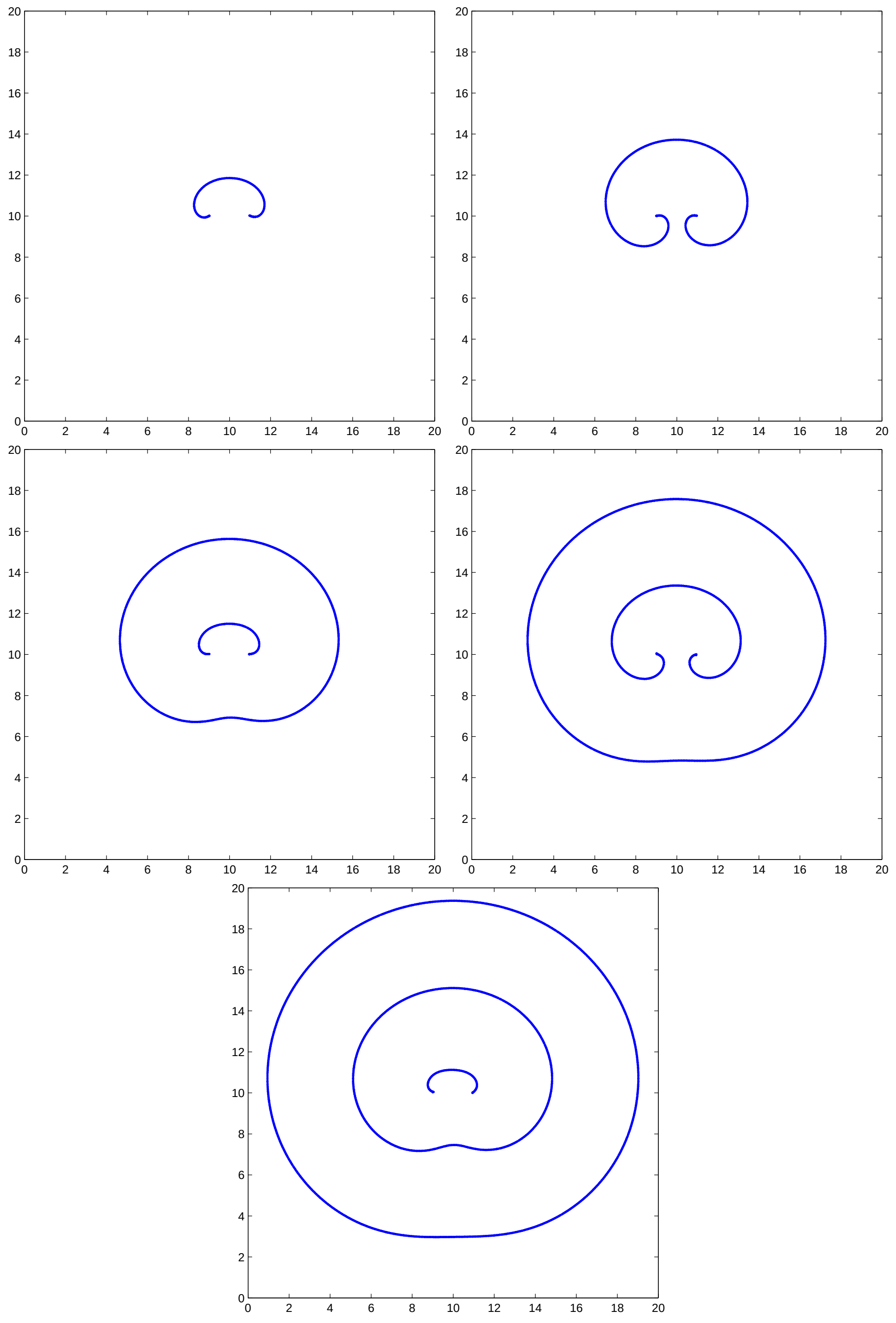
Fig. 11. Motion of a step-line with $\lambda = 0.1$ using an underlining uniform mesh of resolution 257^2.

Two other examples are also taken from [20]. The first example uses an initial line segment of length 8 with $\lambda = 0.1$. The solutions at various time are plotted in figure 10. Another set-up uses a line segment of length 2 with $\lambda = 0.2$, figure 11. Topological change in these solutions are nicely captured using only 257 grids in each direction.

5.1.5 Active Contour for Image Segmentation

Segmentation is one of many fundamental tasks in image processing. Since [13], extensive research have been done on applying variational approach to detect boundaries of objects in an image. In [13], the author proposed the so-called *snake* model, or the active contour model, in where the boundary of an object is detected by a piecewise C^1 curve. Starting from an initial guess in the class

$$\mathcal{C} = \{c : [a, b] \to \Omega, c \text{ piecewise } C^1\}, \tag{7}$$

the *snake* evolves to minimize the functional

$$\tilde{E}(c) = \int_a^b |c'(q)|^2 dq + \beta \int_a^b |c''(q)|^2 dq + \lambda \int_a^b g^2[|G * \nabla u_0(c(q))|] dq \,, \tag{8}$$

with the observed image $u_0 : \Omega = [0, 256]^2 \to [0, 1]$ for some edge detection function

$$g(\xi) = \frac{1}{1 + |\xi|^2} \,. \tag{9}$$

Numerically, one usually discretizes the curve by particles, which corresponds to the Lagrangian description. The implementation is fast since one solves only ODE for these marker particles. Unfortunately, this energy is not intrinsic in the sense that the minimizer depends on the parametrization. Moreover, this formulation does not allow topological change of the boundary and, in fact, this approach can detect only one single object.

One os many very important improvements is the geodesic active contour model [6] in which one minimizes the energy

$$E(c) = \int_a^b g(|G * \nabla u_0(c(q))|)|c'(q)| dq \,. \tag{10}$$

The resulting evolution equation is given by

$$\frac{\partial c}{\partial t} = [\kappa g - \mathbf{n} \cdot \nabla g]\, \mathbf{n}\,. \tag{11}$$

To handle topological change, the authors model the evolution using the level set method. However, this so-called geodesic active contour method mainly compute closed boundaries and it does not incorporate with any constrain on the curve such as the curve has to pass through certain location. For more details about the formulation, we refer interested readers to [19, 17, 1].

In this paper, we follow [6] and evolve the contour according to

$$v_n = \kappa g - \mathbf{n} \cdot \nabla g\,. \tag{12}$$

Unlike the usual geodesic active contour, we do not implicitly define the curve using the level set method. Moreover, we are considering a closed curve with both end-point fixed. Various active contour models could deal with open curves [10, 9], but all open curves were discretized in the Lagrangian formulation. It is therefore not easy to handle topological change of the solution.

In figure 12, we consider an image consists of line segments. The initial curve (red circles) is the lower hemisphere of a circle. As this open curve evolves, it will eventually stop once it collides with the boundary of an object (the black curve in this case).

When we flip the image upside down, as shown in figure 13, the curve is trapped in a local minimizer in which our open curve joins different parts in the image by a straight line segment. This is a typical situation in most active contour models. Since the minimization problem is not convex, the energy has many local minimizers and the solution might very easily get stuck in one of them. Global minimizer of active contour models can be found using different approaches [14, 3], but these approaches are not being studied in the current paper.

Now we slightly modify the image by removing two segments, as shown in figure 14. By doing this, the two fixed end-points do not coincide with the boundary of the object anymore. On the boundary of the object, the edge function g closes to zero and so that particular part of the curve does not contribute to the energy. Away from the boundary of the object, the energy will mimic the curve to minimize $|c'|$. Therefore, the interpretation of this segmentation is to find a path joining the two end-points so that it coincides with the boundary as much as possible. This result cannot be obtained directly using typical geodesic active contours since there is no mechanism to impose the end-point conditions. Again, since the minimization problem is not convex,

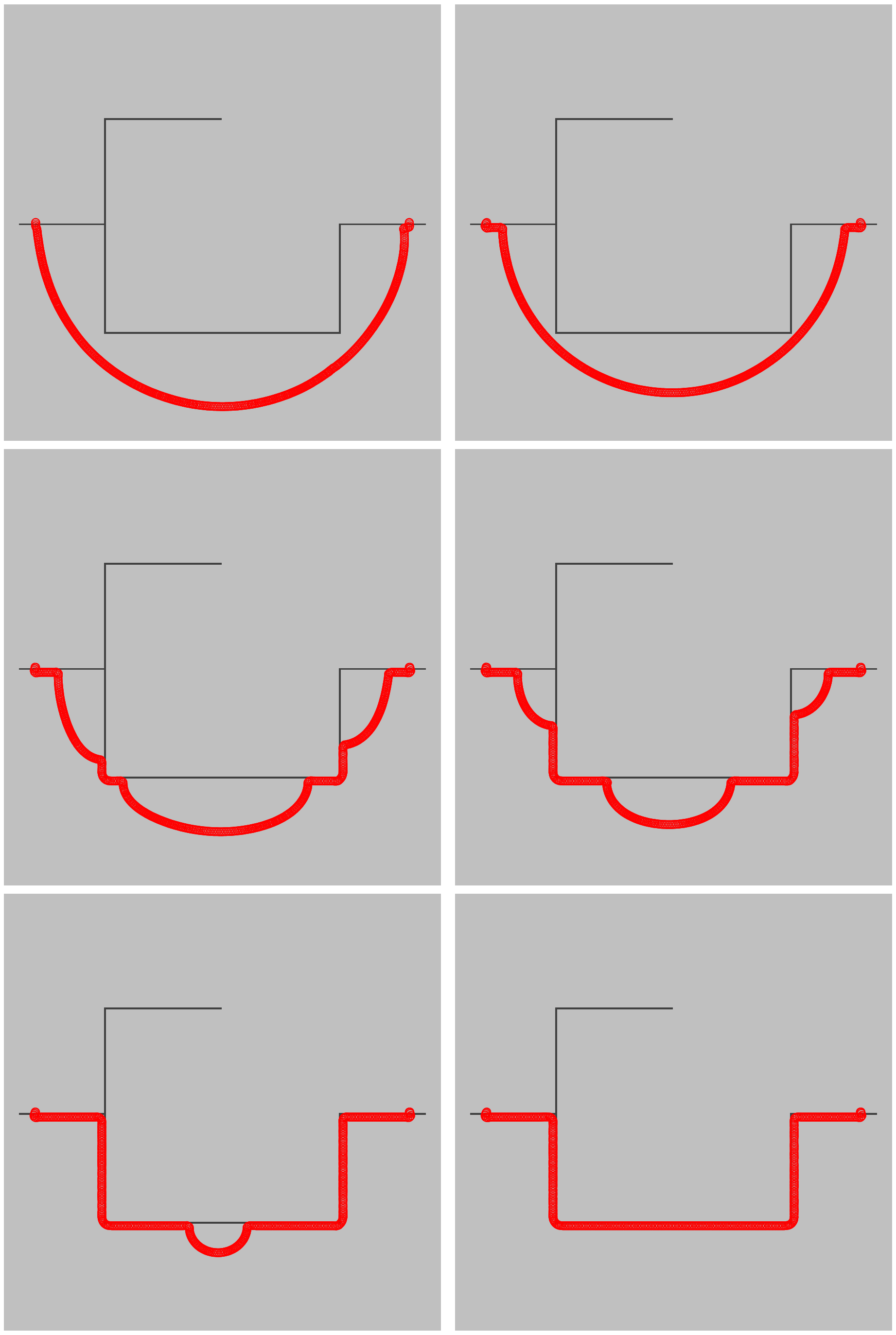

Fig. 12. Evolution of the active contour (red curve) for detecting edges. Two end–points are fixed in the motion.

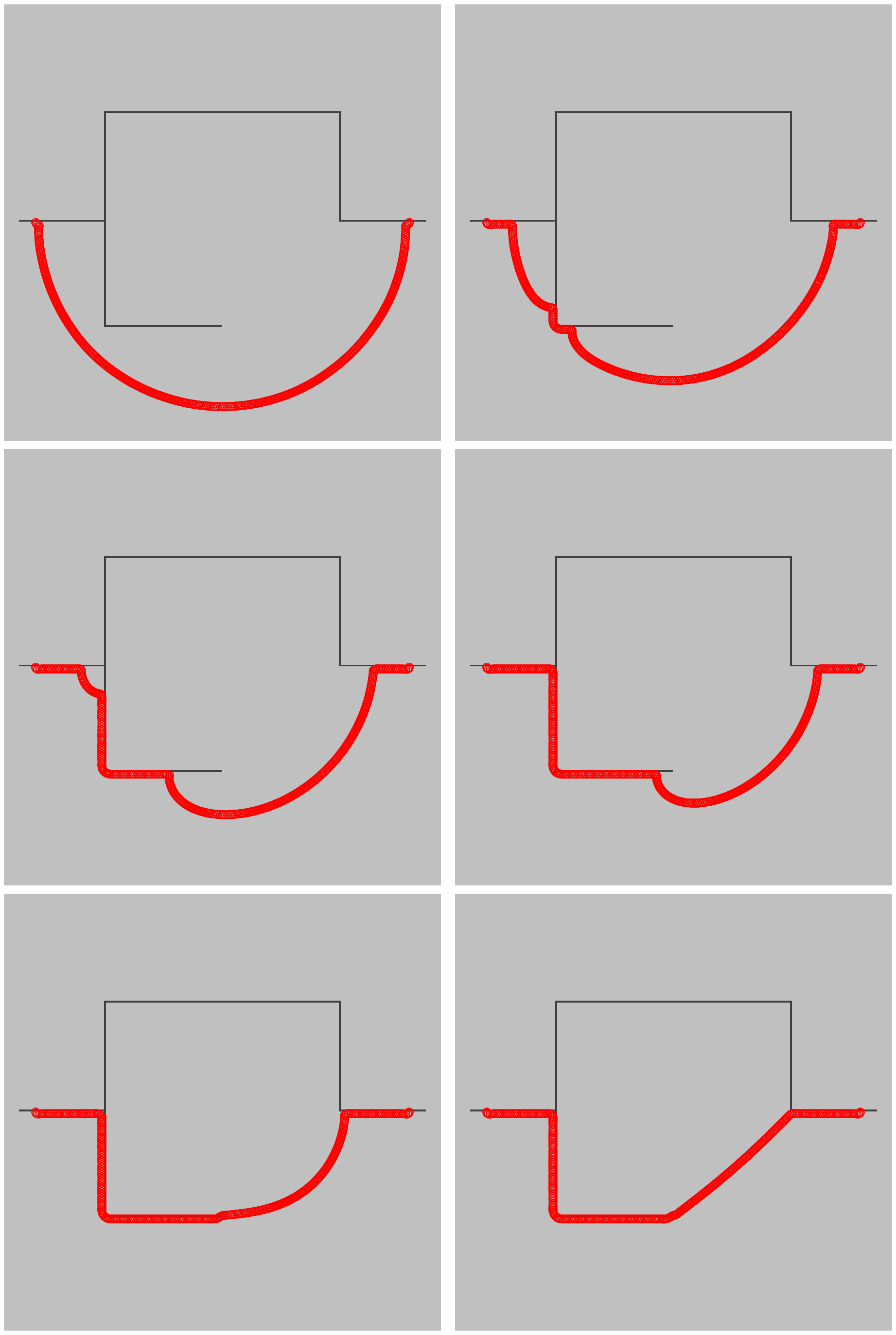
Fig. 13. Evolution of the active contour (red curve) for detecting edges. Two end–points are fixed in the motion.

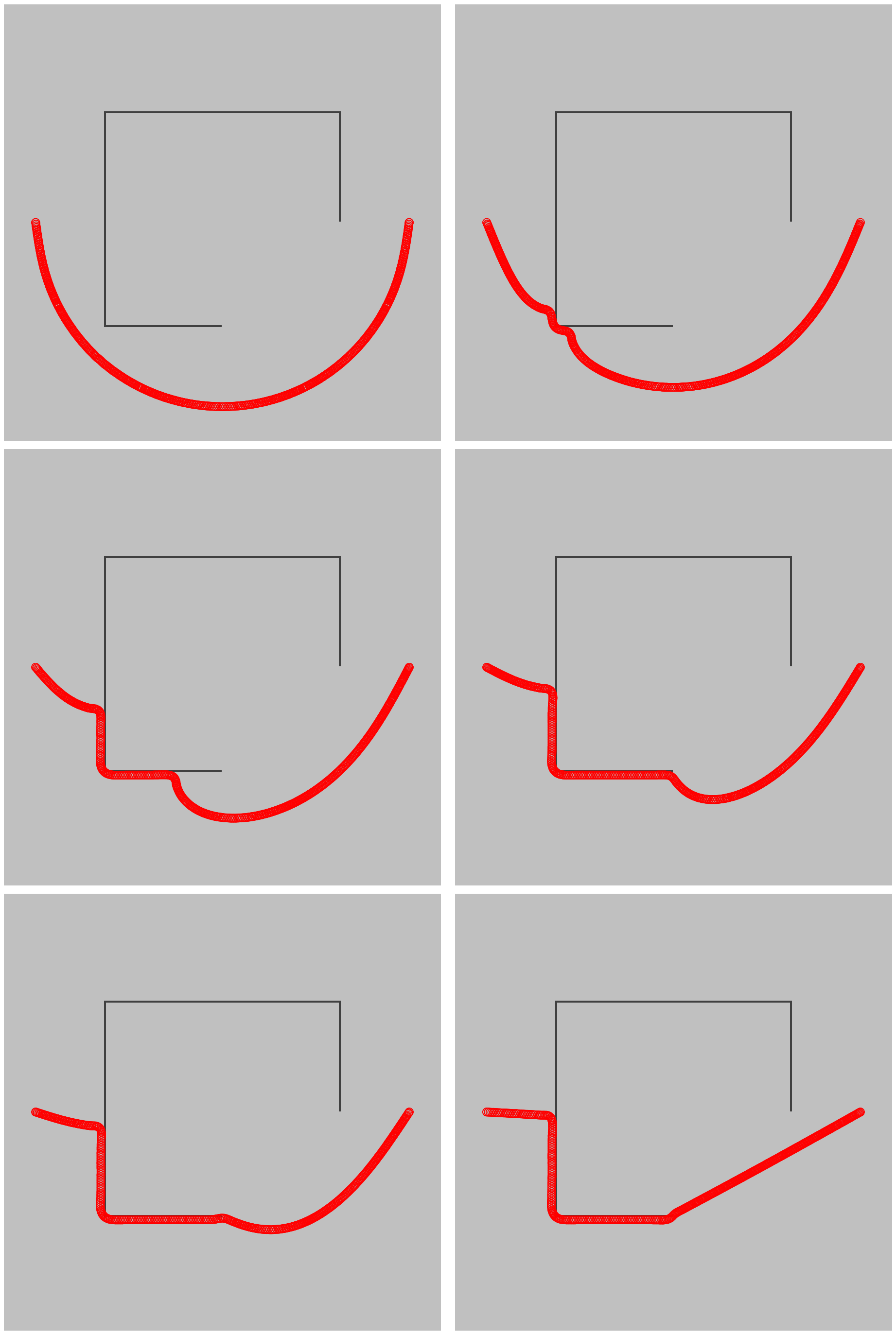

Fig. 14. Evolution of the active contour (red curve) for detecting edges. Two end–points are fixed in the motion.

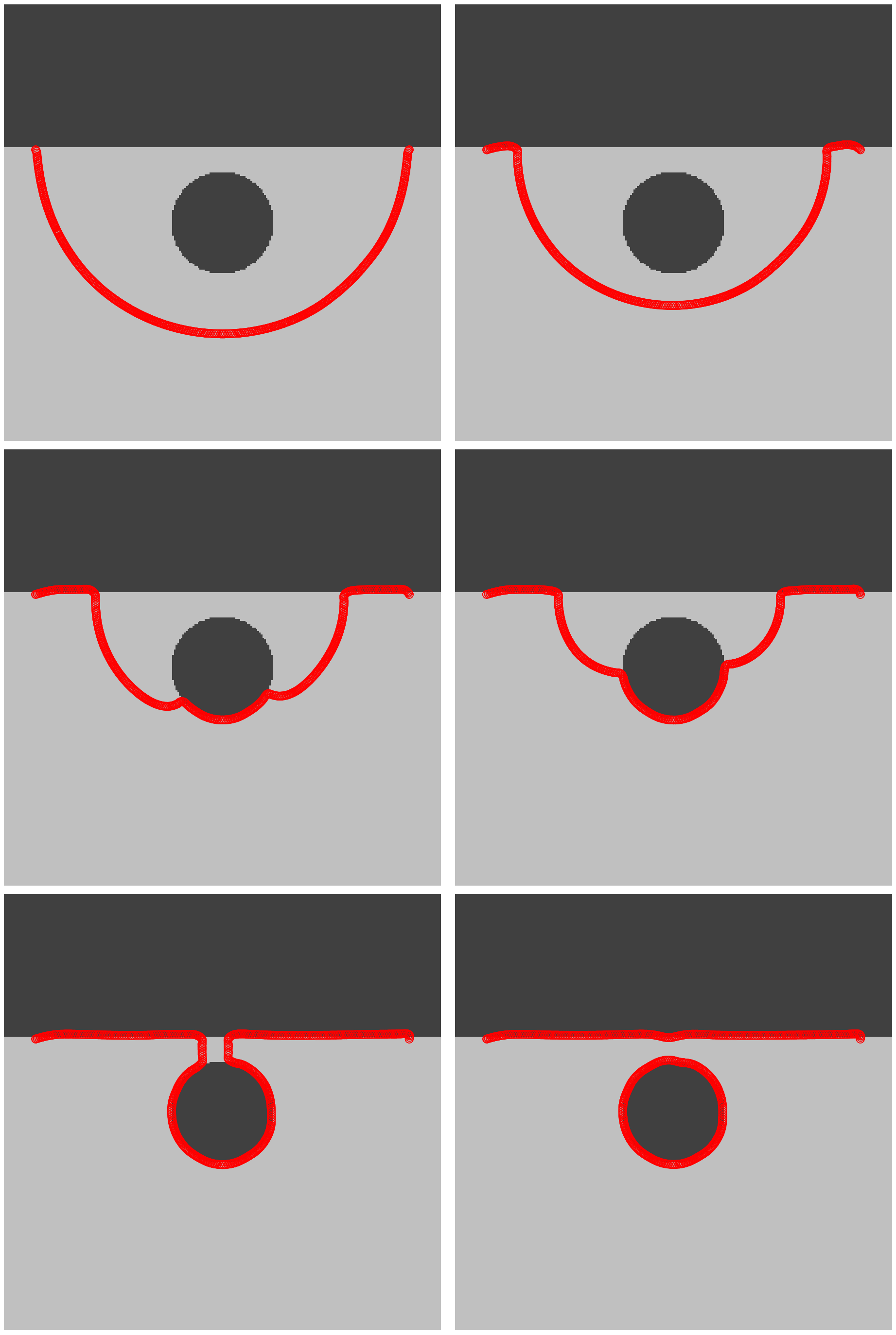

Fig. 15. Evolution of the active contour (red curve) for detecting edges. Two end–points are fixed in the motion. Topological change can be naturally incorporated in the model.

the curve is trapped in a local minimizer and the segmentation result depends highly on the initial guess. Indeed, the global minimizer for this example should be the path going from the upper route taking only piecewise horizonal and vertical segments.

Next we consider an image for which we have a topological change in the evolution of the active contour. Figure 15 uses an image consist of a top black region, a disconnected black circle and a light background. Two end-points of the initial curve (red circles) are chosen such that they touch the boundary of the rectangular dark region. As the curve shrinks, it wraps up the disjoint dark circle and the curve splits into two disconnected pieces. One surrounds the circle and the other one connects the two fixed end-points detecting the boundary of the dark region on the top.

5.2 Open Surface

The motion of an open surface is more interesting. In the first example, we will consider the simple rotation

$$\begin{aligned} u &= 1 - 2y \\ v &= 2x - 1 \\ w &= 0\,, \end{aligned} \tag{13}$$

of an upper hemisphere of a sphere of radius 0.15 initially centered at (0.5,0.75,0.5) for $t = \pi$. The boundary of the upper hemisphere is a circle centered at $(0.5, 0.75)$ with radius 0.15 on the plane $z = 0$. Its evolution is computed by the approach proposed in Section 3 and the solutions at various times are shown in figure 16. The whole rotation of the upper hemisphere is plotted in figure 17, where the solutions in 16 are drawn using red circle. As we can see, the extra condition on the boundary location are satisfied very well. There is no sampling particle lying outside the open surface.

A much more complicated motion is the single vortex flow proposed in [8] where the velocity field is given by

$$\begin{aligned} u_1(x,y,z) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z) \\ u_2(x,y,z) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ u_3(x,y,z) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\,. \end{aligned} \tag{14}$$

The original set-up of this test is to model the evolution of a closed sphere centered at $(0.35, 0.35, 0.35)$ with radius 0.15. In the current paper, we will consider the motion of only the upper hemisphere. Like [8, 12, 16], we consider the flow with reverse motion by multiplying the above velocity field by a factor of $\cos(\pi t/T)$. At the final time $t = T$, the solution should go back to the original configuration. In figure 18, we plotted the evolutions of the boundary of this upper hemisphere using 151 grids in each direction. At $t = 0$, this closed curve is given analytically by $(x, y, z) = (r\cos\theta, r\sin\theta, 0)$. In figure 19,

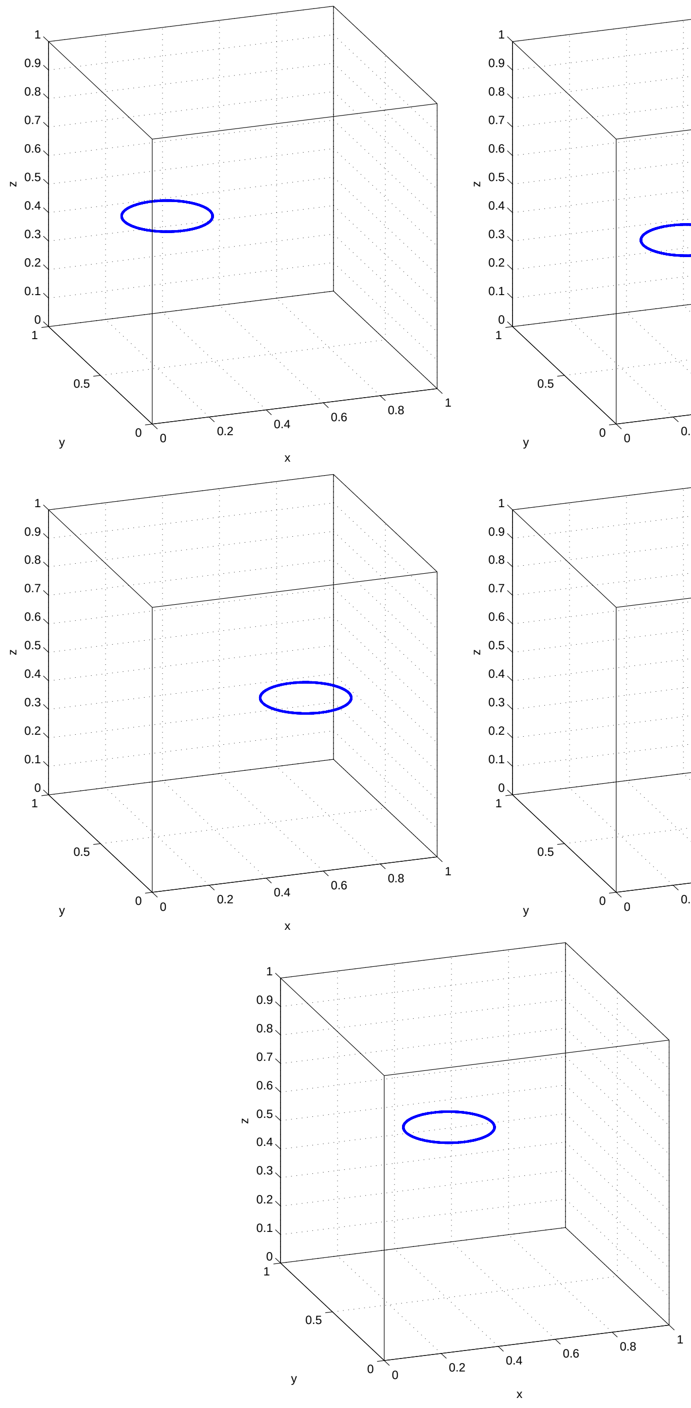

Fig. 16. Motion of the boundary of an upper hemisphere under the rigid body rotation and an underlining uniform mesh of resolution 151^3.

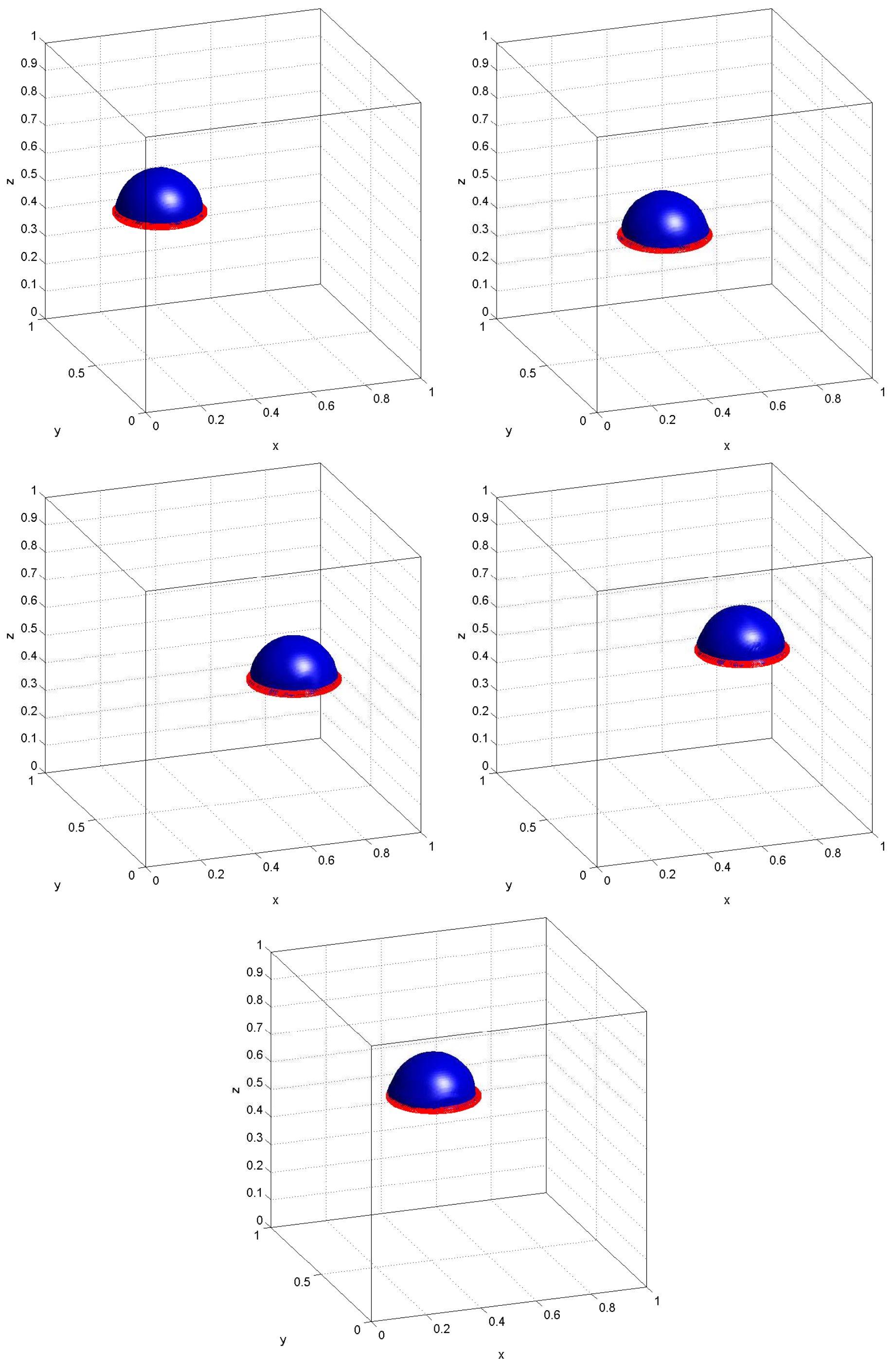

Fig. 17. Motion of an upper hemisphere under the rigid body rotation and an underlining uniform mesh of resolution 151^3. The boundary of the surface is shown using red circle.

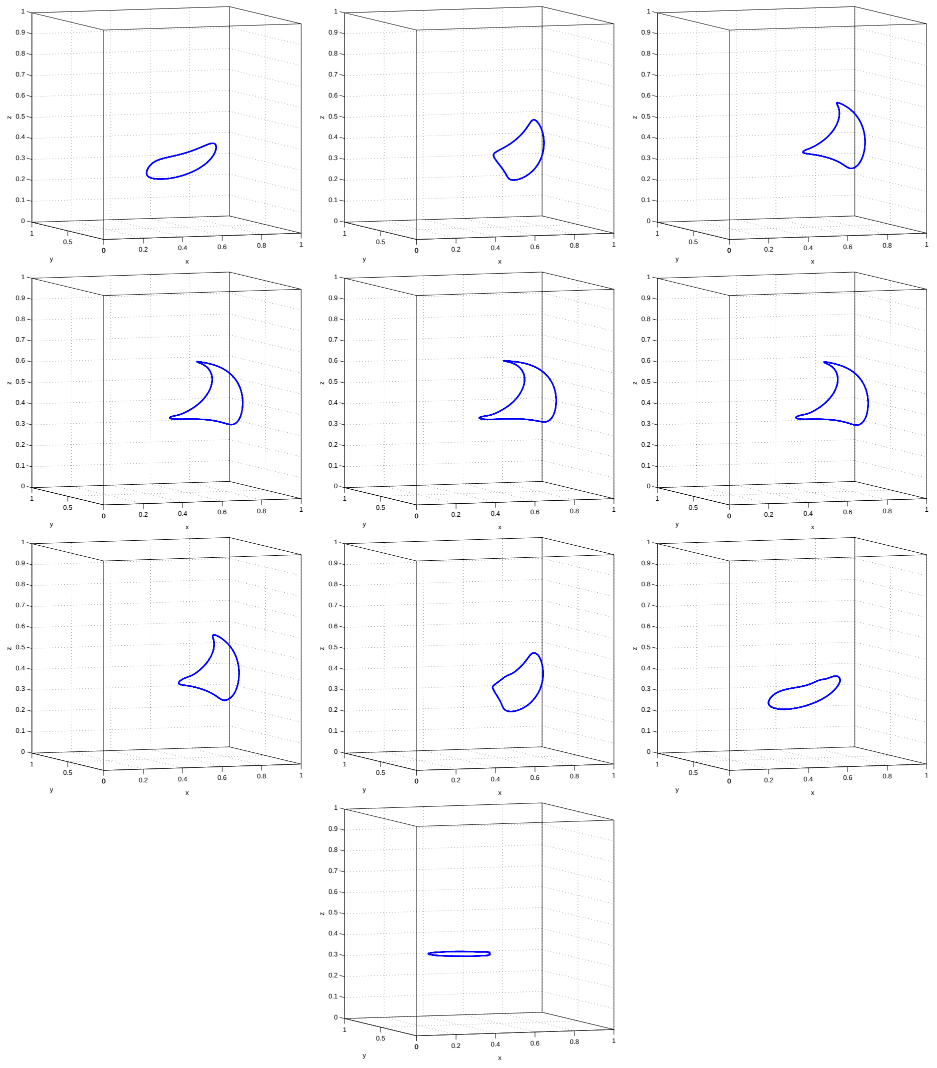

Fig. 18. Motion of the boundary of an upper hemisphere under the vortex motion with rewind motion using $T = 1.5$ and an underlining uniform mesh of resolution 151^3.

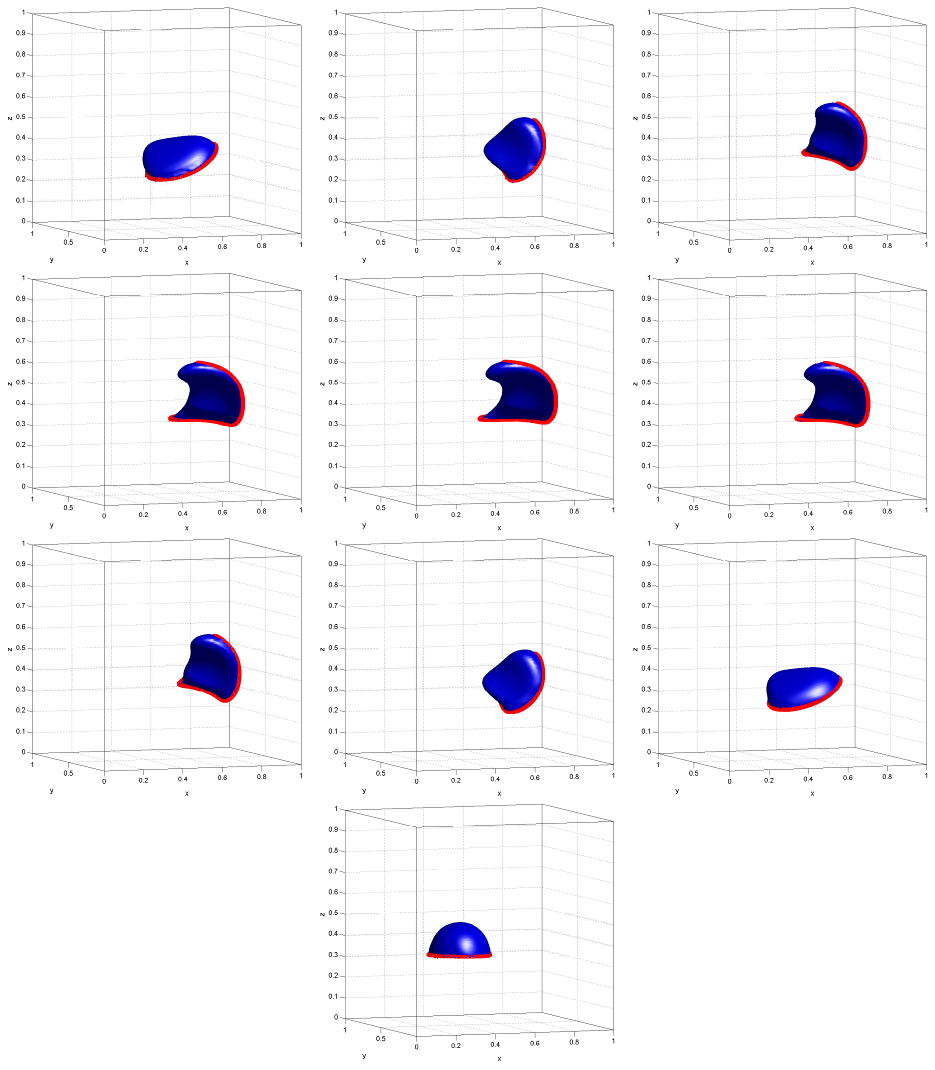
Fig. 19. Motion of an upper hemisphere under the vortex motion with rewind motion using $T = 1.5$ and an underlining uniform mesh of resolution 151^3. The boundary of the surface is shown using red circle.

we show the motion of the whole upper hemisphere with the boundary points from 18 shown by red circle.

Appendix A

In this appendix, we will show that we can determine if $\mathbf{x}^*$ lies inside the convex hull made by the set (x_i, y_i) by considering only the angle θ_i made by $(x_i, y_i) - \mathbf{x}^*$ and the positive x-axis.

Without lost of generality, we assume $\mathbf{x}^* \neq (x_i, y_i)$ is the original and θ_i's have already been sorted in the ascending order. If there exists I such that $|\theta_I - \theta_{I-1}| > \pi$, we rotate the (x, y)-coordinates to make $\tilde{\theta}_i \in (0, \pi) \forall i$. This gives $\tilde{y}_i > 0$ and therefore $0 \notin \tilde{\Omega} = \{(x, y) = \sum_i \alpha_i (x_i, y_i), \alpha_i > 0, \sum_i \alpha_i = 1\}$.

References

[1] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing - Partial Differential Equations and the Calculus of Variations.* Springer, 2006.

[2] S. Basu, D.P. Mukherjee, and S.T. Acton. Implicit evolution of open ended curves. *IEEE International Conference on Image Processing*, 1:261–264, 2007.

[3] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher. Fast global minimization of the active Contour/Snake model. *Journal of Mathematical Imaging and Vision*, 28:151–167, 2007.

[4] P. Burchard, L.-T. Cheng, B. Merriman, and S. Osher. Motion of curves in three spatial dimensions using a level set approach. *J. Comput. Phys.*, 170(2):720–741, 2001.

[5] W.K. Burton, N. Cabrera, and F.C. Frank. The growth of crystals and the equilibrium structure of their surface. *Phi. Trans. Roy. Soc. Lond.*, 243A:299–358, 1951.

[6] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[7] L.-T. Cheng, P. Burchard, B. Merriman, and S. Osher. Motion of curves constrained on surfaces using a level-set approach. *J. Comput. Phys.*, 175:604–644, 2002.

[8] 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.

[9] P. Fua and Y.G. Leclerc. Model driven edge detection. *Machine Vision and Applications*, 3:45–56, 1990.

[10] J. Gilles and B. Collin. Fast probabilistic snake algorithm. *ICIP*, 3:14–17, 2003.

[11] J. Gomes and O. Faugeras. Using the vector distance functions to evolve manifolds of arbitrary codimension. *Lecture Notes In Computer Science*, 2106:1–13, 2001.

[12] S. Hieber and P. Koumoutsakos. A lagrangian particle level set method. *J. Comput. Phys.*, 210:342–367, 2005.

[13] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[14] S. Leung and S. Osher. Fast global minimization of the active contour model with tv-inpainting and two-phase denoising. *Proceeding of the 3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 149–160, 2005.

[15] S. Leung and H.K. Zhao. A grid based particle method for moving interface problems. *Submitted to Journal of Computational Physics*, 2008.

[16] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.*, 225:300–321, 2007.

[17] S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

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

[19] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.

[20] P. Smereka. Spiral crystal growth. *Physica D*, 138:282–301, 2000.

[21] J.E. Solem and A. Heyden. Reconstructing open surfaces from image data. *International Journal of Computer Vision*, 69:267–275, 2006.

[22] G. Ventura, J.X. Xu, and T. Belytschoko. A vector level set method and new discontinuity approximations for crack growth by EFG. *International Journal for Numerical Methods in Engineering*, 54:923–944, 2002.