Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A grid based particle method for evolution of open curves and surfaces

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

[a] Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[b] Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875, USA

ARTICLE INFO

Article history:
Received 24 October 2008
Received in revised form 7 July 2009
Accepted 10 July 2009
Available online 28 July 2009

MSC:
65D17
76M25

Keywords:
Open curves
Open surfaces
Eulerian mesh
Lagrangian particles

ABSTRACT

We propose a new numerical method for modeling motion of open curves in two dimensions and open surfaces in three dimensions. Following the grid based particle method we proposed in [S. Leung, H.K. Zhao, A grid based particle method for moving interface problems. J. Comput. Phys. 228 (2009) 2993–3024], we represent the open curve or the open surface by meshless Lagrangian particles sampled according to an underlying fixed Eulerian mesh. The underlying grid is used to provide a quasi-uniform sampling and neighboring information for meshless particles. The key idea in the current paper is to represent and to track end-points of the open curve and boundary-points of the open surface explicitly and consistently with interior particles. We apply our algorithms to several applications including spiral crystal growth modeling and image segmentation using active contours.

1. Introduction

In a recent work [17], we proposed a novel grid based particle method to represent and track interface motion. In our approach the interface is represented by meshless, i.e., no triangulation or parametrization, Lagrangian particles which are associated to an underlying uniform or an adaptive Eulerian mesh. This results in a quasi-uniform sampling of the interface. The motion of the interface is tracked by these particles. The interface may be reconstructed locally and also be resampled during the evolution. The underlying mesh provides local neighboring information for the meshless particles which is used for local reconstruction by least square fitting in a local coordinate system. Adaptive local sampling of the interface can be easily achieved by local adaptivity in the underlying mesh. Moreover, the meshless representation allows one to control topological changes easily by using both Lagrangian and Eulerian information available. We have successfully applied this technique to different interface motions such as the motion by an external velocity field and various geometric motions to demonstrate the efficiency, the accuracy and also the flexibility of the approach.

The goal of this paper is to generalize the techniques in [17] to handle motions of an open curve in 2D or an open surface in 3D. The key point is to explicitly track the boundary (end-points of the open curve and a closed boundary curve of the open surface) and then incorporate this boundary condition consistently and accurately to confine the sampling and local reconstruction of interior region of open curves and surfaces.

In many applications, it is useful to model slender objects as open curves and to model thin sheets as open surfaces. For explicit Lagrangian tracking methods, open curves and surfaces do not pose extra challenges besides those usual difficulties for tracking methods in dealing with general moving interfaces, such as topological changes. For implicit Eulerian

* Corresponding author.
E-mail addresses: masyleung@ust.hk, syleung@gmail.com, syleung@math.uci.edu (S. Leung), zhao@math.uci.edu (H. Zhao).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.07.017

methods, there is no natural way to represent open curves and open surfaces since there is no distinction of interior and exterior regions. Recently, a few approaches have been proposed for dealing with open curves and surfaces based on the level set method [21]. One approach was the work of [23] for modeling spiral crystal growth. The author used the intersection of two level set functions to represent the codimension-two boundary of the open curve or surface. The curve or surface of interest was implicitly defined as the zero level set of one signed distance function at which another one was positive, i.e. $S = \{x : \phi(x) = 0 \text{ and } \psi(x) > 0\}$. However, one has to define velocities for not only the level set function ϕ which represented the curve or 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 [23] 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 a given motion law. A generalization to this approach was proposed in [24] for constructing open surfaces from point cloud data. The method incorporated the method proposed in [5,9] to allow motion of the boundary. Computationally, all these methods were not efficient since they not only solved one partial differential equation (PDE) for the level set function to get the implicit representation of the open curve or surface, but the number of PDEs, i.e. the number of the level set functions, equals to 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 exactly recovered and there is always an $O(\mu)$ smoothed zone near the interface.

Another method to modeling motion of open curves and surfaces is the vector distance function method [13] which extends 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 $\mathbf{x}$ to the interface. Therefore, this vector valued function embedded not only the distance and the inside or outside information, but also the normal vector. Unfortunately, this representation required solving the same number of PDE's as 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 curves or surfaces which are open or closed in $\mathbb{R}^3$.

Similar to our representation in [17], the vector level set method [25] 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 or reconstruction was considered elsewhere.

There are a few nice properties of our approach to open curves and surfaces. First, the end-points/boundary-points of the open curve/surface are accurately and explicitly tracked like usual tracking methods. This is important for many physical applications. Moreover, the grid based particle method can naturally handle both the viscosity solution and the multivalued solution using meshless Lagrangian particles with an underlying Eulerian mesh. This gives a flexibility to control the change of topology in the solution. Adaptivity can also be applied easily according to local dynamics and features by local mesh refinement. The algorithm can be implemented easily and efficiently since no PDE is involved on the underlying mesh.

This paper is organized as follows. In Section 2, we briefly review of the grid based particle method and introduce important notions for the rest of the paper. In Section 3, we generalize this approach to motion of an closed curve in three dimensions. With that, we explain how to apply the method to model motions of open curves and surfaces in Section 4. Various examples in both two dimensions and three dimensions are given in Section 5 to demonstrate the performance of our algorithm.

2. Grid based particle method

For the convenience of readers, we give a brief summary of the grid based particle method for motions of a closed interface in this section. For a complete and detailed description of the algorithm, we refer the readers to [17].

In [17], we represent the interface by meshless particles which are associated to an underlying Eulerian mesh. 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 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.

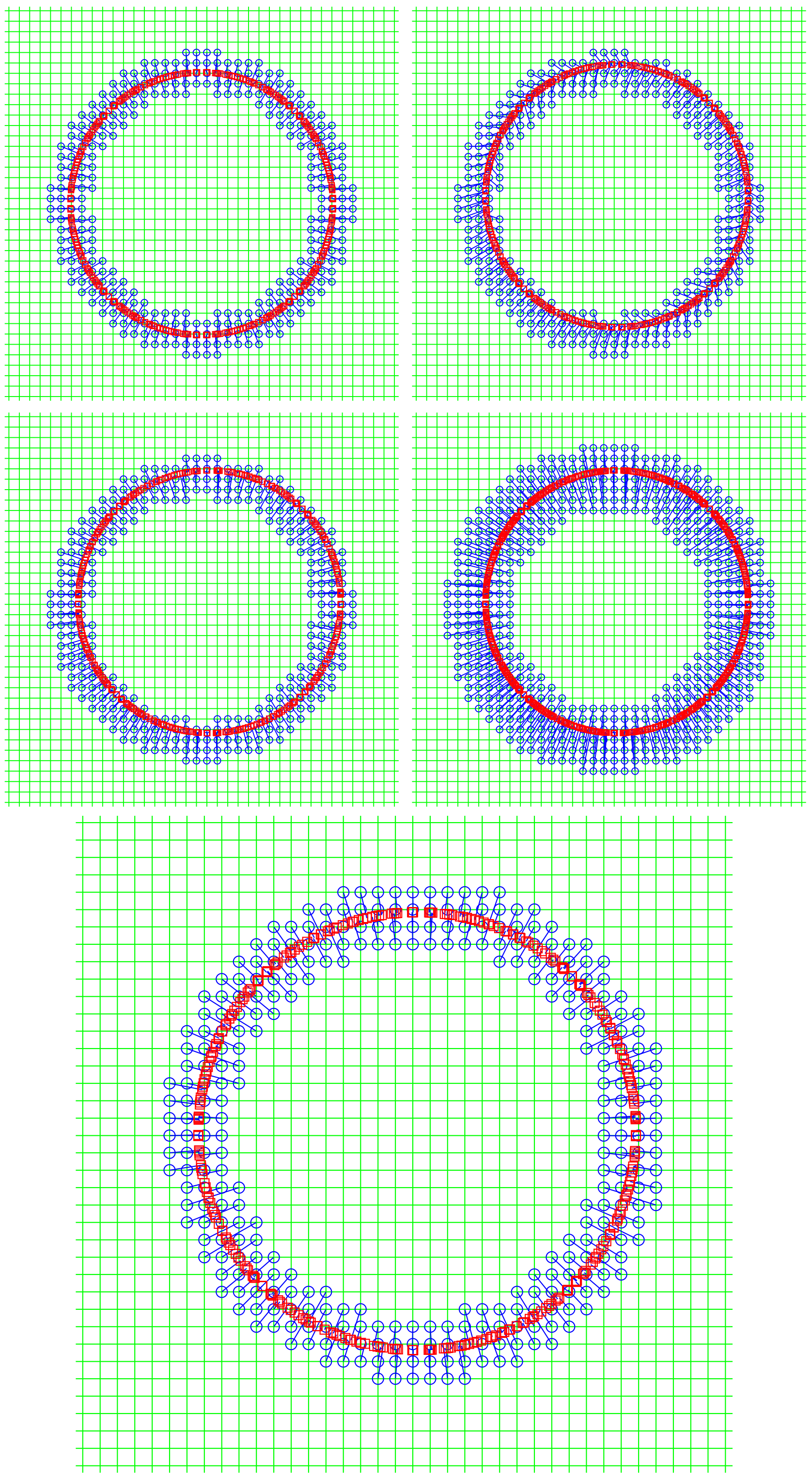


Fig. 1. Grid based particle method. From left to right: (a) initial representation/sampling, (b) after motion, (c) after re-sampling, (d) after activating new grid points with their foot-points and (e) after inactivating grid points with their foot-points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

This initial set-up is illustrated in Fig. 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 sampling of the interface. 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 a velocity field $\mathbf{u} = \mathbf{u}(\mathbf{y})$. One simply tracks the particles just like all other particle-based tracking methods, which is simple and computationally efficient. In particular one can solve a set of ordinary differential equations using high order schemes which gives 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 from its associated active grid point $\mathbf{p}$ anymore. For example, Fig. 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 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, Fig. 2(a). Using this local reconstruction, we find the closest point from this active grid point to the local approximation of the interface, Fig. 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. When different pieces of interface get close, e.g. before merging or crossing, one can classify neighboring particles from different pieces into different groups according to Lagrangian information, such as normal direction and/or some parametrization of the particles. This will allow us to reconstruct different pieces without mixing. Due to the availability of both Eulerian information, i.e., reference to the underlying mesh and Lagrangian information of particles, the meshless representation allows us to detect collision of different interfaces and to control topological changes easily. For more details, we refer interested readers to [17] for each of the above procedures.

To end this section, we summarize the algorithm and also give the computational complexity for modeling motions of a plane curve in two dimensions. In the following we let m be the points used in the local reconstruction of the interface, which can be treated as a small constant and n be the number of underlying grid points in each spacial dimension.

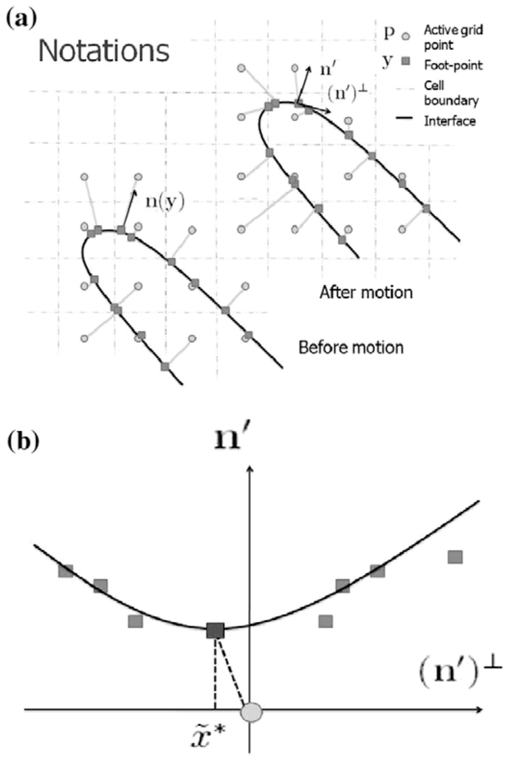


Fig. 2. (a) Definition of a local coordinates and (b) the way how we determine the new foot-point using a local least square reconstruction of the interface.

Algorithm (Grid based particle method).

(1) *Initialization* (Fig. 1(a)) ($O(n)$). 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* (Fig. 1(b)) ($O(n)$). Move all foot-points according to a given motion law.
(3) *Re-sampling* (Fig. 1(c)) ($O(mn \log m)$). 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* (Fig. 1(d) and (e)) ($O(mn \log m)$). 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) *Adaptivity*. Locally refine/coarse the underlying mesh if necessary.
(6) Iteration. Repeat steps 2–5 until the final computational time.

The resampling step in the above algorithm consists of the following three main processes applied to each foot-point: collecting foot-points in the neighborhood, reconstructing the interface locally and determining the new foot-point location. The first process has the computational complexity of $O(m \log m)$ since one has to first collect $O(m)$ points and then sort them according to the distance to the active grid point. To local reconstruct the interface, one has to solve the least square system which has a computational complexity of $O(m)$. For problems in three dimensional space, for example, one has to implement an iterative solver for finding the new foot-point using the local reconstruction. It is hard to estimate the complexity, but we can still limit the number of iterations to $O(m \log m)$. If the iteration does not converge in $O(m \log m)$ iterations, one can simply deactivate the corresponding grid point. Numerically, since we usually have a good initial guess, the iteration converges in only few iterations in practice. This gives the overall complexity for the resampling step $O(mn \log m)$ for $O(n)$ foot-points. Therefore, the overall computational complexity for moving the interface for one time step is $O(mn \log m)$.

3. Closed curves in 3D

As already discussed in [17] the grid based particle method can also model motions of codimension two objects easily and efficiently. In this section, we give a more detailed description for closed curves in 3D which will be used in tracking the boundary of open surfaces in next section. In the grid based particle method one needs local construction of interface for sampling particles as well as computing geometric quantities. For codimension one interfaces, the normal and tangent plane is used as the local coordinate system. The key idea is that in this local coordinate system, the interface can be represented as a graph in the tangent plane and can be easily approximated by least square fitting a collection of neighboring particles on the interface.

For a codimension two interface, such a curve in 3D, the representation is the same. We use meshless particles sampled according to an underlying grid, e.g. particles that are closest points corresponding to grid points in the neighborhood of the

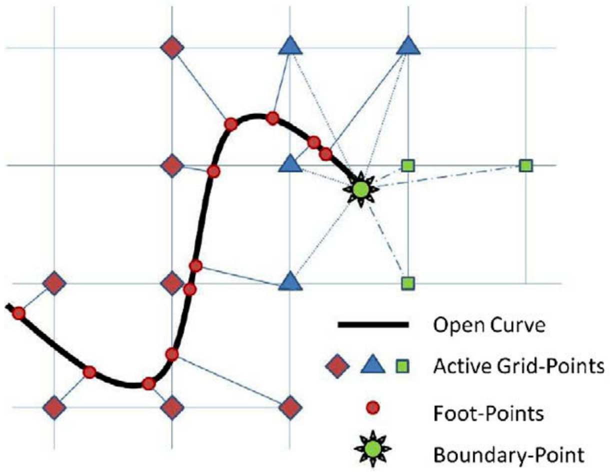


Fig. 3. Representation of the open surface and the boundary in the grid based particle method. Away from the boundary-point (the green sun-symbol), the representation is the same as in [17]. Each active grid point (red diamonds) is associated to its closest point on the interface (red circles). Near the boundary-point, each active grid point (blue triangles and green squares) is associated to its closest point (red circles or green sun-symbol) and also the closest boundary-point (green sun-symbol). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

interface. However, the local reconstruction procedure is a little different. Since a curve is of one dimension, we can parametrize it in the tangent direction locally. For instance, assume we have already collected a set of neighboring particles $\mathbf{y}_i, i = 0, 1, 2 \ldots$ around the particle $\mathbf{y}_0$ on the curve and its tangent vector $\mathbf{t}$ at $\mathbf{y}_0$. For local construction of the curve, we translate $\mathbf{y}_0$ to the original and transform the local coordinates using the Householder reflector such that $\mathbf{t}$ becomes the $(0,0,1)$-axis. By doing so, 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 use least square fitting to obtain $x(z)$ and $y(z)$ from the collected particles, respectively, using quadratic polynomials. We can compute the foot-point associated to an active grid point $\mathbf{p} = (p_1, p_2, p_3)$ expressed with respect to the local coordinates by minimizing

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

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

4. Open curves and open surface

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

4.1. Representation

As defined before, any grid point which is in a γ-neighborhood of the interface is called an active grid point. In Fig. 3, we demonstrate a typical scenario near an end-point of an open curve. Given an open curve, we first collect active grid points which are within the γ-neighborhood. Then, for each of these active grid points we find the corresponding closest points on the open curve. In the figure, we label the active grid points differently (using red diamonds, blue triangles and green squares) and their closest points differently (using red circle and green sun-symbol) to distinguish their roles.

Away from the boundary-point (green sun-symbol), the set-up is exactly the same as before. If an active grid point has a distance greater than γ away from the boundary (green sun-symbol), we simply project the grid point (red square) onto the surface locally reconstructed by the least square fitting. This gives the associated foot-point (red circle) on the surface.

For a grid point within the γ-neighborhood of the boundary, we assign it with *two* foot-points. One is again the closest point from the grid point to the open surface, while the second one is the closest point from the active grid point to the *closed boundary* and we will call this second foot-point the boundary-point. In two dimensions, the second foot-point is just the end-point itself, as shown in Fig. 3. For three dimensions, the boundary-point is exactly the sampling point we obtained in the previous section. There is no extra minimization process required to determine this second foot-point. Such a grid point will be used to sample and track both the interface and its boundary in the framework of grid based particle method in a consistent way. We will describe this further in the next subsection.

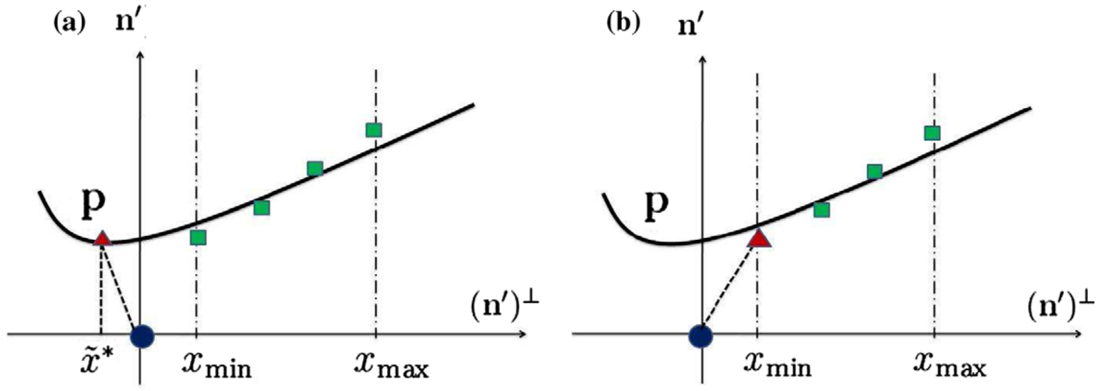


Fig. 4. (a) In the original algorithm [17], any potential new foot-point (red triangle) will be rejected if $x^* \notin [x_{\min}, x_{\max}] = [\min(x_j), \max(x_j)]$. (b) In the new algorithm, we associate the new closest point to the closest boundary-point if the corresponding active grid point is close to the boundary. In this plot, we associate the active grid point (blue circle) to the particle corresponding to $x_{\min}$ (red triangle) assuming it is a boundary-point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Note that these two foot-points could be the same or could be different. In Fig. 3, we plot these special grid points using blue triangles or green squares. Consider the blue triangles, one of their associated foot-points is still obtained by local least square approximation of the interface, which is plotted using red circles. Since these blue triangle active grid points are within γ from the end point (green sun-symbol), their second foot-points are also activated which are the closest boundary-points (the connectivity is plotted using a dotted line). Fig. 3 shows the case in two dimensions. For three dimensions, these boundary-points are in general different among different grid points. For some other active grid points near the boundary, the closest point on the open curve could just be the closest boundary-point itself. In this case, the two foot-points associated to an active grid point coincide. In Fig. 3, we plot this type of active grid points using green squares and the connectivity between the active grid points and their foot-points using a dashed-dotted line.

4.2. Motion and resampling

Now we discuss how our algorithm incorporates this boundary information in the evolution step and the resampling step. As before, the motion phase of the algorithm is relatively straight-forward. We simply move all sampling points (including both the closest points and the boundary-points) as in the usual Lagrangian type methods. The motion law of the closest points can be very general. We can naturally deal with the motion by an external velocity field or geometrical motions such as motion by the mean or the Gaussian curvature. The motion law imposed on boundary-points can be explicitly given or can be determined by local geometry of the *boundary/surface* such as the curvature or the torsion, which can be easily computed from local reconstruction of the boundary/surface as described in the previous section.

We separate the reconstruction and sampling of the boundary-points from that of the surface. For two dimensional cases, there is no need to resample end-points of the open curve since they are just explicitly tracked points. For three dimensions, we resample the boundary using only boundary-points as described in Section 3.

Away from the boundary, the resampling step follows the procedures in standard grid based particle method. However, resampling of the open surface near the boundary requires more care since we need to take into account the boundary of the open surface.

The local reconstruction phase of the algorithm is similar as before. For each of the active grid point **p**, we consider its neighboring active grids and collect a set of their corresponding closest points and, if any, also a separate set of their corresponding boundary-points. If **p** is close to the boundary of the open surface, its neighboring active grid point might be assigned *two* foot-points which might or might not be the same (the blue triangles and the green squares in Fig. 3, respectively). We will distinguish these two types of foot-points in this local reconstruction step.

If the set of boundary-points is empty, we will simply use the set of closest points for local reconstruction, as in the original algorithm in [17]. Otherwise, we will form a set of sampling points for local reconstruction using both the set of closest points and also the set of boundary-points. These sampling points have to satisfy the following two conditions. The first one is the same as what we have proposed in [17] that any two sampling points should be separated on the order of $O(h)$, where h is the local mesh size. This removes redundancy in the sampling which may cause degeneracy in the local reconstruction. The second constraint is that boundary-points in the sampling set should define at least parts of the boundary of $\widetilde{\Omega}$, where $\widetilde{\Omega}$ is the convex hull formed by the projection of the sampling points on the tangent plane. This condition is to enforce that sampling particles are confined by the boundary of the open curve or surface and this is the first place where the boundary information is incorporated in the local reconstruction.

In two dimensions, here is the way to construct this sampling set from both the closest points and the boundary-points. The set of the sampling point starts with collection of a set of closest points that are well separated as in [17]. Then we go

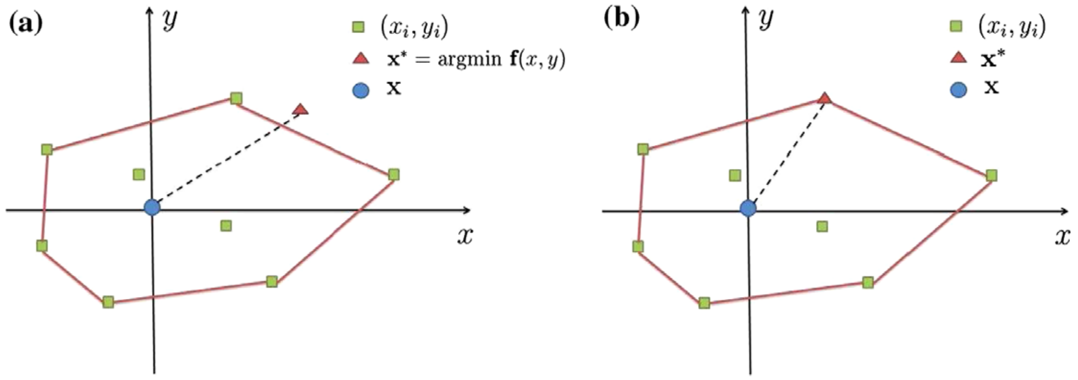


Fig. 5. (a) In the original algorithm [17], any potential new foot-point (red triangle) will be rejected if $\tilde{\mathbf{x}}^* \notin \widetilde{\Omega}$. (b) In the new algorithm, we associate the new closest point to the closest boundary-point if the corresponding active grid point is close to the boundary. In this plot, we associate the active grid point (blue circle) to the particle corresponding to the red triangle assuming that corresponds to the closest boundary-point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

through the set of neighboring boundary-points. For each boundary-point, we check if it is $O(h)$ away from all of the already collected sampling points or not. If not, then we will reject that particular boundary-point and then repeat the procedure with the next boundary-point. Otherwise, in the local coordinates system $\{(\mathbf{n}')^{\perp}, \mathbf{n}'\}$ where we denote the boundary-point by $(\hat{x}, \hat{y})$ and the set of the accepted sampling points by (x_j, y_j), we check if $\hat{x} \in [\min(x_j), \max(x_j)]$. If not, then we will add this boundary-point to the list of the sampling point. Otherwise, we will use this boundary-point to replace the sampling point corresponding to either $\min(x_j)$ or $\max(x_j)$, whichever closer to $\hat{x}$. If a boundary point is accepted it is involved in both the local reconstruction and defining end points of the reconstruction interval.

Now, with these sampling points, we construct a local least square fitting and we denote it by $y = f(x)$. To determine the new closest point, we minimize the distance from the grid point $\mathbf{p}$ to the function $y = f(x)$. If the minimum is attained at $x^* \in [\min(x_i), \max(x_i)]$, we follow the same procedure as in [17] and determine the new closest point $(x^*, f(x^*))$, accordingly. In the previous algorithm, we deactivated any grid point if this new foot-point leads to an extrapolation, i.e. if $x^* \notin [\min(x_i), \max(x_i)]$, Fig. 4(a). For active points near the boundary, we again enforce the boundary information at this step of the algorithm. We now go back and check if the set of boundary-points is non-empty, i.e. if any of the active grid points in the neighborhood is of the blue triangle or the green square type as in Fig. 3. If so, we will assign x^* to this boundary-point, for example $x^* = x_{\min}$ as shown in Fig. 4(b). Of course, if the distance between the grid point $\mathbf{p}$ and the boundary point $\mathbf{x}^*$ is larger than γ, this association will be removed. On the other hand, if none of the neighboring active grid points is associated to a boundary-point, we will simply deactivate this active grid point as in [17].

For three dimensional cases, we follow a similar procedure. To determine the set of sampling points, we combine the set of the closest points and the set of the boundary-points, if any. When we are adding a boundary-point to the set of the sampling point, the more complicated step is to check if a boundary-point on the tangent plane is inside the convex hull formed by the projection of all accepted sampling points. Assume we have collected a set of accepted sampling points and translated them to the local coordinate system which aligns the normal with z-axis and the tangent plane with $x - y$ plane as was done in [17]. Denote (x_i, y_i) to be the $x - y$ coordinates of those collected sampling points. Let $(\hat{x}, \hat{y})$ be the projection of the boundary-point on the tangent plane. We propose the following $O(m \log m)$ algorithm which efficiently determines if $(\hat{x}, \hat{y})$ lies inside the convex hull $\widetilde{\Omega}$ constructed by the set (x_i, y_i), in the $x - y$ (tangent) plane, see Fig. 5(a). There could be better (in the sense of the computational efficiency or the easiness in implementation) algorithm for the job. But since it will not improve the overall complexity of the algorithm, we will not discuss in details here.

Algorithm. to check if $(\hat{x}, \hat{y}) \in \widetilde{\Omega}$

(1) Compute the angle θ_i made between $(x_i - \hat{x},\ y_i - \hat{y})$ and the positive x-axis.
(2) Sort θ_i in an ascending order.
(3) If the absolute value of the difference between any two adjacent θ_i's is great than π, the point $(\hat{x}, \hat{y})$ lies outside $\widetilde{\Omega}$. Otherwise, $(\hat{x}, \hat{y})$ lies inside the convex hull.

The proof of this algorithm is straight-forward. Without loss of generality, we assume $(\hat{x}, \hat{y}) \neq (x_i, y_i)$ is the origin 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 $\tilde{\theta}_i \in (0, \pi), \forall i$. Denote $(\tilde{x}_i, \tilde{y}_i)$ the coordinates of (x_i, y_i) after the rotation. We have $\tilde{y}_i > 0$ and therefore, $0 \notin \widetilde{\Omega} = \left\{(x, y) = \sum_i \alpha_i (\tilde{x}_i, \tilde{y}_i), \alpha_i \geqslant 0, \sum_i \alpha_i = 1\right\}$, the convex hull constructed by $(\tilde{x}_i, \tilde{y}_i)$.

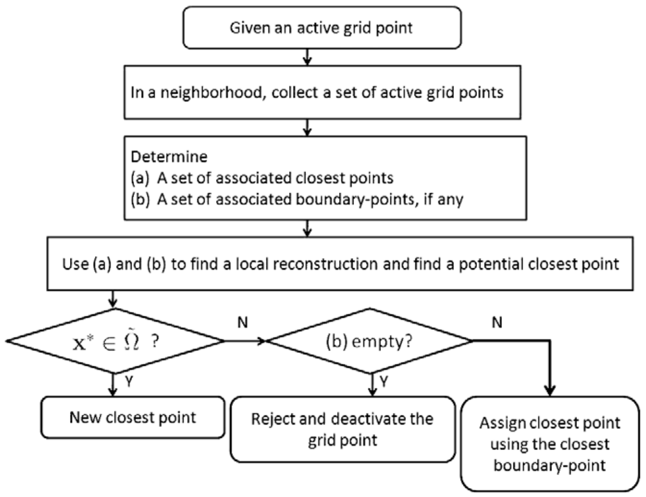

Given an active grid point
In a neighborhood, collect a set of active grid points
Determine
(a) A set of associated closest points
(b) A set of associated boundary-points, if any
Use (a) and (b) to find a local reconstruction and find a potential closest point
x* ∈ Ω̃ ?
N
(b) empty?
N
Y
Y
New closest point
Reject and deactivate the grid point
Assign closest point using the closest boundary-point

Fig. 6. Incorporating the boundary information in the resampling phase. For each active grid point in the resampling step, we summarize the procedure to determine how we assign new closest point or deactivate the corresponding grid point.

Again, if the boundary-point is at least $O(h)$ away from all sampling points and $(\hat{x}, \hat{y}) \notin \widetilde{\Omega}$, we accept this particular boundary-point as a sampling point. Otherwise, we will use this boundary-point to replace the sampling point corresponding to one of the (x_i, y_i) on the boundary of $\widetilde{\Omega}$, whichever closest to $(\hat{x}, \hat{y})$. This particular sampling point can be found by first sorting

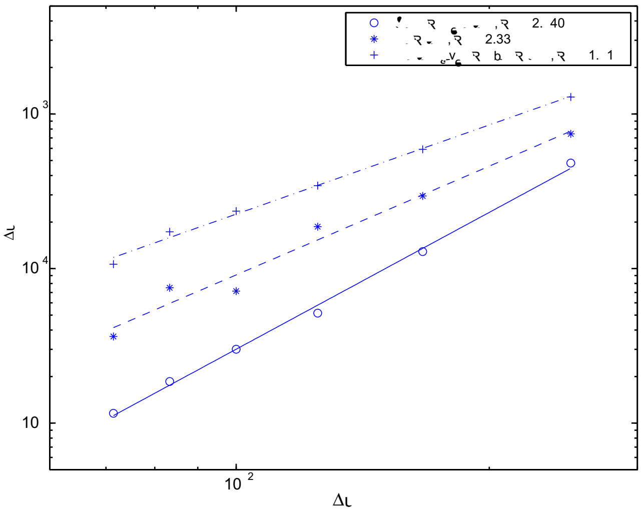


Fig. 7. Convergence analysis using the L^2 norm for the simple translation (circle and solid line), the rigid body rotation (star and dashed line) and the motion by mean curvature (plus and dashed-dot line). The convergence rates for these motions are approximately 3, 2 and 2, respectively.

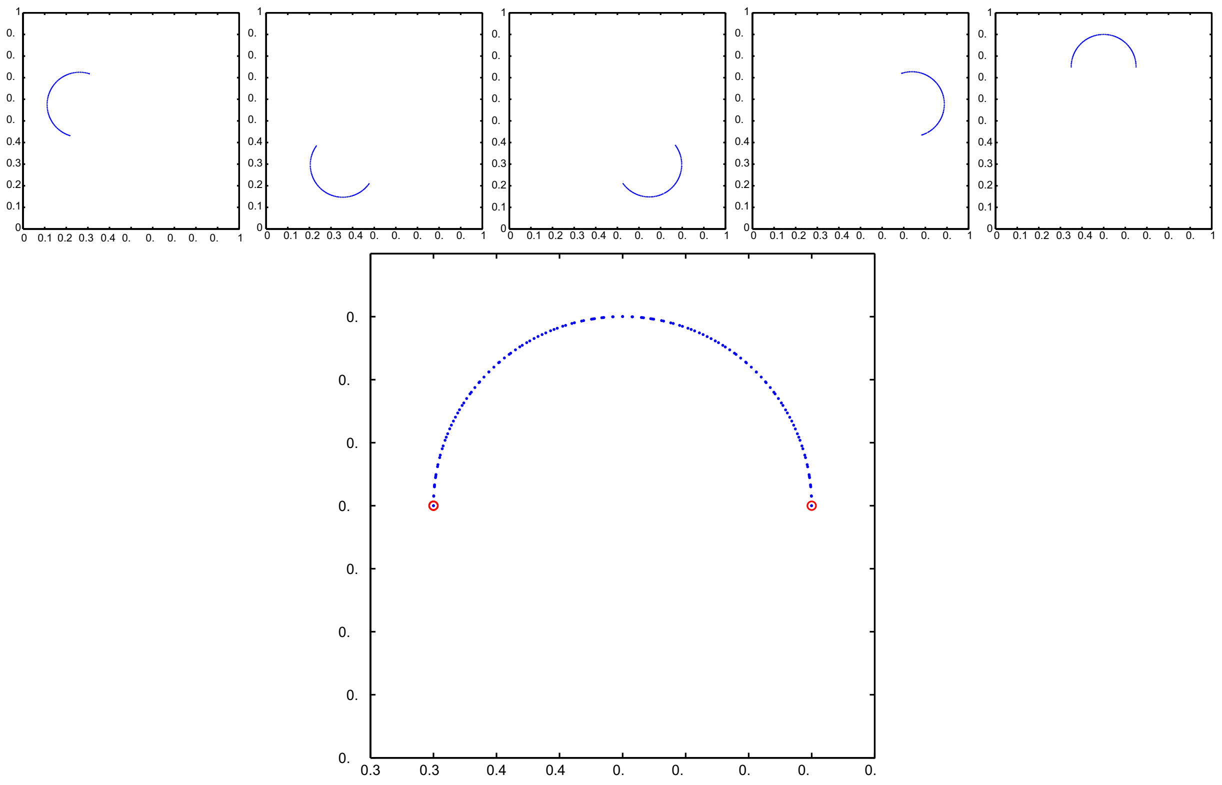


Fig. 8. Rotation of an upper half circle using an underlining uniform mesh of resolution 129^2. The second row shows the solution at the final time $t = \pi$ with the exact endpoint locations plotted in red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

all sampling points on the tangent place according to the distance to $(\hat{x},\hat{y})$ in an ascending order and then determining the first one on the boundary of $\widetilde{\Omega}$.

The next step is to obtain the local reconstruction of the surface $z = f(x,y)$ by least square fitting. Let $\mathbf{x}^* = (x^*, y^*, f(x^*, y^*))$ be the closest point of the active grid point $\mathbf{p}$. We again use the above $O(m \log m)$ algorithm to determine if (x^*, y^*) lies inside the convex hull $\widetilde{\Omega}$ constructed by the set $(x_i, y_i), i = 1, \ldots, m$ in the $x - y$ (tangent) plane, see Fig. 5(a). If so, then we accept $\mathbf{x}^*$ and assign the corresponding point on the surface to be the new foot-point. Otherwise, we again check if the set of boundary-points is non-empty, i.e. if any of them is of the blue triangle or the green square type as in Fig. 3. If so, we will associate it to the closest point on the boundary of the open surface, Fig. 5(b). Otherwise, we will simply remove this particular active grid point from the computational tube and will also delete the corresponding associated particle.

We have summarized the algorithm in the resampling phase in Fig. 6. All other steps in our algorithm will be the same as described in [17]. We approximate any Lagrangian information associated to this particle using the local reconstruction of the surface. For example, the normal vector at this new foot-point is approximated by the normal vector of the local reconstruction at $\mathbf{x}^*$. The curvature and the global parametrization can also be updated accordingly.

5. Examples

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 six particles in two dimensions and 10 particles in three dimensions. The time integration is done using the total variation diminishing second order Runge–Kutta (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 simply plot the solution (interface location) using unconnected dots which shows the foot-point 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 functions isosurface and patch.

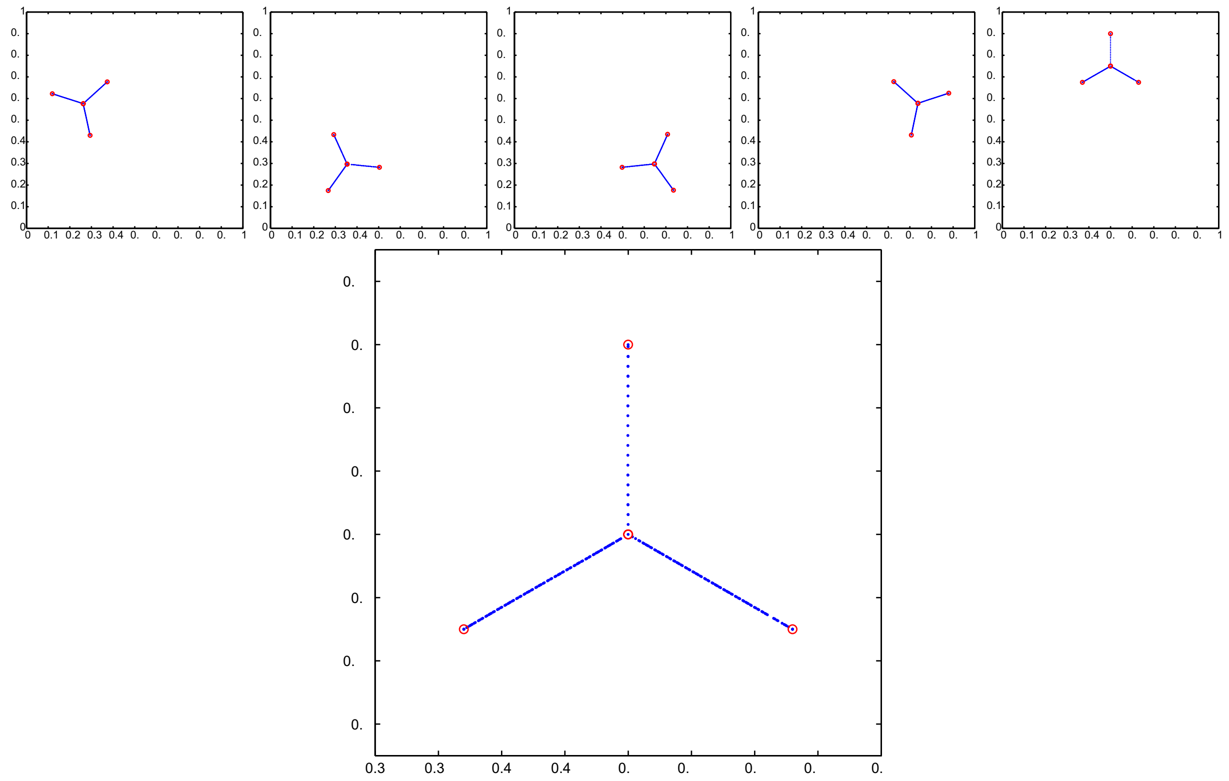


Fig. 9. Rotation of a single triple junction consisting of three open segments using an underlining uniform mesh of resolution 129^2. The second row shows the solution at the final time $t = \pi$ with the exact endpoint locations plotted in red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.1. Open curve

5.1.1. Simple translation and rigid body rotation

In the first example, we consider an upper half of a circle of radius 0.1 initially centered at (0.25,0.25) moving under a simple translation $(u, v) = (1, 1)$. At time $t = 0.5$, the half circle with the same radius will be centered at (0.75,0.75). We do not have a theoretical estimates on the order of convergence, but numerical experiments show that the proposed method has similar convergent rates as stated in [17]. For instance, for this simple translation motion, the error at $t = 0.5$ converges to zero like $O(\Delta x^3)$. Fig. 7 shows (log-plot) the convergence as we refine the underlying grid. For a fixed Δx, we define the following L^2-error made at all foot-points by

$$E_{\Delta x} = \left[\int |\mathbf{y}_{\text{exact}} - \mathbf{y}_{\Delta x}|^2 dx \right]^{1/2}. \tag{3}$$

The solid line shows log-plot of the least square fitting of the error in the form

$$E_{\Delta x} = c_1 (\Delta x)^{c_2}, \tag{4}$$

whose slope gives an approximation to c_2 and the rate of convergence.

Next we consider the rigid body rotation of a half circle of radius 0.15. The velocity is given by

$$\begin{aligned} u &= 1 - 2y, \\ v &= 2x - 1. \end{aligned} \tag{5}$$

The curve will rotate around the point (0.5,0.5) in a period of π. Solutions at $t = m\pi/5$ for $m = 1, \dots, 5$ are shown on the top row in Fig. 8. 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. Similar to the simple translation motion, we also perform a convergence analysis, the stars and the dashed line in Fig. 7. The convergent rate is of $O(\Delta x^2)$, which is due to using RK2 in the time discretization.

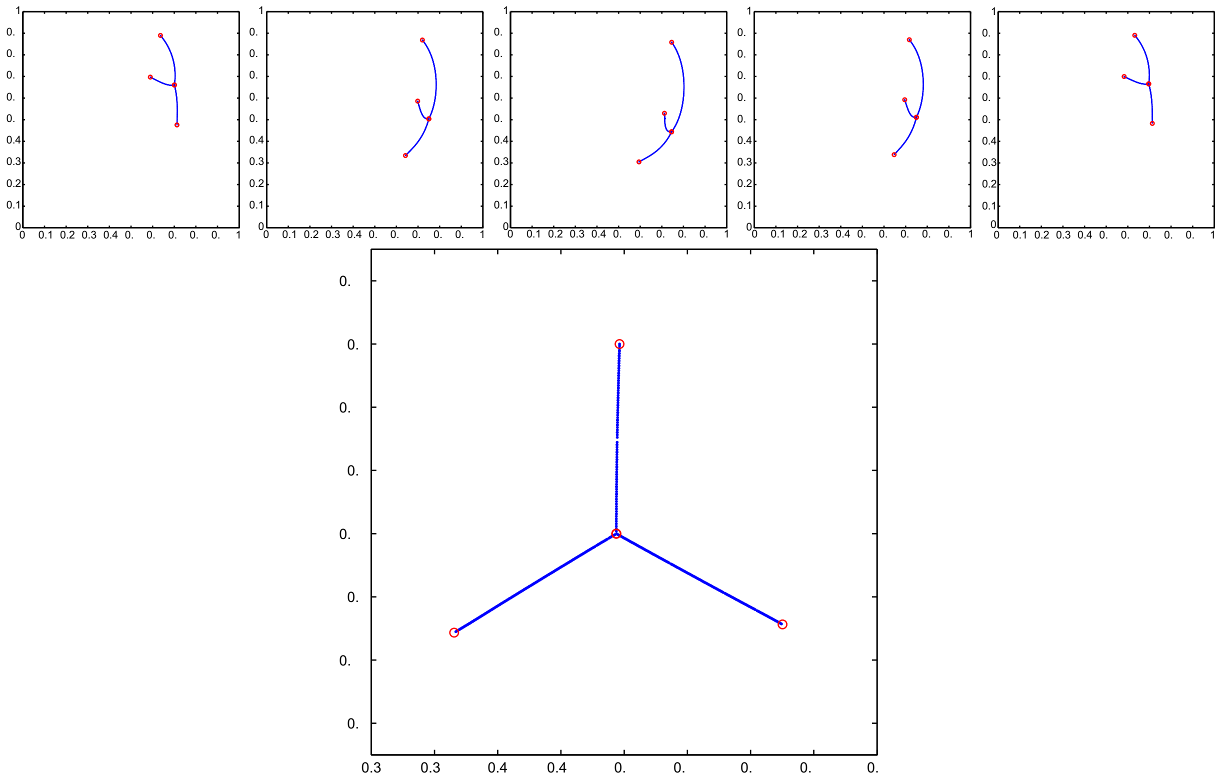


Fig. 10. Motion of a single triple junction under the time-reversal vortex flow at $t = 0.25, 0.50, \dots, 1.25$. The single triple junction consists of three open segments. The solution is computed using an underlining uniform mesh of resolution 513^2. The second row shows the solution at the final time $t = 1.5$ with the exact endpoint locations plotted in red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.1.2. Motions of a triple junction

One potential application of the proposed algorithm is on motions of a triple junction. In this paper, we represent a triple junction by three open segments with a single common open end-point at the middle of the junction. This single junction point will be explicitly tracked as a boundary point we have discussed above.

Fig. 9 shows the simple rigid body rotation of a single triple junction at $t = \pi/5, 2\pi/5, \dots, \pi$. The length of each arm of the triple junction is 0.15 with angles between any two adjacent arms is $2\pi/3$. The initial center of the triple junction is at (0.5,0.75). Under the same rigid body flow as in the previous section, the object will rotate about (0.5,0.5) in a period of π.

We have also consider the following simple vortex flow [10,14,19] which acts as an important test case for various numerical methods. It was originally proposed by Bell et al. [3] to test if a numerical method is able to resolve very thin filaments. The velocity field is defined by the following stream function

$$\Psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y). \tag{6}$$

Following [18], we study the time-reversal version of the velocity field by multiplying it by $\cos(\pi t/T)$. Fig. 10 shows the motion of the same triple junction as in the simple rigid body rotation for $t = 0.25, 0.50, \dots, 1.25$ and $T = 1.5$ using an underlining uniform mesh of resolution 513^2. At the final time $t = 1.5$, i.e. second row in Fig. 10, the triple junction should have the same shape as $t = 0$.

5.1.3. 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. Fig. 11 shows the solutions at $t = 0.5m$ for $m = 1, \dots, 5$. 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 ordinary differential equation (ODE) solver.

5.1.4. Motion by mean curvature

We next consider the motion by mean curvature of an upper half 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-

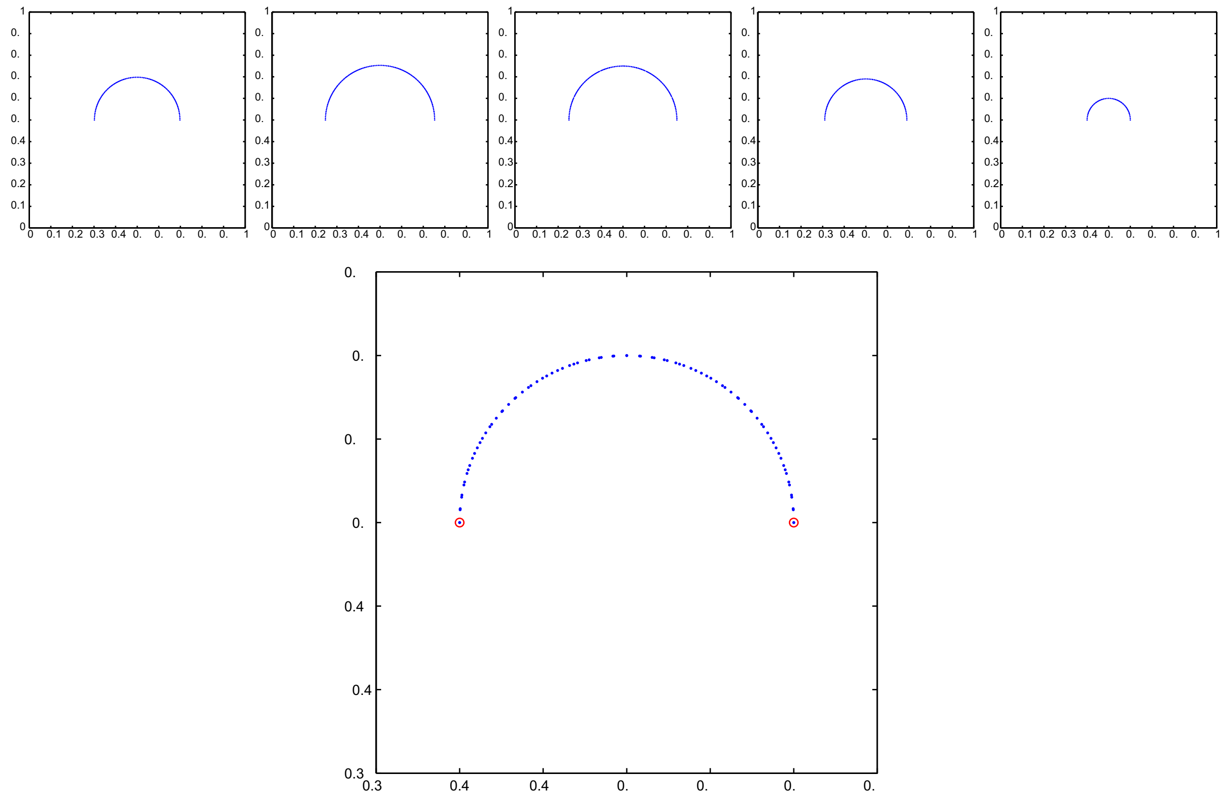


Fig. 11. Normal motion of an upper half circle using an underlining uniform mesh of resolution 129^2. The half circle first expands in the outward normal direction for $t = 0.25$ and then collapses in the inward normal direction for $t = 0.25$. The second row shows the solution at the final time $t = 0.5$ with the exact endpoint locations plotted in red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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

$$\mathbf{v}(\mathbf{y};t) = -\mathbf{n}(\mathbf{y};t)/|x - x_c|. \tag{7}$$

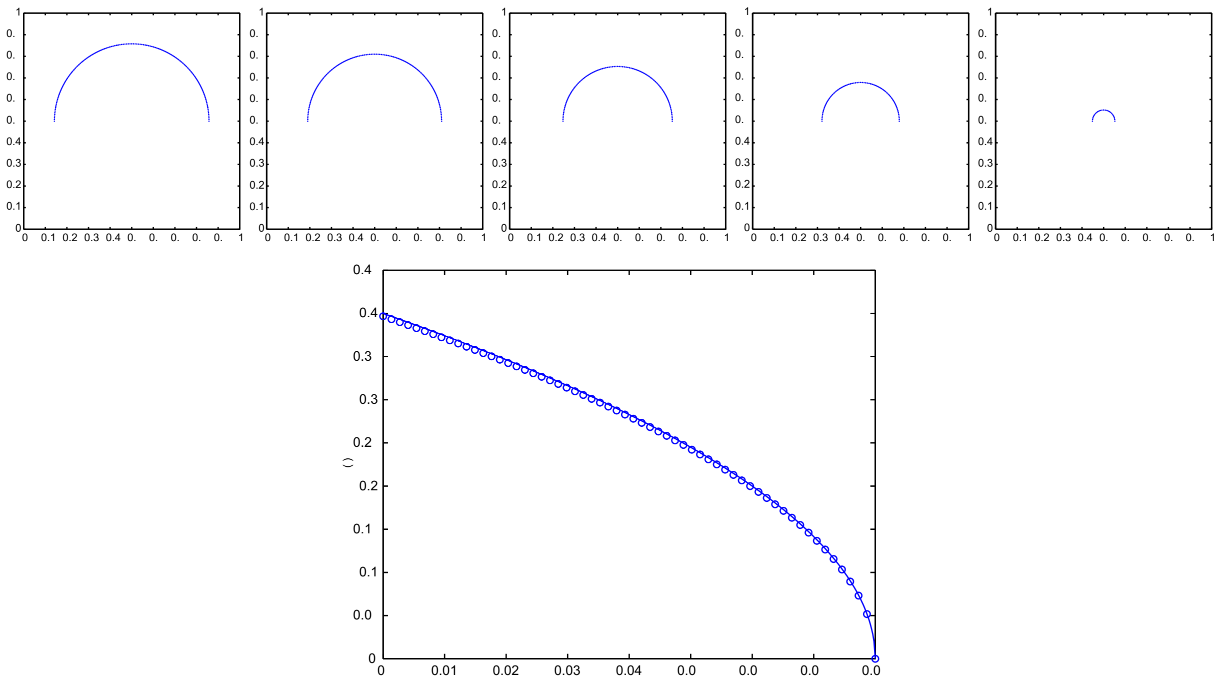


Fig. 12. Normal motion of an upper half circle using an underlining uniform mesh of resolution 129^2. The second row shows the change in the mean distance from the center (0.5,0.5). The solid line is the exact distance, the blue circle is the computed solution. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

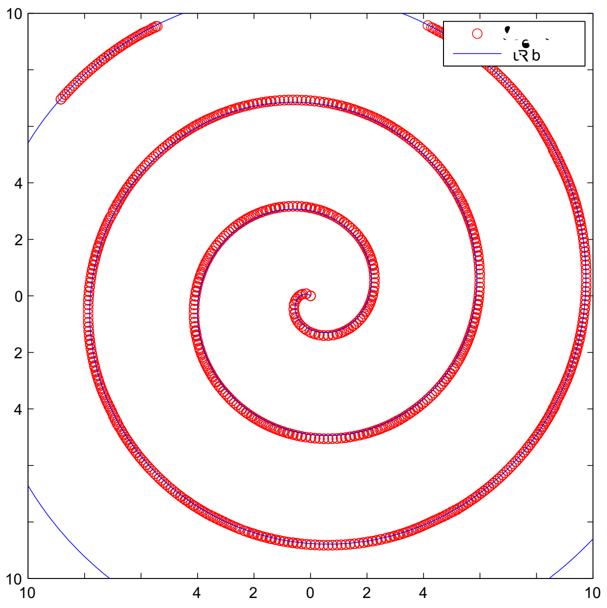


Fig. 13. Motion of a step-line at $t = 25$ with $\lambda = 0.2$ using an underlining uniform mesh of resolution 129^2. The approximate solution in [6] is plotted in solid line, while our computed solution is shown by red circles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The solutions at various times are shown in Fig. 12. Fig. 12 uses an underlying mesh of 129 grids in each direction. On the second row, we plot the change in both the total number of particles and also the mean radius of the circle. The computed mean radius (the circles) matches with the exact solution (solid line) very well. In Fig. 7, we have also shown the convergence of the algorithm under this motion. We start with the same initial circle of radius $r = 0.4$ and stop at a fixed time $t = 0.05$. Since the underlying PDE is non-linear parabolic, which imposes a more restrictive CFL condition of $\Delta t = O(\Delta x^2)$. Numerically we pick $\Delta t = 0.5\Delta x^2$. The rate of convergence is $O(\Delta x^2)$.

5.1.5. Spiral crystal growth

As an interesting application of the proposed algorithm, we follow [23] and compute the motion of a slip-line in the spiral crystal growth. Proposed in [6], 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

$$\mathbf{v} = (1 - \lambda\kappa)\mathbf{n}, \tag{8}$$

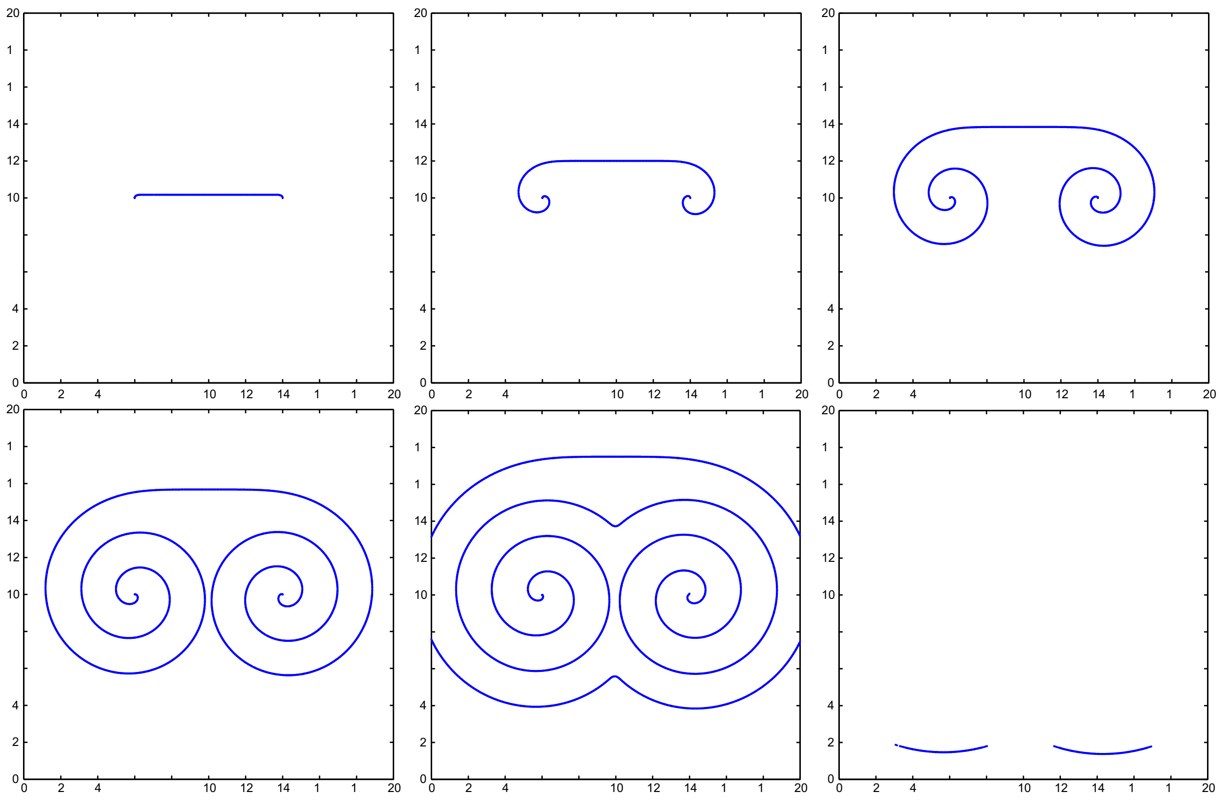
with the initial normal pointing downward.

In Fig. 13, we consider the motion of 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). We fix the end-point at the origin. Assuming that the slip-line is almost a straight line ($\kappa \simeq 0$) near the other end-point, we impose the motion law $(u, v) = (0, 1)$ on this moving end. The critical radius λ equals 0.2. Fig. 13 shows the computed solution (red circles) at $t = 25$ together with an approximate solution (blue solid line) stated in [6,23]

$$\theta = \frac{\sqrt{3}}{2(1+\sqrt{3})}\left[\frac{r}{\lambda} + \log\left(1 + \frac{r}{\lambda\sqrt{3}}\right)\right] + \theta_0(t). \tag{9}$$

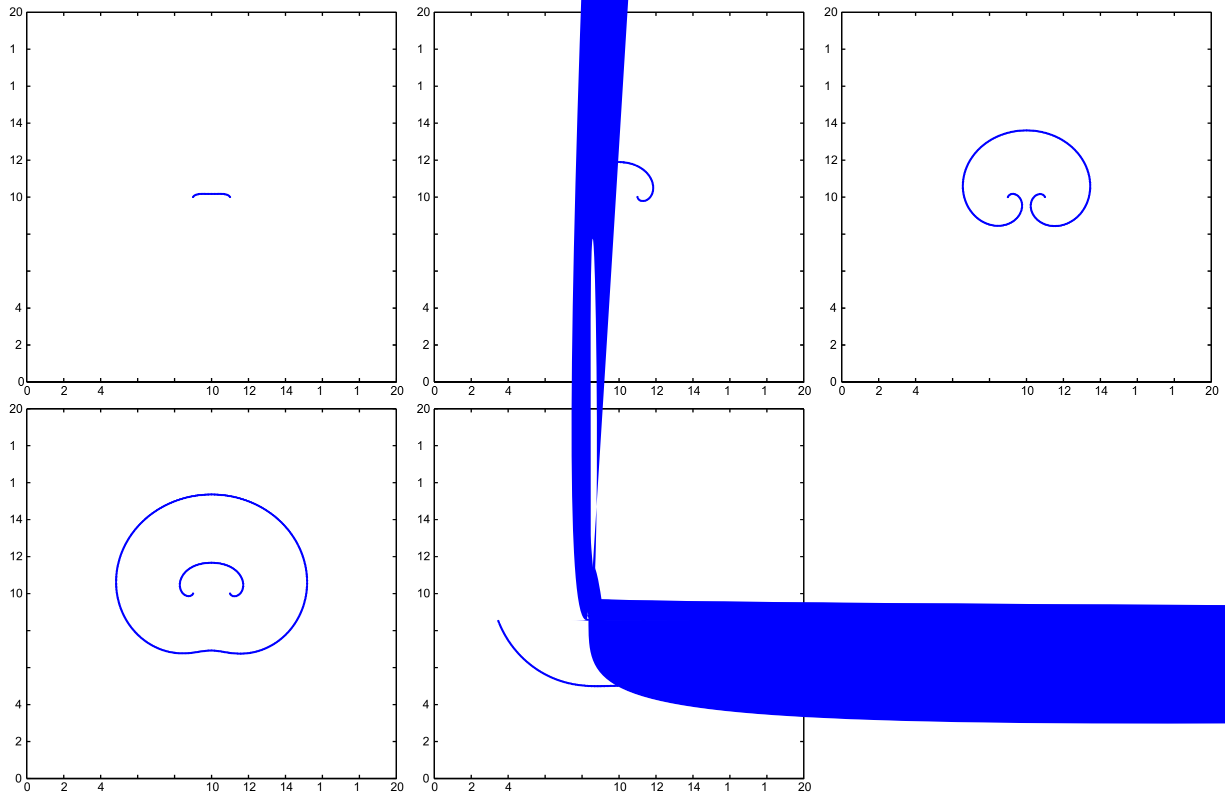
These two solutions match very well.

Two other examples are also taken from [23]. The first example uses an initial line segment of length 8 with $\lambda = 0.1$. The solutions at various time up to $t = 10$ are plotted in Fig. 14. Another set-up uses a line segment of length 2 with $\lambda = 0.2$, Fig. 15. Topological change in these solutions are nicely captured using an underlying uniform mesh of only 257 grids in each direction.



5.1.6. *Active contour for image segmentation*

Segmentation is one of many fundamental tasks in ima[illegible]rocessing. Since [15], extensive research have been done on applying variational approach to detect boundaries of obje[illegible]n an image. In [15], 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

$$\mathscr{C} = \{c : [a,b] \to \Omega, c \text{ piecewise } C^1\} \tag{10}$$

for some domain Ω for the observed image, the *snake* evolves to minimize the functional

$$\widetilde{E}(c) = \int_a^b |c'(q)|^2 dq + \beta \int_a^b |c''(q)|^2 dq + \lambda \int_a^b g^2[|G * \nabla u_0(c(q))|] dq, \tag{11}$$

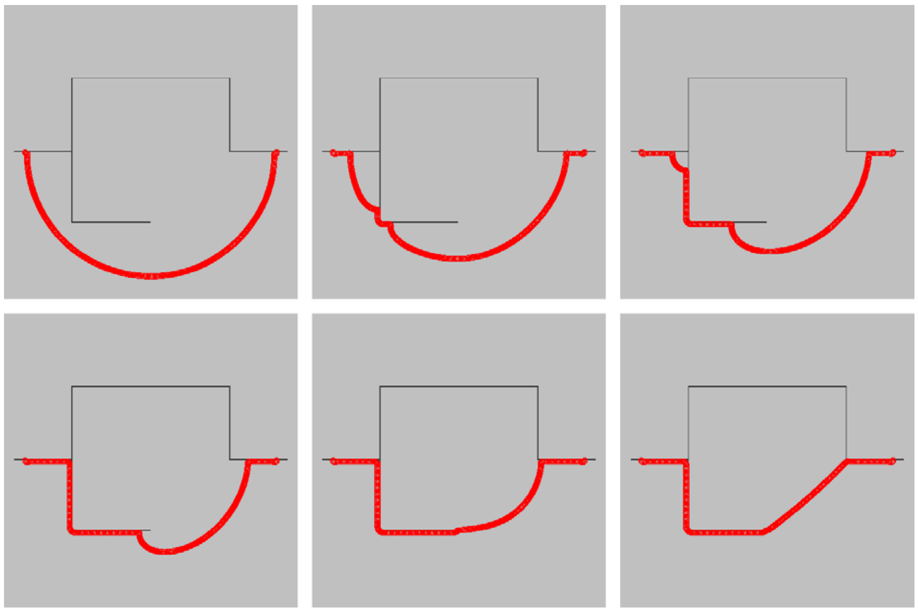


Fig. 17. Evolution of the active contour (red curve) for detecting edges. Two end-points are fixed in the motion. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

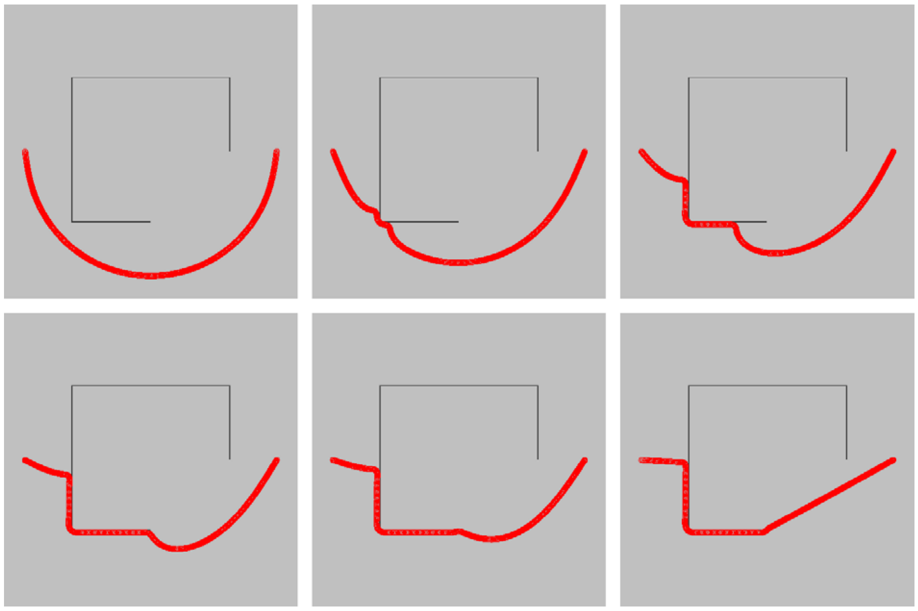


Fig. 18. Evolution of the active contour (red curve) for detecting edges. Two end-points are fixed in the motion. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where G is the usual Gaussian kernel, $*$ is the convolution operation, $u_0 : \Omega \to [0,1]$ is the observed image and g is an edge detection function given by

$$g(\xi) = \frac{1}{1+|\xi|^2}. \tag{12}$$

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

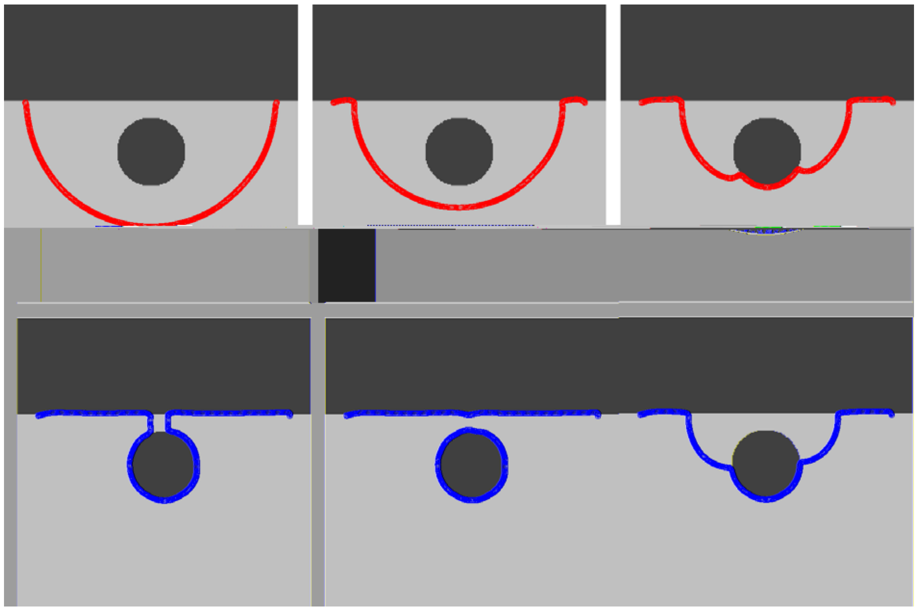


Fig. 19. Evolution of the active contour (red curve) for detecting edges. Two end-points are fixed in the motion. Topological change can be naturally incorporated in the model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

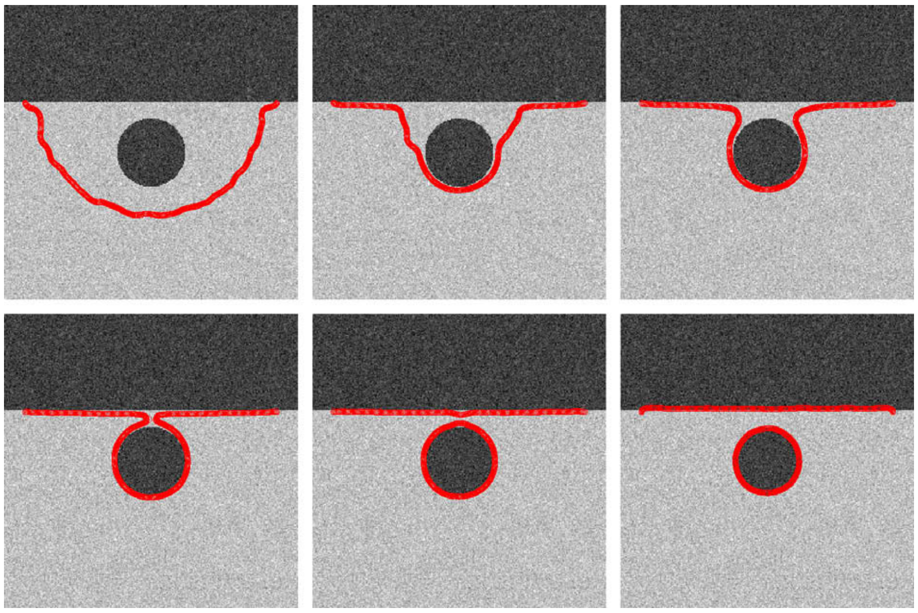


Fig. 20. Evolution of the active contour (red curve) for detecting edges in an noisy image. Two end-points are fixed in the motion. Topological change can be naturally incorporated in the model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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 of many very important improvements is the geodesic active contour model [7] in which one minimizes the energy

$$E(c) = \int_a^b g(|G * \nabla u_0(c(q))|)|c'(q)|dq. \tag{13}$$

The resulting evolution equation is given by

$$\frac{\partial c}{\partial t} = [\kappa g - \mathbf{n} \cdot \nabla g]\mathbf{n}. \tag{14}$$

To handle topological change, the authors in [7] modeled 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 [22,20,1].

In this paper, we follow [7] and evolve the contour according to

$$v_n = \kappa g - \mathbf{n} \cdot \nabla g. \tag{15}$$

Unlike the usual geodesic active contour, we do not implicitly define the curve using the level set method. Moreover, we are considering an open curve with both end-point fixed. Various active contour models could deal with open curves [12,11], but all open curves were discretized in the Lagrangian formulation. It is therefore, not easy to handle topological change of the solution.

In Fig. 16, 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 comes to the boundary of an object (the black curve in this case).

When we flip the image upside down, as shown in Fig. 17, 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 [22,20,1]. 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. In this example, in particular, the exact/global minimizer (the minimizer which gives the smallest energy) should be a line going through the upper part of the object, i.e. the red curve obtained by flipping the red curve in Fig. 16 upside down. However, since the energy is not convex, the steady state solution to the evolution Eq. (15) depends on the initial condition, as demonstrated in Figs. 16 and 17. Global minimizer of active contour models can be found using different approaches [16,8,4], but these approaches are not being studied in the current paper.

Now, we slightly modify the image by removing two segments, as shown in Fig. 18. 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 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, the curve is trapped in a local minimizer and the segmentation result depends highly on the

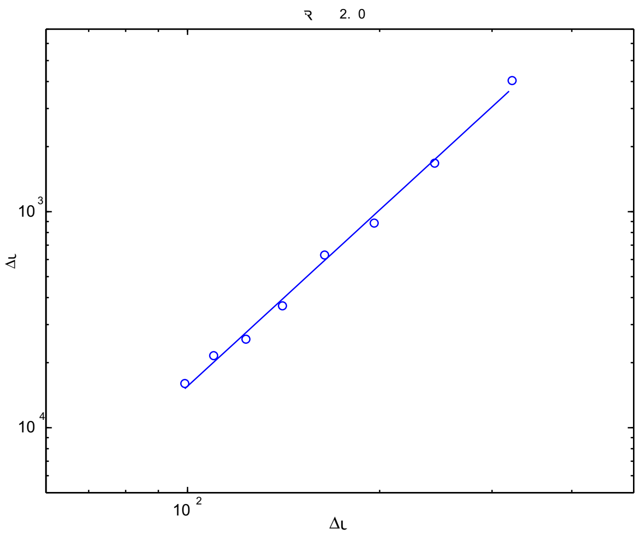
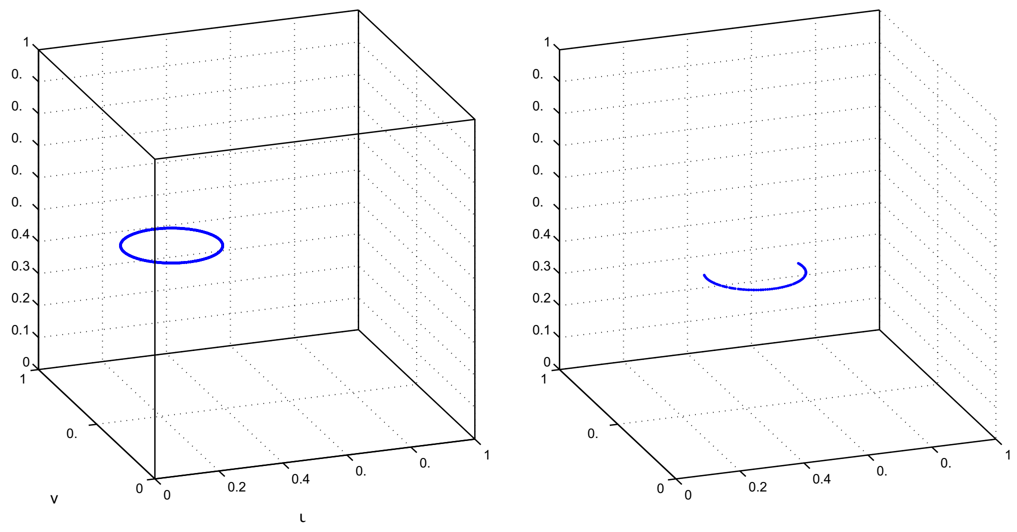


Fig. 21. Convergence analysis using the L^2 norm for the simple translation. The convergence rates is approximately 3.



initial guess. Indeed, the global minimizer for this example should be the path going from the upper route taking only piece-wise horizonal and vertical segments.

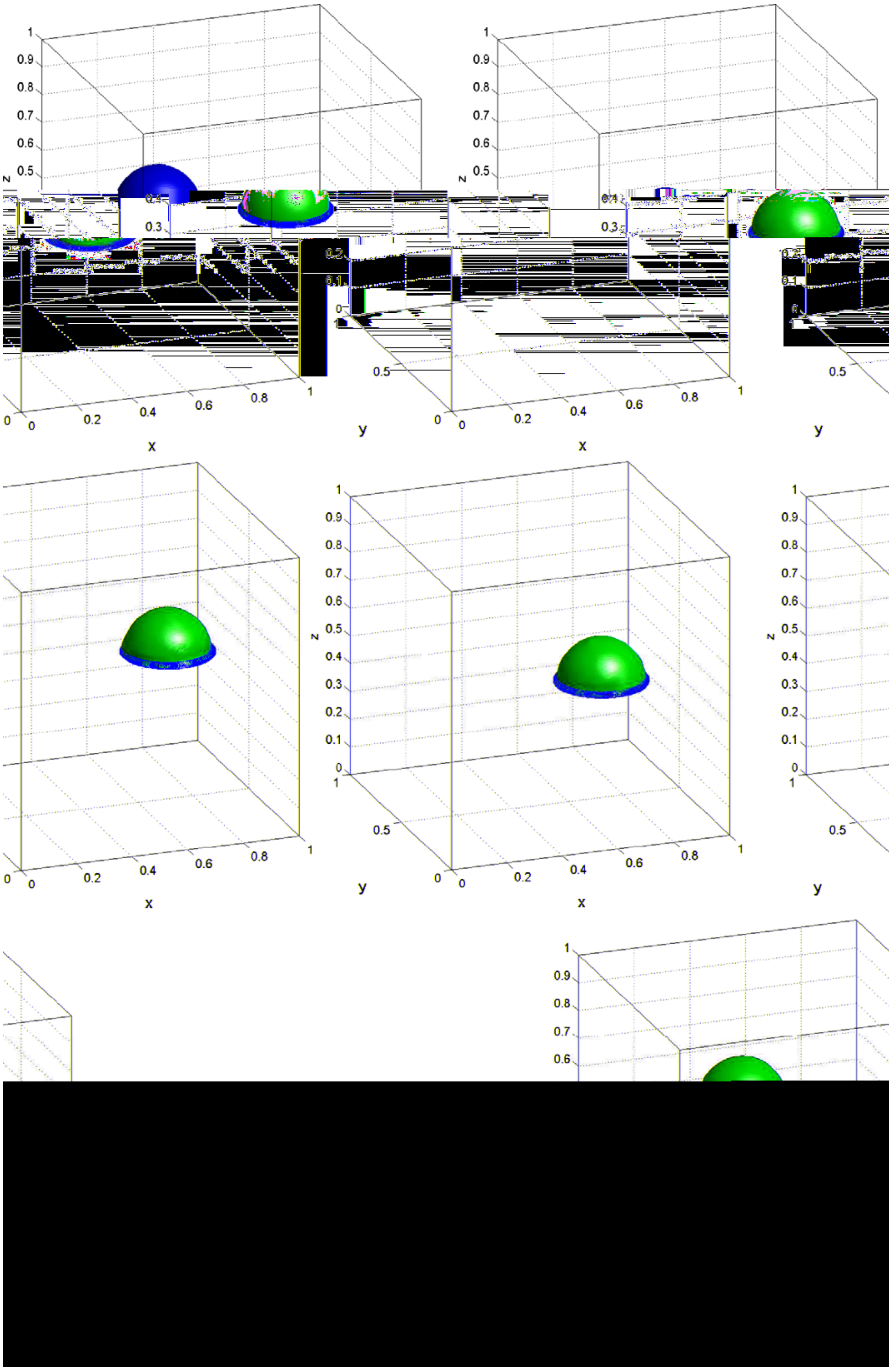
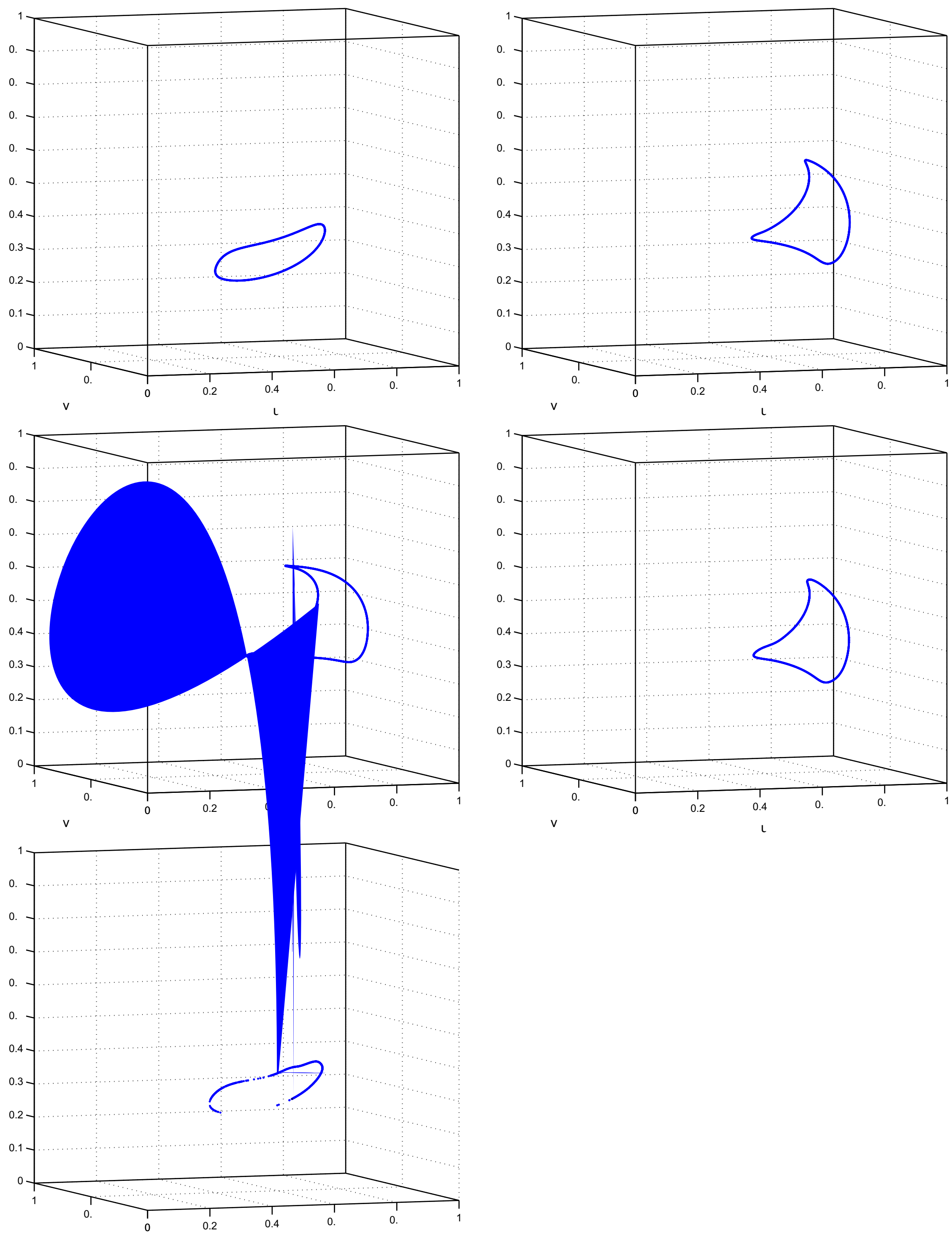


Fig. 23. Motion of an upper hemisphere under the rigid body rotation using an uniform mesh of resolution 151^3. The boundary of the surface is shown using red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



Next we consider an image for which we have a topological change in the evolution of the active contour. Figs. 19 and 20 use a clean image and a noisy image, respectively, 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

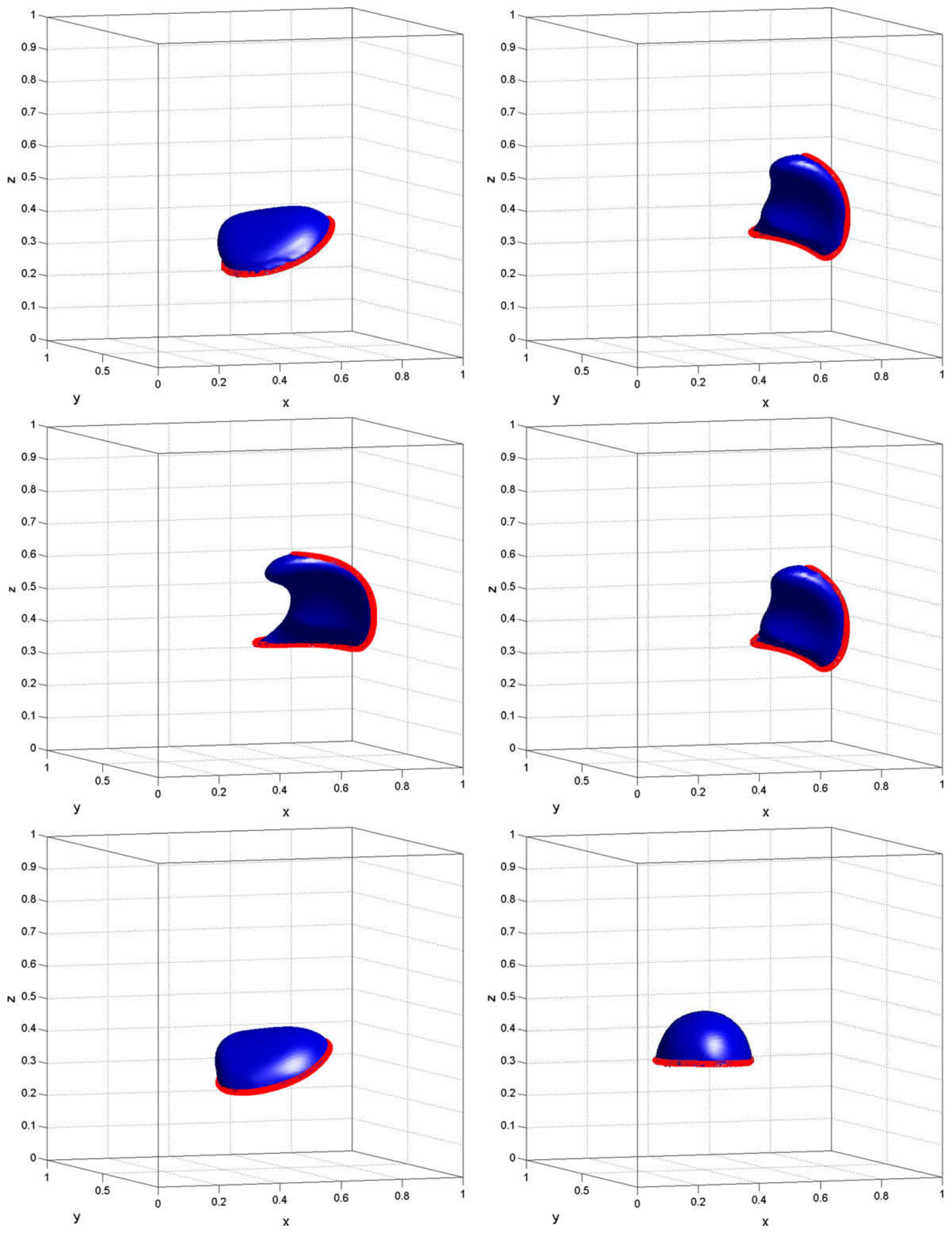


Fig. 25. Motion of an upper hemisphere under the vortex motion with time-reversed motion using $T = 1.5$ and an underlying uniform mesh of resolution 151^3. The boundary of the surface is shown using red circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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 and challenging. In the first example, we will consider the simple translation motion $(u, v, w) = (1, 1, 1)$ of an upper hemisphere of a sphere of radius 0.2 initially centered at (0.25,0.25,0.25) for $t = 0.5$. We follow a similar definition of (3) and use it to study the rate of convergence. Fig. 21 shows (log-plot) the convergence as we refine the underlying grid. The error converges to zero at an approximately $O(\Delta x^3)$ rate under grid refinement.

Next, we consider simple rotation of an upper hemisphere of a sphere of radius 0.15 initially centered at (0.5,0.75,0.5) for $t = \pi$ under the velocity field

$$\begin{aligned} u &= 1 - 2y, \\ v &= 2x - 1, \\ w &= 0. \end{aligned} \tag{16}$$

The boundary of the upper hemisphere is a circle centered at (0.5,0.75) with radius 0.15 on the plane $z = 0.5$. Its evolution is computed by the approach proposed in Section 3 and the solutions at various times are shown in Fig. 22. The whole rotation of the upper hemisphere is plotted in Fig. 23, where the solutions in Fig. 22 are drawn using red circle. As we can see, the extra condition on the boundary location described in Section 4 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 [10] where the velocity field is given by

$$\begin{aligned} u_1(x,y,z) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z), \\ u_2(x,y,z) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z), \\ u_3(x,y,z) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z). \end{aligned} \tag{17}$$

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 [10,14,19], 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 Fig. 24, we plotted the evolutions of only 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 Fig. 25, we show the motion of the whole upper hemisphere with the boundary-points from Fig. 24 shown by red circles.

References

[1] G. Aubert, P. Kornprobst, Mathematical Problems in Image Processing – Partial Differential Equations and the Calculus of Variations, Springer, 2006.
[2] S. Basu, D.P. Mukherjee, S.T. Acton, Implicit evolution of open ended curves, in: IEEE International Conference on Image Processing, vol. 1, 2007, pp. 261-264.
[3] J.B. Bell, P. Colella, H.M. Glaz, A second order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[4] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, S. Osher, Fast global minimization of the active Contour/Snake model, J. Math. Imaging Vision 28 (2007) 151–167.
[5] P. Burchard, L.-T. Cheng, B. Merriman, S. Osher, Motion of curves in three spatial dimensions using a level set approach, J. Comput. Phys. 170 (2) (2001) 720–741.
[6] W.K. Burton, N. Cabrera, F.C. Frank, The growth of crystals and the equilibrium structure of their surface, Philos. Trans. Roy. Soc. Lond. 243A (1951) 299–358.
[7] V. Caselles, R. Kimmel, G. Sapiro, Geodesic active contours, Int. J. Comput. Vision 22 (1) (1997) 61–79.
[8] T.F. Chan, S. Esedoglu, M. Nikolova, Algorithms for finding global minimizers of image segmentation and denoising models, SIAM J. Appl. Math. 66 (2006) 1632–1648.
[9] L.-T. Cheng, P. Burchard, B. Merriman, S. Osher, Motion of curves constrained on surfaces using a level-set approach, J. Comput. Phys. 175 (2002) 604–644.
[10] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[11] P. Fua, Y.G. Leclerc, Model driven edge detection, Machine Vision Appl. 3 (1990) 45–56.
[12] J. Gilles, B. Collin, Fast probabilistic snake algorithm, ICIP 3 (2003) 14–17.
[13] J. Gomes, O. Faugeras, Using the vector distance functions to evolve manifolds of arbitrary codimension, Lect. Notes Comput. Sci. 2106 (2001) 1–13.
[14] S. Hieber, P. Koumoutsakos, A lagrangian particle level set method, J. Comput. Phys. 210 (2005) 342–367.
[15] M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contour models, Int. J. Comput. Vision 1 (4) (1988) 321–331.
[16] S. Leung, S. Osher, Fast global minimization of the active contour model with tv-inpainting and two-phase denoising, in: Proceeding of the 3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision, 2005, pp. 149–160.
[17] S. Leung, H.K. Zhao, A grid based particle method for moving interface problems, J. Comput. Phys. 228 (2009) 2993–3024.
[18] R. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (1996) 627.
[19] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, J. Comput. Phys. 225 (2007) 300–321.
[20] S. Osher, R.P. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2003.
[21] S.J. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[22] G. Sapiro, Geometric Partial Differential Equations and Image Analysis, Cambridge University Press, 2001.
[23] P. Smereka, Spiral crystal growth, Physica D 138 (2000) 282–301.
[24] J.E. Solem, A. Heyden, Reconstructing open surfaces from image data, Int. J. Comput. Vision 69 (2006) 267–275.
[25] G. Ventura, J.X. Xu, T. Belytschoko, A vector level set method and new discontinuity approximations for crack growth by EFG, Int. J. Numer. Methods Eng. 54 (2002) 923–944.