Published in final edited form as:
Int J Comput Vis. 2009 August 1; 84(1): 63–79. doi:10.1007/s11263-009-0231-3.

A Moving Grid Framework for Geometric Deformable Models

Xiao Han[1], **Chenyang Xu**[2], and **Jerry L. Prince**[3]

[1] CMS Software, Elekta Inc., St. Louis, MO 63043

[2] Siemens Corporate Research, Princeton, NJ 08540

[3] Center for Imaging Science, Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218

Abstract

Geometric deformable models based on the level set method have become very popular in the last decade. To overcome an inherent limitation in accuracy while maintaining computational efficiency, adaptive grid techniques using local grid refinement have been developed for use with these models. This strategy, however, requires a very complex data structure, yields large numbers of contour points, and is inconsistent with the implementation of topology-preserving geometric deformable models (TGDMs). In this paper, we investigate the use of an alternative adaptive grid technique called the moving grid method with geometric deformable models. In addition to the development of a consistent moving grid geometric deformable model framework, our main contributions include the introduction of a new grid nondegeneracy constraint, the design of a new grid adaptation criterion, and the development of novel numerical methods and an efficient implementation scheme. The overall method is simpler to implement than using grid refinement, requiring no large, complex, hierarchical data structures. It also offers an extra benefit of automatically reducing the number of contour vertices in the final results. After presenting the algorithm, we demonstrate its performance using both simulated and real images.

Keywords

Adaptive grid method; Geometric deformable model; Deformation moving grid; Topology preservation; Level set method

1 Introduction

Active contour and surface models, also called deformable models, have become one of the most widely used tools in image segmentation and shape modeling since their introduction by Kass et al. [1]. Deformable models are classified as either *parametric deformable models* (PDMs) (cf., [1–3]) or *geometric deformable models* (GDMs) (cf., [4–8]) according to their representation and implementation. In particular, PDMs are represented explicitly as parameterized curves or surfaces in a Lagrangian formulation. GDMs, on the other hand, are represented implicitly as level sets of higher-dimensional level set functions which evolve according to an Eulerian formulation [9].

Advantages of GDMs over PDMs include computational stability, straightforward application in higher dimensions, and topological flexibility. The last property, however, is not always desirable. In particular, when a specific object (target) is sought and its composition — i.e.,

Corresponding Author: J.L. Prince, Prince@jhu.edu.

the number of components and the homology of each component — is known, then it is most natural to seek the target in a way that yields the correct composition or topology. For example, in the analysis of 3D brain images, it is desirable that a construction of the cortical surface has a topology that is consistent with brain anatomy [10]. In this application and others like it, the topology flexibility is considered as a liability rather than an advantage. To address this problem, a topology-preserving geometric deformable model (TGDM) method was proposed in [11] that guarantees the topology correctness of the final segmentation results. Recently, several extensions of the original TGDM method have also been proposed in the literature [12–14].

A major remaining drawback of GDMs (both standard and topology preserving ones), however, is their computational cost. Although powerful techniques to improve the computational efficiency of GDMs have been developed (cf. [5,15,16]), the resolution of the final result is still dependent on the resolution of the computational grid used to solve the GDM partial differential equations (PDEs). In addition, the resolution of the computational grid also limits the level of details of the embedded implicit contour that the model can reliably represent. For example, the contour shown in Fig. 1(a) cannot be represented by a conventional GDM because different parts of the curve pass through the same edge, as indicated by the two arrows. With implicit contour embedding, the level set function will have the same sign at both ends of the edge, indicating that there are no contour intersections on the edge. Thus, if we apply a standard GDM (SGDM) to recover the truth contour, the result would most likely resemble Fig. 1(b), which has a totally different topology than that of the truth. The TGDM method introduced previously [11] would shift one of the two curves in order to keep a grid point between them, as shown in Fig. 1(c). Although the topology is maintained, the accuracy is adversely affected. For some applications – on very large images or with real-time requirements – relatively coarse grids may be necessary, and these kind of problems may have a significant impact on the accuracy of the active contour model.

Similar problems have been faced in the solution of general PDEs. Higher accuracy typically requires a highly refined grid, which leads to intolerably high computational cost. An elegant solution to address this dilemma is the use of *adaptive grid techniques* [17], which locally refine or redistribute grid points according to salient features of the PDE solution. Such techniques achieve high accuracy with a small increase in computational cost. There are two competing adaptive grid techniques, the *local refinement method* and the *moving grid method*. In the local refinement method, additional grid nodes are inserted when and where they are needed; in the moving grid method, a fixed number of grid points are repositioned according to an adaptivity criterion. The local refinement method has been used previously in solving level set PDEs [18,19]; and a local refinement adaptive grid method has also been proposed for geometric deformable models [20]. Another related work is the use of locally refined trianglular mesh in a 2D GDM in [21]. The multiresolution approach in [5] can also be viewed as a special case of local refinement, where the grid is uniformly refined within the narrow band region. The moving grid technique has been applied previously in the computer vision literature for adaptive image reconstruction [22,23]. However, this method has been largely overlooked by the level set community, especially in application to image segmentation with GDMs.

In our research on topology-preserving geometric deformable models (TGDMs), we have found local refinement methods to be incompatible with the digital topology principles needed to implement TGDM. This is because the refined grid no longer has a uniform structure and the conventional digital topology principles no longer apply. In contrast, the uniform reference grid maintained by a moving grid method allows the same topology preserving principle to be applied as in the uniform grid case. In this work, we adapt a particular moving grid method, the *deformation moving grid method*, which was developed by Liao et al. [24–28], to solve the

level set PDEs required in GDMs. In particular, we develop the methods necessary to compute signed distance functions, to handle topology preservation, and to adapt the grid in an image-driven fashion, which improves the results in many applications. We show that the proposed approach resolves the resolution problem identified in Fig. 1(a) by moving the grid, as shown in Fig. 1(d). [1] We also show that the resulting method has an additional advantage, that of reducing the number of vertices produced in the final model. Thus, this overall approach automatically implements a kind of mesh simplification by putting more vertices in regions of high curvature and fewer where the boundaries are relatively flat.

The paper is organized as follows. In Section 2, we give a brief overview of geometric deformable model methods, both the standard and the topology preserving ones. In Section 3, we summarize the deformation moving grid method and discuss its practical implementation, including the introduction of a new non-degeneracy constraint that ensures the validity of the generated adaptive grid. We then describe the *moving grid geometric deformable model* (MGGDM) framework in Section 4, where the focus is placed on the design of grid adaptation criteria and the design of suitable numerical methods and an efficient implementation scheme. The non-degeneracy criterion of Section 3 and the methods described in Section 4 constitute the major contribution of this paper. Experimental results on both simulated and real images are presented in Section 5 to demonstrate the advantages of using moving adaptive grid with geometric deformable models. Finally, we conclude the paper in Section 6.

In the following presentation of the method, we focus mainly on the 2D case for the sake of clarity and notation convenience. Additional comments will be added where necessary to help clarify the 3D implementation. A preliminary report of this work, which studies solely the 2D formulation, can be found in a conference proceedings [29].

2 SGDM and TGDM

In this section, we first give a brief overview of SGDM. Then we will summarize the TGDM, which was developed in [11] as a topology preserving variant of a GDM.

Let $I: \Omega \rightarrow R^+$ be a given image, where $\Omega \subset R^2$ for a 2D image and $\Omega \subset R^3$ for 3D. In GDMs, the evolving contours are first embedded as the zero level set of a signed distance function (the level set function) $\varphi(\vec{x}, t): \Omega \times R^+ \rightarrow R$, and propagate implicitly through the temporal evolution of φ.

There are various forms of GDMs proposed in the literature. We can, however, summarize their level set evolution equation in the following general form (cf., [9,30]):

$$\frac{\partial \varphi(\vec{x}, t)}{\partial t} = [F_{\text{curv}}(\vec{x}, t) + F_{\text{prop}}(\vec{x}, t)]|\nabla \varphi(\vec{x}, t)| + \vec{F}_{\text{adv}}(\vec{x}, t) \cdot \nabla \varphi(\vec{x}, t) \quad (1)$$

where F_{prop}, F_{curv}, and $\vec{F}_{\text{adv}}$ are user-designed force (or speed) terms. F_{curv}, the curvature force, controls the regularity (smoothness) of the implicit contour. F_{prop} and F_{adv} are two forms of images forces (scalar and vector respectively) that drive the contour to the desired object boundary.

The numerical solution of (1) is traditionally obtained by approximating the time derivative by a forward difference and the spatial derivatives on the right-hand side by upwind finite-

[1] We note that in all cases, the resulting curves will be approximated by line segments.

difference numerical schemes on a rectilinear computational lattice (for details see [9]), which gives:

$$\varphi(\vec{\xi}, t_{m+1}) = \varphi(\vec{\xi}, t_m) + \Delta t \Delta\varphi(\vec{\xi}, t_m), \tag{2}$$

where $\vec{\xi}$ denotes a grid point, Δt is the time step-size, and $\Delta\varphi$ represents an appropriate discrete approximation to the right-hand side of (1) at the grid point. Then, at the time step $t_{m+1} = (m + 1)\,\Delta t$, we update the value of the level set function φ at each grid point $\vec{\xi} = (\xi, \eta)$ from its previous value $\varphi(\vec{\xi}, t_m)$, until convergence or after a user specified number of time steps.

In this framework, the updating of the level set function φ is performed on fixed grid points; thus, no parameterization of the deforming contour is needed. The explicit parametric representation is computed after the evolution is completed by taking the zero level set of φ at the last time step, which requires an isocontour algorithm [11,31]. It is well known that the topology of the embedded contour can change during the evolution of the level set function φ. This means that the topology of the final contour is unpredictable, which can be an undesired drawback in some applications [10,12–14,32].

TGDM was originally introduced in [11] to provide a simple mechanism to impose a topology constraint during the evolution of the level set function. It is based on the observation that the implicit contour is homeomorphic to the boundary of the digital object delineated by the level set function on the computational lattice (the digital object simply consists of all grid nodes with a non-positive signed distance value). The topology of the implicit contour can thus be maintained by controlling the topology of the digital object. This is achieved by applying the *simple point* criterion from the theory of *digital topology* [33], and preventing the level set function from changing sign at non-simple points. Later on, the simple point criterion was generalized in [13] to allow more flexible control of the model topology. In [12,14], self-repelling forces derived from global topology constraints were introduced to further improve the performance of TGDM and make it less sensitive to contour initialization. TGDM has found important applications in brain image analysis [10].

3 Deformation Moving Grid Method

In this section, we summarize the *adaptive moving grid by deformation* method of Liao et al. [24–28] and discuss its practical implementation. The advantage of this particular moving grid method as compared with many others is that it allows a much simpler construction of the grid mapping and can be easily extended to higher dimensions. The description in this subsection mainly follows the presentation of the cited references [24–28], but is rewritten for clarity and consistency with later adaptation to the GDM framework.

3.1 Basic principles

Like other moving grid methods, this deformation method maintains a fixed reference (logical) grid, but moves the actual physical grid points according to the desired salient features to be sought. This grid reallocation can be described as a grid mapping $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}, t)$ from the reference domain Ω_r to the actual physical or image domain Ω as demonstrated in Fig. 2, where $\boldsymbol{\xi} = (\xi, \eta)$ ((ξ, η, ζ) in 3D) denotes a point in Ω_r, and $\mathbf{x} = (x, y)$ ((x, y, z) in 3D) is its image in the physical domain Ω. Typically, the reference domain Ω_r is simply a replica of the physical domain Ω itself, but covered with a fixed uniform grid $\{(\xi, \eta) \mid \xi = 0, \Delta\xi, \ldots, N_\xi \Delta\xi, \eta = 0, \Delta\eta, \ldots, N_\eta \Delta\eta\}$, where N_ξ and N_η denote the grid size, and $\Delta\xi$ and $\Delta\eta$ are the uniform grid spacing in the coordinate directions. The time variable t simply indicates that the grid can be dynamically adapted, for example, to follow the evolution of the time-dependent level set PDE.

Different moving grid methods differ in how to construct the grid mapping $\mathbf{x}(\boldsymbol{\xi}, t)$, and thus to control the adaptivity of the physical grid. The deformation moving grid method [24–28] controls the grid adaptivity by specifying the Jacobian determinant $J(\mathbf{x})$ of the grid mapping:

$$J(\mathbf{x}) \triangleq \left|\frac{\partial \mathbf{x}}{\partial \xi}\right| = 1/f(\mathbf{x}, t), \tag{3}$$

where $f(\mathbf{x}, t)$ is called the *monitor function*. Since the Jacobian determinant of the grid mapping is simply the ratio of the size of an area (volume in 3D) element in Ω and the size of its inverse image in Ω_r, (3) says that the size of a physical grid cell is inversely proportional to f. Therefore, the grid will be condensed in regions of high f values and coarsened where f is small. In addition, by restricting $f(\mathbf{x}, t)$ to be positive and finite, $J(\mathbf{x})$ will be positive over Ω, and the grid is theoretically guaranteed not to fold onto itself.

Directly solving for the grid mapping $\mathbf{x}(\boldsymbol{\xi}, t)$ from the Jacobian condition (3) is difficult and impractical. Instead, the deformation moving grid method [24–28] constructs the grid map through a simple deformation scheme. We first assume that the grid map has been computed at time t_{k-1}, and we are to find the mapping at time t_k. t_{k-1} and t_k can be two consecutive time steps in discretizing (1), or can be just two dummy time instants when new adaptive grids are generated. The latter is the case we usually encounter when applying this method to the GDM framework. The monitor function $f(\mathbf{x}, t)$ is also assumed to be known at the two time instants t_{k-1} and t_k. Typically, at $k = 0$, the grid mapping $\mathbf{x}(\boldsymbol{\xi}, t_0)$ is assumed to be the identity map, i.e., $\mathbf{x}(\boldsymbol{\xi}, t_0) = \boldsymbol{\xi}$, and the monitor function is simply a constant function $f(\mathbf{x}, t_0) = 1$.

The deformation scheme generates the grid mapping at t_k by deforming the grid at t_{k-1}. It essentially consists of three major steps:

1. Solve for a scalar potential function $P(\mathbf{x}, t_k)$ from the following Poisson equation:

$$\nabla^2 P(\mathbf{x}, t_k) = -\frac{\partial f(\mathbf{x}, \tau)}{\partial \tau} = -\frac{f(\mathbf{x}, t_k) - f(\mathbf{x}, t_{k-1})}{t_k - t_{k-1}}, \tag{4}$$

   with the Neumann boundary condition. Here, $\tau \in [t_{k-1}, t_k]$ denotes the parameterization of the deformation process, and the monitor function $f(\mathbf{x}, \tau)$ is assumed to be a linear function of τ for $\tau \in [t_{k-1}, t_k]$.

2. Compute the deformation velocity $\vec{v}(\mathbf{x}, \tau)$ by

$$\vec{v}(\mathbf{x}, \tau) = \frac{\nabla P(\mathbf{x}, \tau)}{f(\mathbf{x}, \tau)}. \tag{5}$$

3. At each grid node indexed by the reference coordinates $\boldsymbol{\xi} = (\xi, \eta)$ ((ξ, η, ζ) in 3D), solve the following initial valued ordinary differential equation (ODE) over τ by a proper numerical integration method:

$$\begin{aligned} &\frac{D}{D\tau}\mathbf{x}(\xi, \tau) = \vec{v}(\mathbf{x}(\xi, \tau), \tau), \quad t_{k-1} < \tau \le t_k \\ &\mathbf{x}(\xi, \tau = t_{k-1}) = \mathbf{x}_{t_{k-1}}(\xi). \end{aligned} \tag{8}$$

The deformation velocity $\vec{v}(\mathbf{x}, \tau)$ defined in Step 2 above guarantees that $f(\mathbf{x}, \tau) \cdot J(\mathbf{x}, \tau)$ be a constant for all τ as proved in [24,25,27,28]; especially, $J(\mathbf{x}, t_k) \cdot f(\mathbf{x}, t_k) = J(\mathbf{x}, t_{k-1}) \cdot f(\mathbf{x},$

t_{k-1}). Since $J(\mathbf{x}, 0) \cdot f(\mathbf{x}, 0) = 1$ by construction, $J(\mathbf{x}, t_k) \cdot f(\mathbf{x}, t_k) = 1$, and hence $\mathbf{x}(\boldsymbol{\xi}, t_k)$ is the desired grid transformation for every t_k.

3.2 Practical Considerations

There are several technical considerations in implementing the above algorithm that must be explained. First, in order for the Poisson equations to be solvable with the Neumann boundary condition, the integral of the source term in (4) must be zero (cf. [34]):

$$\int_{\Omega} \frac{f(\mathbf{x}, t_k) - f(\mathbf{x}, t_{k-1})}{t_k - t_{k-1}} d\mathbf{x} = 0, \text{ for all } k.$$

With the specification that $f(\mathbf{x}, t_0) = 1$, we must have

$$\int_{\Omega} [f(\mathbf{x}, t_k) - 1] d\mathbf{x} = 0, \text{ for all } k. \tag{7}$$

Therefore, any chosen monitor function must be normalized according to (7).

The second issue concerns the numerical solution of the Poisson equation in Step 1. The authors in [27] solve (4) using the successive over-relaxation (SOR) iterative method. The speed of SOR method is unsatisfactory when repeated grid generation is necessary. We previously proposed in [29] to use a spectral solver to accelerate the solution process, but later switched to the multigrid method [35]. Both the spectral and the multigrid methods are several times faster than the SOR method. No matter which solver is used, the Poisson equation is first discretized on a uniform rectilinear grid, which should be a finer grid, in general, than the uniform reference grid.

The third issue is related to the numerical integration of the grid equation, (6). Both explicit and implicit methods can be used to solve this ODE. For example, Liao et al. [27] suggested solving (6) using an implicit level set method. The advantage of using a level set method is not clear; besides, the implicit method produces the inverse mapping $\mathbf{x}^{-1}$, while the direct mapping $\mathbf{x}$ is needed for solving the original PDE. Thus, in our implementation, we apply the explicit Euler method to do the integration [35]. Note that although the deformation method guarantees the bijectivity of the grid mapping in the continuous case, numerical errors in the computation can easily ruin this property. This is especially a problem for the implicit grid-generation method of [27] since the level set method it applies for the grid generation is known to lack topology control. We thus impose an explicit non-degeneracy check during the numerical integration to ensure a proper final grid mapping, as will be detailed in the next sub-section.

3.3 Additional constraint for non-degenerate grid generation

As mentioned above, although the deformation method guarantees that in the continuous case no grid overlapping or folding can happen since the grid Jacobian is specified to be positive everywhere [24–27], in practice, discretization and numerical errors during the integration of (6) can easily ruin this property and lead to degenerate grid maps (we note that most other moving grid methods do not even guarantee the grid non-degeneracy in the continuous case). To ensure a proper grid mapping, we have to add an additional constraint to limit the grid node motion during the deformation based grid construction.

The design of this constraint is based on a nondegeneracy criterion developed in [36] and [37], which presents a sufficient condition for a deformed hexahedral grid to remain non-degenerate (similar criteria were proposed later in [38] to ensure correct topology of image

displacement fields in deformable image registration). This nondegeneracy criterion is better explained by looking at Fig. 3 and Fig. 4, which illustrate a uniform reference grid and the corresponding physical grid in 2D and 3D, respectively. The criterion basically requires that a complete set of "discrete" Jacobians of the grid mapping evaluated at every grid node should be all positive. Each discrete Jacobian corresponds to the signed area (signed volume in 3D) of a triangle (a tetrahedron in 3D) formed by the current grid point and two (three in 3D) other neighbors from the same grid cell. To make the explanation easier, let us call the *grid triangle* a triangle formed by three grid points from the same 2D grid cell (e.g., triangles *AOD* and *OAF* in Fig. 3), and the *grid tetrahedron* a tetrahedron formed by four grid points (non-coplanar in the reference domain) of a 3D grid cell (e.g., tetrahedra *AODE* and *DOBE* in Fig. 4). The non-degeneracy criterion then requires that every grid triangle (grid tetrahedron in 3D) keeps the same orientation (i.e., their signed area or volume keeps the same sign) in the physical grid as in the reference grid. Intuitively, this condition makes sure that each reference grid cell remains convex after mapping to the physical domain, and thus grid overlapping or folding can never happen. We note that in 2D the signed areas of the grid triangles correspond exactly to evaluating the Jacobian using standard finite difference operators. In the 3D case, however, extra Jacobian approximations other than the standard finite differences also need be considered.

To impose the non-degeneracy criterion, during the numerical integration of (6) that corresponds to the sequential deformation of each grid node we restrict the grid node movement to ensure that none of the grid triangles (tetrahedra in 3D) affected by it ever changes orientation. The restriction is achieved by truncating the integration time step size (denoted by $\Delta\tau$ in the following). Consider for example that the grid mapping at the central point *O* in Fig. 3 is to be updated and consider one associated grid triangle *AOD*. The orientation of *AOD* changes if and only if the point *O* moves to the other side of line *AD* after the update. Given the deformation velocity, $\mathbf{v}_O$, at point *O*, the time of intersection, denoted by t_{OAD}, for *O* to move to the line *AD* can be simply computed using two vector products and a division as follows

$$t_{OAD}=\frac{\vec{N}_{AD}\cdot(\mathbf{x}_A-\mathbf{x}_O)}{\vec{N}_{AD}\cdot\vec{v}_O}, \tag{8}$$

where $\vec{N}_{AD}$ denotes a vector normal to the line *AD*. If $t_{OAD} > 0$ (negative means *O* moves away from *AD*) and smaller than the integration time-step $\Delta\tau$, we must limit the deformation of *O* by truncating $\Delta\tau$ to be a value smaller than t_{OAD}. In 2D, each grid node (except for the boundary ones) is shared by 12 grid triangles (3 from each quadrant). The above computation and truncation need to be performed for all 12 triangles. Since only positive intersection time matters, the computational cost can be reduced by taking into account the direction of the deformation velocity at the point.

In 3D, instead of checking when a moving point hits a line, we need to check when the point moves to the plane formed by the other three points of a grid tetrahedron. The formula is the same as (8) except that $\vec{N}_{AD}$ is now the normal of the plane under consideration. In 3D, it can be checked that each interior grid point is shared by 232 grid tetrahedra (29 from each octant). To avoid grid folding, we compute 232 times-of-intersection, find the minimum $t_{\min}$ of all positive values, and then truncate $\Delta\tau$ to be smaller than $t_{\min}$ if initially $\Delta\tau \geq t_{\min}$.

With the restriction on the grid node deformation, the generated physical grid is guaranteed to have no folding or cell overlapping. The bilinearly (trilinearly in 3D) interpolated continuous grid mapping $\mathbf{x}(\xi, t)$ is thus guaranteed to be a homeomorphism from the reference domain to

the physical domain, which then enables us to study the topology of the implicit contour(s) on the reference grid.

4 Moving Grid Geometric Deformable Model

In this section, we discuss the adaptation of the deformation moving grid method to the GDM framework. We focus on the design of the monitor function, and the development of new numerical schemes to compute the signed distance function and to solve the level set PDEs on an adaptive grid. We also discuss the design of a suitable isosurface algorithm when the adaptive grid method is applied. Our discussion generally applies to both the standard GDMs without the topology constraint and the topology-preserving ones. The differences in their implementation will be explicitly stated when necessary. We start with a discussion about the solution of level set PDEs with the moving grid technique. The moving grid SGDM/TGDM algorithm and its implementation details will then be presented.

4.1 Solving level set PDEs with the moving grid method

The original level set PDE, (1), is defined on the image domain Ω (i.e., the physical domain in a moving grid method). We can rewrite it in the following compact form to emphasize its dependency on the spatial derivatives of the level set function:

$$\varphi_t = L(\varphi_x, \varphi_y, \varphi_{xx}, \varphi_{yy}, \varphi_{xy}), \tag{9}$$

where $L(\cdot)$ summarizes the right-hand side of (1) as a function of the various spatial derivatives of the unknown level set function φ.

When a moving adaptive grid is used, the computational grid in the image domain is no longer uniform and conventional finite-difference based numerical schemes for GDMs cannot be directly applied. In order to simplify the numerical solution of the differential equation, the original equation must be first transformed to the reference domain Ω_r through the grid mapping; and finite-difference numerical methods can then be applied to solve the transformed equation on the uniform reference grid.

We first consider the transformation of the spatial derivatives (we use the 2D case as an example, the derivation for the 3D case is similar and can be found in [39]). Let $\Phi(\xi, \eta, t) = \varphi(x(\xi, \eta, t), y(\xi, \eta, t), t)$. Applying the chain rule yields

$$\begin{aligned} \Phi_\xi &= \varphi_x x_\xi + \varphi_y y_\xi \\ \Phi_\eta &= \varphi_x x_\eta + \varphi_y y_\eta. \end{aligned} \tag{10}$$

From (10), we can solve for φ_x and φ_y in terms of Φ_ξ and Φ_η as

$$\begin{aligned} \varphi_x &= \frac{y_\eta}{J}\Phi_\xi - \frac{y_\xi}{J}\Phi_\eta \\ \varphi_y &= -\frac{x_\eta}{J}\Phi_\xi + \frac{x_\xi}{J}\Phi_\eta, \end{aligned} \tag{11}$$

where $J = x_\xi y_\eta - x_\eta y_\xi > 0$ is the Jacobian determinant of the grid mapping. Higher order derivatives such as φ_{xx}, φ_{yy}, and φ_{xy} can be obtained similarly [17].

The time derivative must also be transformed, since φ_t in (9) assumes that the physical coordinates (x, y) is fixed, not the reference grid (ξ, η). This transformation is found by applying the chain rule as follows

$$\left(\frac{\partial \Phi}{\partial t}\right)_{(\xi,\eta)\text{fixed}} = \left(\frac{\partial \varphi}{\partial t}\right)_{(x,y)\text{fixed}} + \nabla\varphi \cdot \mathbf{x}_t. \tag{12}$$

Substituting (12) into (9) yields

$$\Phi_t = L(\varphi_x, \varphi_y, \varphi_{xx}, \varphi_{yy}, \varphi_{xy}) + \nabla\varphi \cdot \mathbf{x}_t.$$

We can then substitute all the partial derivatives with respect to the physical coordinates by partial derivatives with respect to the reference coordinates using relationships similar to (11), and arrive at a new PDE defined on the reference grid:

$$\Phi_t = \tilde{L}(\Phi_\xi, \Phi_\eta, \Phi_{\xi\xi}, \Phi_{\eta\eta}, \Phi_{\xi\eta}) + \left(\frac{y_\eta x_t - x_\eta y_t}{J}\Phi_\xi + \frac{x_\xi y_t - y_\xi x_t}{J}\Phi_\eta\right). \tag{13}$$

This PDE can be solved using finite-difference numerical schemes (to be presented later) since the reference grid is a uniform rectilinear grid.

In the above derivations, the grid is assumed to deform continuously together with the temporal advancement of the time-dependent PDE. This pairwise solution is not necessary when solving the level set PDEs associated with GDMs, and the grid adaptation may be done sporadically. At times when the grid is held fixed, i.e., when $\mathbf{x}_t = 0$, the second term of the right-hand side of (12) and (13) disappears, and we need only solve

$$\Phi_t = \tilde{L}(\Phi_\xi, \Phi_\eta, \Phi_{\xi\xi}, \Phi_{\eta\eta}, \Phi_{\xi\eta}). \tag{14}$$

When grid adaptation is performed, we can assume that the physical evolution of the level set function $\varphi(x, y, t)$ is stopped, i.e., $\tilde{L}(\Phi_\xi, \Phi_\eta, \Phi_{\xi\xi}, \Phi_{\eta\eta}, \Phi_{\xi\eta}) = 0$, and only solve

$$\Phi_t = \frac{y_\eta x_t - x_\eta y_t}{J}\Phi_\xi + \frac{x_\xi y_t - y_\xi x_t}{J}\Phi_\eta. \tag{15}$$

That is, we separate the temporal evolution of the original PDE from the sporadic grid adaptation.

To further illustrate this idea and for clarity, we change the time variable t in (15) to a dummy variable τ, which is consistent with the previous use of τ in describing the computation of the adaptive grid:

$$\Phi_\tau = \frac{y_\eta x_\tau - x_\eta y_\tau}{J}\Phi_\xi + \frac{x_\xi y_\tau - y_\xi x_\tau}{J}\Phi_\eta. \tag{16}$$

Equation (16) updates the physical function value associated with each reference grid node to reflect the change in the physical coordinates of the grid node. Another approach is to compute the functional value at the new grid locations using some interpolation procedure after the new grid is generated. By using (16) and solving it together with the grid generation equation, (6), we can avoid artifacts that might arise when applying a simple interpolation procedure. In addition, the topology-preserving constraint can be easily applied when solving (16) since it is in the form of a simple level set PDE, as will become clear below.

4.2 Moving grid SGDM/TGDM algorithm

With a proper framework to solve the level set evolution equation on a moving adaptive grid, we can now present the new moving grid SGDM/TGDM algorithm. The TGDM version is almost identical to the SGDM one except for an additional topology check as described in Step 6 of the following algorithm. The deformation moving grid algorithm described in the previous section guarantees that the generated grid mapping has no folding or overlapping cells and thus constitutes a homeomorphism from the reference grid to the physical grid. As a result, the topology preservation constraint can be enforced on the uniform reference grid, allowing us to borrow the techniques from the original TGDM method.

In the following algorithm, we assume that the grid adaptation is performed each time the level set function is re-initialized as a signed distance function. For clarity, we present the algorithm in 2D, so that $\boldsymbol{\xi} = (\xi, \eta)$ denotes the fixed reference coordinates of a grid node and $\mathbf{x}(\boldsymbol{\xi}, t_k) = (x(\xi, \eta, t_k), y(\xi, \eta, t_k))$ denotes the actual physical coordinates of the grid node at the k-th adaptive grid. t_k is a pseudo time variable used to denote the time instant when the k-th adaptive grid is generated. We adopt the narrow band framework for the implementation of the new moving grid GDMs, with or without the topology constraint.

Algorithm 1 (Moving Grid Narrow Band Algorithm)

1. Set $k = 0$, $t_{k=0} = 0$, $f(\mathbf{x}, t_0) = 1$, and $\mathbf{x}(\xi, \eta, t_0) = (\xi, \eta)$ (the identity map). Initialize $\Phi(\xi, \eta, t_0)$ to be the signed distance function of the initial contour. Note that $\Phi(\xi, \eta, t_0)$ is the level set function evaluated at the physical location $(x(\xi, \eta, t_0), y(\xi, \eta, t_0))$.
2. If $k > 0$, re-initialize the level set function $\Phi(\xi, \eta, t_k)$ to be a signed distance function using the Fast Sweeping Method [40,41] (see Section 4.4 for details).
3. Assume that the grid deformation is parameterized by τ, where $\tau \in (t_k, t_{k+1}]$.
    a. Construct a new monitor function $f(\mathbf{x}, t_{k+1})$ from the signed distance function $\Phi(\xi, \eta, t_k)$ or the underlying image (see Section 4.3 for details).
    b. Solve the moving grid Poisson equation, (4).
    c. Compute the grid deformation velocity field $\vec{v}(\mathbf{x}, \tau)$ using (5).
    d. Integrate the moving grid ODE, (6), for each grid node from $\tau = t_k$ until $\tau = t_{k+1}$ to get the new physical grid $\mathbf{x}(\xi, \eta, t_{k+1})$.
4. Advect the level set function to follow the grid motion. This is achieved by solving (16) and using $\Phi(\xi, \eta, t_k)$, i.e., $\Phi(\mathbf{x}(\xi, \eta, t_k), t_k)$, as initialization. The solution at $\tau = t_{k+1}$, $\Phi(\xi, \eta, t_{k+1})$, gives the level set function sampled at the new physical grid $\mathbf{x}(\xi, \eta, t_{k+1})$. Note that (16) can be solved together with the integration of (6) in Step 3(d) to avoid the extra storage of the grid velocity (x_τ, y_τ).
5. Build the narrow band on the new grid $\mathbf{x}(\xi, \eta, t_{k+1})$ by finding all the reference grid nodes (ξ, η) such that $\Phi(\xi, \eta, t_{k+1})$ is within the narrow band range.

6. Transform (1) to the reference grid using the mapping $\mathbf{x}(\xi, \eta, t_{k+1})$. Advance the level set function $\Phi(\xi, \eta, t_{k+1})$ on the new grid $\mathbf{x}(\xi, \eta, t_{k+1})$ until re-initialization is required. If TGDM, the topology-preserving constraint is applied during every level set function update. Since (1) is first transformed to and then solved on the uniform reference grid (ξ, η), the simple point criterion check described in [11] for applying the topology constraint can be directly performed on the reference grid without any modification.
7. If the solution has not converged, set $k = k + 1$, and go to Step 2. Otherwise, stop.

4.3 Construction of the monitor function

We present two different schemes to define the monitor function for the moving grid GDMs. The first one makes the grid to follow the motion of the implicit contour, and the second one defines the monitor function directly based on the image to be segmented. The first scheme is more general and has been proposed in Liao et al.'s work [27]. When the second scheme is applicable, however, the adaptive grid need only be generated once beforehand, which drastically improves the efficiency of the overall moving grid GDM method. We assumed the first scheme when presenting the moving grid narrow band algorithm, Algorithm 1. The algorithm can be easily modified to accomodate the second scheme by performing the adaptive grid generation step only once at $k = 0$.

Monitor Function Design Based on the Level Set Function—When solving the level set PDE for GDMs, one only cares about the accuracy of the zero level set. Thus, a natural definition of the monitor function is [27],

$$f(\mathbf{x}) = \begin{cases} f_1 & \text{if} \quad |\varphi(\mathbf{x})| \le W; \\ f_2 & \text{if} \quad |\varphi(\mathbf{x})| > W, \end{cases} \tag{17}$$

where $f_1 > f_2$, $W > 0$ are user-defined constants that control the concentration of physical grid nodes around the zero level set. Since $\varphi(\mathbf{x})$ is the signed distance function of its zero level set, this monitor function creates fine grid cells around the zero level set and coarse cells otherwise. We note that $f(\mathbf{x})$ must be normalized according to (7).

This definition of the monitor function makes the grid nodes concentrate around the zero level set of the signed distance function. Since the zero level set keeps deforming, the grid has to be updated frequently to follow the evolution of the zero level set, as seen in Algorithm 1. Thus, this moving grid GDM scheme is not efficient in practice.

Image-based Monitor Function Design—This second monitor function design provides a more efficient moving grid GDM algorithm. In image segmentation problems, GDMs are often designed to be attracted to salient image features such as locations of high image gradient. Thus, a proper computational grid should be condensed at desired image feature locations and coarsened otherwise. It may not be easy to come up with a construction that is suitable for all applications. However, for applications that rely on object boundary information, e.g., applications where the geodesic deformable model [6] is applicable, one can define the monitor function in a way similar to the definition of the metric term in the geodesic deformable model, for example,

$$f(\mathbf{x}) = 1 + \frac{|\nabla I_\sigma|^2}{K}, \tag{18}$$

where $|\nabla I_\sigma|$ is the gradient magnitude of the image (smoothed by a Gaussian filter with standard deviation σ), and K is a normalization factor. This monitor function provides fine grid cells at regions of high image intensity gradient, and coarse grids at homogeneous regions. The advantage of this monitor function design is that the adaptive grid is only generated once, significantly reducing the computation time of the method. Again, $f(\mathbf{x})$ must be normalized according to (7).

4.4 Distance transform on adaptive grids

Efficient computation of the signed distance function is an important part of GDM implementation. In a traditional uniform grid, the fast marching method (FMM) provides a very efficient approach to build the signed distance function from a given contour during both initialization and re-initialization steps. As introduced before, FMM is an $O(N \log N)$ algorithm which solves the following Eikonal Equation

$$\begin{aligned} &|\nabla T(\mathbf{x})| = 1 \text{ in } \Omega, \\ &T = 0 \text{ on } \Gamma, \end{aligned} \tag{19}$$

where Γ is the given contour.

When the moving grid is used, the above equation must be first transformed to the uniform reference grid so that finite difference numerical methods can be used. By applying the transformation of (11), we get the transformed equation on the reference grid as (using the 2D case for illustration)

$$\frac{1}{J}\sqrt{g_{22}T_\xi^2 + g_{11}T_\eta^2 - 2g_{12}T_\xi T_\eta} = 1, \tag{20}$$

where $g_{11} = x_\xi^2 + y_\xi^2$, $g_{22} = x_\eta^2 + y_\eta^2$ $g_{12} = x_\xi x_\eta + y_\xi y_\eta$, and J is the Jacobian determinant as in (11). Unfortunately, (20) is no longer an Eikonal equation, and the FMM is no longer applicable.

There exist two methods proposed in the literature that can be used to solve the type of Hamilton-Jacobi equation appearing in (20): the *ordered-upwind method* by Sethian et al. [42] and the *fast sweeping method* by Tsai et al. [40,41]. We chose to use the fast sweeping method to compute the signed distance function T in (20). Note that the fast sweeping method requires that the Hamiltonian is strictly convex, which in our case requires that $g_{11}, g_{22} > 0$ and $g_{11}g_{22} > g_{12}^2$. It can be proved that this condition is satisfied if and only if $J > 0$, which is guaranteed in our implementation of the deformation moving grid method.

The fast sweeping method uses an upwind and monotonic Godunov flux to approximate the Hamiltonian, and solves the equation by a Gauss-Seidel-type iterative algorithm with alternating sweeping directions. The computational complexity of the algorithm is $O(N)$, where N is the number of grid points at which the distance values are to be found. The details can be found in [40,41].

4.5 Numerical schemes to solve the Level Set PDEs on adaptive grids

We now discuss the numerical schemes to approximate the transformed level set PDE on the reference grid. As in the case of a fixed uniform grid, different schemes are required to approximate the different types of force terms. Consider the general level set PDE of (1). The first term is the curvature force term. After transforming this term to the reference grid, we can approximate it using the centered finite difference scheme as in the traditional level set method, which involves the use of centered finite difference approximations to both the spatial derivatives of the level set function, Φ_ξ, Φ_η, $\Phi_{\xi\xi}$, etc., and the derivatives of the grid mapping, x_ξ, x_η, y_ξ, and y_η.

The propagation force term $F_{\text{prop}}|\nabla\Phi|$ requires entropy-satisfying numerical schemes to avoid numerical instability such as oscillations. In a previous paper [29], we suggested the use of a *local Lax-Friedrichs* (LLF) scheme [43] to approximate $\nabla\Phi$ in this term. As known in the literature, the LLF scheme can be over-diffusive and often smoothes out fine details of the implicit contour. It is also tedious to generalize to the 3D case. We therefore designed an alternative method to approximate this propagation force term. The new scheme has been shown to work well through our experiments, but its convergence properties have not been rigorously analyzed. For clarity, we present the method in 2D here; the generalization to 3D is straightforward, which can be found in [39].

We note that on the reference grid, $|\nabla\Phi|$ has the following expression:

$$|\nabla\Phi| = \frac{1}{J}\sqrt{g_{22}\Phi_\xi^2 + g_{11}\Phi_\eta^2 - 2g_{12}\Phi_\xi\Phi_\eta}, \tag{21}$$

where g_{11}, g_{22}, g_{12}, g_{21}, and J come from the coordinate transformation, and are the same as in (20). We approximate the coordinate transformation terms by centered finite difference operators. To satisfy the entropy condition and ensure stability, the key is to choose suitable approximation to the partial derivatives of Φ with respect to the reference grid coordinates. We first denote the forward, backward, and centered difference operators in the two coordinate directions by $D^{+\xi}$ ($D^{+\eta}$), $D^{-\xi}$ ($D^{-\eta}$), and $D^{0\xi}$ ($D^{0\eta}$), respectively, and construct four upwind finite difference operators by taking into consideration the sign of F_{prop}, which is similar to what is proposed in the standard level set method [9]:

$$\begin{aligned}
D'^{+\xi}\Phi &= \text{sign}^+(F_{\text{prop}})(D^{+\xi}\Phi)^- - \text{sign}^-(F_{\text{prop}})(D^{+\xi}\Phi)^+,\\
D'^{-\xi}\Phi &= \text{sign}^+(F_{\text{prop}})(D^{-\xi}\Phi)^+ - \text{sign}^-(F_{\text{prop}})(D^{-\xi}\Phi)^-,\\
D'^{+\eta}\Phi &= \text{sign}^+(F_{\text{prop}})(D^{+\eta}\Phi)^- - \text{sign}^-(F_{\text{prop}})(D^{+\eta}\Phi)^+,\\
D'^{-\eta}\Phi &= \text{sign}^+(F_{\text{prop}})(D^{-\eta}\Phi)^+ - \text{sign}^-(F_{\text{prop}})(D^{-\eta}\Phi)^-,
\end{aligned} \tag{22}$$

where sign(·) is 1 if the argument is positive and −1 otherwise, and $x^+ = \max(x, 0)$, $x^- = \min(x, 0)$.

Substituting Φ_ξ with either $D'^{+\xi}\Phi$ or $D'^{-\xi}\Phi$ and Φ_η with either $D'^{+\eta}\Phi$ or $D'^{-\eta}\Phi$ into (21) will give us four different evaluations of $|\nabla\Phi|$. We can also compute a fifth value of $|\nabla\Phi|$ by substituting into (21) Φ_ξ with $D^{0\xi}\Phi$ and Φ_η with $D^{0\eta}\Phi$. We finally take the maximum of all the five values as the value of $|\nabla\Phi|$ used in the computation of $F_{\text{prop}}|\nabla\Phi|$. This is the numerical scheme we currently applied to evaluate the propagation force term. In the 3D case, the computation of this force term involves the comparison of nine different evaluations of $|\nabla\Phi|$, as detailed in [39].

The next force term to approximate is the advection force term $\vec{F}_{\text{adv}} \cdot \nabla\Phi$. Since this force term is a linear term, a simple upwind approximation suffices. Again, consider the 2D case, and denote $\vec{F}_{\text{adv}} = (F_1, F_2)$. This term has the following form in the reference grid:

$$\vec{F}_{\text{adv}} \cdot \nabla\Phi = \frac{F_1 y_\eta - F_2 x_\eta}{J}\Phi_\xi + \frac{F_2 x_\xi - F_1 y_\xi}{J}\Phi_\eta.$$

Denote $a = (F_1 y_\eta - F_2 x_\eta)/J$ and $b = (F_2 x_\xi - F_1 y_\xi)/J$. The upwind approximation to this term can then be obtained as

$$\vec{F}_{\text{adv}} \cdot \nabla\Phi = a^- D^{-\xi}\Phi + a^+ D^{+\xi}\Phi + b^- D^{-\eta}\Phi + b^+ D^{+\eta}\Phi, \tag{23}$$

where $a^+ = \max(a, 0)$, and $a^- = \min(a, 0)$ as in (22). We use the forward difference to compute the grid derivatives in a^+ and b^+, and the backward difference to compute a^- and b^-.

The transformed level set PDE is in general much more complicated than the original due to the involvement of the grid mapping terms. Its solution is thus also more time-consuming, especially in 3D. There is, however, an important simplification that we can make, especially during early stages of the model deformation. Again, unlike a traditional PDE, what is important in the level set PDE for a GDM is actually the image forces, which drive the contour to its optimal location. As a result, we can initially ignore the non-uniformity of the physical grid, i.e., we solve the original level set PDE directly on the reference grid but with the image forces being pulled from the corresponding physical grid locations. After this simplified computation is converged, we solve the full version of the transformed level set PDE for a few more iterations in order to fine tune the location of the final contour(s). Our experiences have shown that this computational scheme does not degrade the accuracy of the overall method but largely improves the efficiency. In particular, it normally improves the computational speed to two to three times the original for both 2D and 3D cases. This more efficient scheme is used in all the experiments presented later.

4.6 Final contour extraction on an adaptive grid

The design of a proper isocontour or isosurface algorithm is a necessary step for the moving grid GDM as well, but it is actually very easy. The uniform rectilinear reference grid allows the direct adoption of the connectivity consistent isocontour algorithm [11] to extract the final contour(s) from the level set function after convergence. In particular, the isocontour algorithm is performed on the reference grid, which is followed by an additional linear interpolation step to find the physical coordinates of each contour node. Again, in the case of TGDM, since the topology constraint is enforced on the reference grid, the topology of the contour on the reference grid is guaranteed to be correct. In addition, since the grid mapping is guaranteed to be a homeomorphism, the physical contour is then homeomorphic to the reference-grid contour, and thus will also have the correct topology and have no self-intersections.

5 Results

In this section, we present several experiments to demonstrate the benefits of applying the moving grid method to GDMs. The results are mainly in 2D, but a preliminary result on 3D cortical surface reconstruction is also presented. The experiments were run on a 2.2 GHz Intel Pentium4 PC equipped with a Linux operating system.

5.1 2D experiments

Fig. 5(a) shows a phantom image comprising two circular cells. The initial contours for the deformable model are also shown as two dark curves. The image is of size 128 × 188. Using a computational grid of the same size, a TGDM with a signed pressure force and a curvature force produces the final boundary segmentation in Fig. 5(b). The topology constraint has no effect in this case since the two boundaries are well separated. The segmentation of Fig. 5(b) is used as the "ground truth" when comparing the results of later coarse grid segmentations.

We now choose a coarse computational grid of size 33 × 45 and apply both SGDM (i.e., GDM without the topology-preservation constraint) and TGDM with the same forces as before. The results of SGDM and TGDM segmentations are shown as dark curves in Fig. 5(c) and Fig. 5 (d), respectively. The "ground truth" contours are also shown in the two figures as the white curves. Obviously, without the topology constraint, the two cells are wrongly merged. Both results have large errors as compared to the truth (the largest distance from the truth contour to either of the two coarse grid results is greater than 3 pixels). Figs. 5(e)–(g) illustrate the results when applying the moving adaptive grid. Figs. 5(e) and (f) show the deformed grid at an intermediate and the final stage together with its corresponding zero level set. The accuracy is improved, as shown in Fig. 5(g), where the final contours (dark curves) coincide well with the truth (white curves). The largest distance from the truth to the moving grid result is reduced to about 0.7 pixels. We note that the final contour in Fig. 5(g) comprises 146 vertices, which is comparable to the contours in Figs. 5(c) and 5(d), which have 130 and 136 vertices, respectively. In contrast, the "ground truth" contour in Fig. 5(b) has 542 vertices. In fact, the vertex reduction is roughly proportional to the reduction in grid lines in each direction. We note that if a local refinement adaptive grid or a multiresolution GDM implementation is used, the final contour will have similar size as the ground truth (if similar accuracy is maintained) since the effective computational grid in these methods are identical to a uniform grid as far as the deformable contour is concerned. Thus, in many applications, the moving grid GDM or TGDM implementation can have a strong advantage over the local grid refinement or multiresolution level set methods by yielding much smaller final contour size.

Fig. 5(h) shows the grid produced using the image derived monitor function [cf., (18)], which is created once before the contour evolution. Fig. 5(i) shows the final contours produced by this grid. The contours have only 142 vertices but are almost indistinguishable from the truth. The image derived grid adaptation requires solving the grid deformation equations only once, which improves the overall computational efficiency by another large factor.

In terms of computational speed, the time for generating the truth contour takes about 1.3s. SGDM or TGDM running on the uniform coarse grid takes about 0.2s. When the moving grid is applied, the computation for each adaptive grid generation takes about 0.26s, which is almost equal to the time needed for the implicit contour propagation part (0.28s). Thus, the moving grid is more efficient if the grid generation only need be computed a very few number of times, as in the case of image-based grid generation.

To get a more complete quantitative evaluation, we tested the algorithms using a phantom image shown in Fig. 6(a). This image has a size of 512 × 512 pixels and consists of a hand-shaped object. We applied the geodesic active contour model on several grids of different types and sizes, and compared the accuracy and contour size of the final results, and also compared the computation time. To evaluate the accuracy of the results, we first ran SGDM with a uniform grid of size 512 ×512 and took the resulting contour as the truth; this is shown as the red curve overlaid on the image in Fig. 6(a). We then measured the errors of other results by computing the distance from each vertex of the truth contour to the other contours. All the models were initialized using a small circle inside the object, as shown by the blue curve in Fig. 6(a). When

a moving grid GDM is applied, the adaptive grid was generated only once using the monitor function definition of (18).

The results are summarized in Table I, which is categorized into four groups: uniform grid SGDM, uniform grid TGDM, moving adaptive grid SGDM (MG-SGDM), and moving adaptive grid TGDM (MG-TGDM). Three different grid sizes are used in each of the 4 groups. The columns show the grid size, the total computation time, the numbers of final contours and contour vertices, and the maximum and mean errors of each result as compared to the truth. The computation time reported in Column 3 includes the time for the grid generation when an adaptive grid is used. The errors in Columns 6 and 7 are measured in the unit of the original image pixel size. The topology correctness of the final results is indicated by the number of final contour pieces shown in Column 4. In particular, the truth contour has the topology of a circle that contains one single connected piece. But as shown in Fig. 6(d) where a uniform grid SGDM is applied, the initial single contour will get split into two disjoint parts at the place where the two middle fingers are closely adjacent to each other, which results in a contour count of two.

From Table I, it is seen that the uniform grid SGDM results all have the wrong topology, even when a fine 256 × 256 grid is used. The topology problem does not exist in the TGDM results due to the topology preservation constraint of TGDM. On the other hand, the correct topology is observed in all the adaptive grid SGDM results as well. This is due to the high grid resolution along the object boundary as achieved by the grid adaptation, which can be seen in Fig. 6(b). In terms of accuracy, Table I shows that with the same grid type a finer grid results in better accuracy, but also requires longer computation time. The more important comparison is between the uniform grid and the adaptive grid. For example, if we compare Row 8 with Row 1 or Row 11 with Row 4, we can see that the moving grid method allows the use of a much smaller grid size while achieving a much better accuracy in a comparable computation time. The resulting contour also has a smaller size when using the moving grid than using the uniform grid. If we use an even smaller grid as in Row 9 (or Row 12) of the table, the MG-SGDM (or MG-TGDM) still achieves a comparable accuracy in average as the fine uniform grid. At the same time, the computation time is significantly reduced, and the final contour is further simplified. Comparing the performance between TGDM and SGDM, it is seen that TGDM gives slightly better accuracy than SGDM on the same uniform grid due to the preserved contour topology, but the computation time for TGDM is slightly longer. For the adaptive grid cases, MG-TGDM gives the same results as MG-SGDM since the MG-SGDM already gives the correct topology thus the topology constraint has no effect in these cases.

To demonstrate the performance of the adaptive grid GDM method under the condition of image noise, we generated a noise-contaminated phantom image by adding Gaussian white noise (with zero mean and with variance equal to 30% of the maximum image intensity value) to the image in the previous experiment. The resulting noisy image is shown in Fig. 7(a). We repeated the previous experiment on this new noisy image, and the results for the TGDM cases are summarized in Table II. From Table II, it is seen that MG-TGDM with a much smaller grid size (64 × 64) still achieves comparable accuracy as the uniform grid TGDM on a much larger grid (256 × 256), although the accuracy is degraded for both as comparing to the corresponding results in Table I. It is also noticed that increasing grid size for the MG-TGDM no longer leads to accuracy improvement as significant as in the previous experiment. This result is expected since the accuracy in this case is inherently limited by the added image noise. Comparing Fig. 7(b) with Fig. 6(b), it is clear that image noise also degraded the quality of the generated adaptive grid and made it less regular. But since the final grid density is controlled by the relative magnitudes of the monitor function across the computational domain instead of the absolute values, the grid is still mostly condensed around the true object boundary. The spatial Gaussian smoothing inside (18) also helped regularize the monitor function ($\sigma = 1.0$ was used

in this experiment). On the other hand, in practice it can be expected that pre-processing a noisy image with more advanced image restoration or noise reduction filters may further improve the accuracy of TGDM (on both uniform and adaptive grids), but the investigation of which is beyond the scope of this paper.

In the last 2D experiment, we applied the moving grid GDMs to segment a real CT image of carpal bones. Fig. 8(a) shows the original image with the initial contours overlaid. The image is of size 151 × 220 pixels. We used a binary-flow GDM, which tries to separate the mean intensity of the region inside the evolving contour from the mean of the outside. On a uniform grid of the same size as that of the image, SGDM produced the result in Fig. 8(b) and TGDM gave the result of Fig. 8(c). Due to the close adjacency of the two bones, SGDM created the wrong topology even at the finest grid. We then applied the moving grid TGDM with a reference grid of size 50 × 65. Fig. 8(d) shows the final physical grid together with the final contour segmentation. Fig. 8(e) displays this segmentation on the original image (dark curves), together with the "true" contours as in Fig. 8(c) (white curves). Fig. 8(f) is a magnified view at the gap between the two bone cells. Again, the two sets of contours are almost indistinguishable (the average distance is only 0.17 pixels in the unit of the original image), but the moving grid contours have only 290 vertices while the original grid contours have 838 vertices.

5.2 Moving grid TGDM for cortical surface reconstruction

In this 3D experiment, we apply a moving grid topology-preserving geometric deformable surface model to the brain cortical surface reconstruction problem [10,32] and compare its performance against the uniform grid TGDM. We show that applying the moving grid framework to TGDM can potentially yield a better cortical surface reconstruction method in terms of better computational efficacy and simultaneous surface mesh size reduction, which is an important advantage especially for the processing of high resolution brain images of large sizes.

Details about the overall cortical surface reconstruction method can be found in [10]. In this experiment, we focus on the reconstruction of the central cortical surface only, the procedures of which can be briefly explained with the help of Fig. 9. First, a given 3D brain image is preprocessed to remove the skull, skin, and other non-brain tissues (a 2D cross-section of the image after the preprocessing is shown in Fig. 9(a)). A fuzzy segmentation method is then applied to generate a soft segmentation of the brain tissues, which produces three membership functions corresponding to the three major tissue classes: white matter, gray matter, and cerebrospinal fluid (the corresponding 2D cross-section of the gray matter (GM) segmentation is shown in Fig. 9(b)). Image forces are then derived from the membership functions and used to drive a TGDM model to find the central surface of the brain cortex as shown in Fig. 9(d). A key image force is a vector valued *gradient-vector-flow* (GVF) force [3] that is derived from the GM membership function and points to the central layer of the GM. This image force corresponds to the advection force $\vec{F}_{\text{adv}}$ in (1).

In this experiment we chose six brain images with manually picked central surface landmarks as the test data sets, but upsampled the original images by a factor of two in each coordinate direction in order to simulate high-resolution brain images. Such an upsampling step is also desirable in practice in order to improve the accuracy of cortical segmentation. The original images all have a size of 256 × 256 × 198, and after upsampling, the new images have size 512 × 512 × 396. This large image size causes a huge increase in computation time and memory usage if the uniform grid TGDM implementation is used, as will be shown later. The resulting surface mesh also has a huge number of vertices, which can cause problem for post-processing steps, such as cortical unfolding or flattening, and surface-based data processing.

After upsampling, we apply the same cortical surface reconstruction procedure on the new images until the deformable surface reconstruction step. We then apply TGDM on three different computational grids to extract the central cortical surfaces. The first one uses a fine computational grid (the same size as the upsampled image). The second one uses a coarse computational grid of the original image size (a uniform grid of size 256 × 256 × 198). The third one applies the moving grid TGDM method, where the reference grid is chosen to be the same as the coarse uniform grid (also with size 256 × 256 × 198). The adaptive grid is computed only once before the surface deformation, where the GM membership function is used as the monitor function so that the grid is condensed inside GM and coarsened otherwise. The image derived forces are computed initially from the upsampled image (i.e., on the fine uniform grid), and interpolated to the coarse uniform grid and the adaptive grid in the latter two TGDM implementations. After the central surfaces are reconstructed from all six studies using the three TGDM implementations, distances from the pre-picked landmark points to the reconstructed surfaces are then computed and used as a measurement of surface accuracy (i.e., the landmark error).

The overall average landmark error, computation time, memory usage, and resulting surface mesh size (also an average of the six brains) from using the three different grids are summarized in Table III. As we can see from Table III, the fine uniform grid provides the best accuracy, that is, the smallest average landmark error. The computation time and memory usage is greatly increased, however. The resulting surface mesh is also huge, over four times the size of the coarse grid result. The coarse uniform grid gives the fastest computation and the smallest surface mesh, but the landmark error is much worse. As further demonstrated in Fig. 10, the coarse grid surfaces differ quite largely from the fine grid (and the adaptive grid) results and have obvious large errors within narrow sulcal regions. The adaptive grid produces comparable accuracy as the fine uniform grid, as can be seen both from the landmark error in Table III and from the visual comparison in Fig. 10. The computation time is also largely reduced compared with the fine uniform grid and the resulting surface mesh size is only slightly larger than that of using the coarse uniform grid. These results clearly demonstrated the potential advantages of applying the moving grid TGDM to the cortical reconstruction problem. Note that in this experiment, the moving grid slows down the TGDM computation by a factor of 3 comparing to the uniform grid of the same size, This factor is consistent across all the 6 brains used in this experiment, which is mainly due to the increased grid point density around the target surface location as the adaptive grid is designed to achieve. It is also due to the fact that in this experiment, the initial surface (the WM boundary surface) is very close to the final surface location; hence most of the computation for the non-uniform grid TGDM is computed inside a dense grid region. This factor would become smaller if the initial surface is further away from the final surface.

6 Discussion and Conclusion

In this work, we adapted the deformation moving grid method to the framework of both standard and topology preserving geometric deformable models. We designed new numerical methods both for the implementation of the adaptive grid method and for solving level set PDEs on an adaptive grid. We also introduced a grid nondegeneracy constraint to make sure that the computed grid map has no folding or overlapping; and proposed an image based grid monitor function design, which further increases the efficiency of the overall method. As demonstrated by the experimental results, the grid adaptation increases the accuracy and efficiency of solving the level set PDEs associated with the geometric deformable models. Compared with the local refinement or multiresolution techniques, the moving grid method also provides an additional advantage of producing contours or surface meshes with fewer vertices, which can be a very significant advantage in 3D applications.

One limitation of the current method is that the computational speed is not yet satisfactory when applying the moving grid geometric deformable model in 3D. The time spent for generating the adaptive grid is almost comparable to that for solving the level set PDE. Thus the adaptive grid generation procedure need be further optimized. As another shortcoming, the deformation moving grid method that we currently apply does not have direct control over the grid quality, such as grid orthogonality and smoothness. It is known that poor grid quality — for example, the departure from orthogonality or grid skewness — can limit the accuracy that can be gained when applying the moving grid method. A recent improvement in order to address the grid skewness problem was proposed by Cao et al. [44]. This new moving grid method, however, is extremely time-consuming in computation, which prohibits its use for 3D applications. Thus, the deign of a better and faster moving grid generation algorithm still need further investigation. Fortunately, the proposed MGGDM framework separates the adaptive grid generation step from the implementation of the model deformation, and should thus allow other alternative moving adaptive method to be easily incorporated.

Last, it is worth noting that other implicit surface representation methods such as radial basis function (RBF) and algebraic models may offer the potential of improved computational efficiency than the conventional level set method [45–47], but such methods have not yet been widely used in deformable model based image segmentation. On the other hand, we expect that many existing GDM image segmentation methods may benefit from the proposed moving grid framework due to its simplicity over other adaptive grid techniques, its natural handling of both standard and topology-preserving GDMs, and its ability to produce more accurate contours or surfaces with significantly reduced number of contour or surface nodes.

Acknowledgments

This work was supported in part by NSF/ERC Grant CISST#9731748 and by NIH/NINDS Grant R01NS37747.

References

1. Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models. Intl J Comp Vision 1988;1:312–333.
2. Cohen LD. On active contour models and balloons. CVGIP: Image Understanding 1991;53:211–218.
3. Xu C, Prince JL. Snakes, shapes, and gradient vector flow. IEEE Trans Imag Proc 1998;7(3):359–369.
4. Caselles V, Catte F, Coll T, Dibos F. A geometric model for active contours in image processing. Numerische Mathematik 1993;66:1–31.
5. Malladi R, Sethian JA, Vemuri BC. Shape modeling with front propagation: A level set approach. IEEE Trans PAMI 1995;17:158–175.
6. Caselles V, Kimmel R, Sapiro G. Geodesic active contours. International Journal of Computer Vision 1997;22:61–79.
7. Yezzi A, Kichenassamy S, Olver P, Tannenbaum A. A geometric snake models for segmentation of medical imagery. IEEE Trans Med Imag 1997;16:199–209.
8. Siddiqi K, Lauziere YB, Tannenbaum A, Zucker SW. Area and length minimizing flow for shape segmentation. IEEE Trans Image Proc 1998;7:433–443.
9. Sethian, JA. Level Set Methods and Fast Marching Methods. Vol. 2. Cambridge University Press; Cambridge, UK: 1999.
10. Han X, Pham D, Tosun D, Rettmann M, Xu C, Prince JL. CRUISE: Cortical reconstruction using implicit surface evolution. NeuroImage 2004;23:997–1012. [PubMed: 15528100]
11. Han X, Xu C, Prince JL. A topology preserving level set method for geometric deformable models. IEEE Trans Patt Anal Machine Intell 2003;25:755–768.
12. Sundaramoorthi G, Yezzi A. Global regularizing flows with topology preservation for active contours and polygons. IEEE Trans Imag Proc 2007;16:803–812.

13. Ségonne F. Active contours under topology control – genus preserving level sets. Intl J Comput Vision 2008;79(2):107–117.
14. Le Guyader C, Vese LA. Self-repelling snakes for topology-preserving segmentation models. IEEE Trans Imag Proc 2008;17:767–779.
15. Adalsteinsson D, Sethian JA. A fast level set method for propagating interfaces. J Comput Phys 1995;118:269–277.
16. Peng D, Merriman B, Osher S, Zhao H, Kang M. A PDE-based fast local level set method. J Comput Phys 1999;155:410–438.
17. Knupp, P.; Steinberg, S. Fundamentals of Grid Generation. CRC Press; Boca Raton, FL: 1994.
18. Milne, B. PhD dissertation. Dept. of Math; UC Berkeley: 1995. Adaptive Level Set Methods Interfaces.
19. Sussman M, Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. An adaptive level set approach for incompressible two-phase flow. J Comput Phys 1999;148:81–124.
20. Droske, M.; Meyer, B.; Schaller, C.; Rumpf, M. An adaptive level set method for medical image segmentation. In: Insana, MF.; Leahy, RM., editors. Proc IPMI 2001. Springer Verlag; 2001. p. 416-422.LNCS 2082
21. Xu M, Thompson PM, Toga AW. An adaptive level set segmentation on a triangulated mesh. IEEE Tran Med Imag 2004;23:191–201.
22. Terzopoulos, D.; Vasilescu, M. Proc CVPR'91. Lahaina, HI: 1991. Sampling and reconstruction with adaptive meshes; p. 70-75.
23. Vasilescu, M.; Terzopoulos, D. Proc CVPR'92. Champaign, IL: 1992. Adaptive meshes and shells; p. 829-832.
24. Liao G, Pan T, Shu J. Numerical grid generator based on Moser's deformation method. Numer Meth Part Diff Eq 1994;10:21–31.
25. Bochev P, Liao G, dela Pena G. Analysis and computation of adaptive moving grids by deformation. Numer Meth Part Diff Eq 1996;12:489–506.
26. Liao, G.; de la Pena, G.; Liao, G. A deformation method for moving mesh generation. Proc. 8th Intl. Meshing Roundtable; South Lake Tahoe, CA. 1999. p. 155-162.
27. Liao G, Liu F, de la Pena G, Peng D, Osher S. Level-set-based deformation methods for adaptive grids. J Comput Phys 2000;159:103–122.
28. Liao G, Xue J. Moving meshes by the deformation method. J Comput Appl Math 2006;195:83–92.
29. Han, X.; Xu, C.; Prince, JL. A 2D moving grid geometric deformable model. Proc. of CVPR (CVPR'03); Madison, Wisconsin. June 2003; p. 153-160.
30. Xu, C.; Yezzi, A.; Prince, JL. On the relationship between parametric and geometric active contours. The 34th Asilomar Conference on Signals, Systems, and Computers; Pacific Grove, USA. 2001. p. 483-489.
31. Lorensen WE, Cline HE. Marching cubes: A high-resolution 3D surface construction algorithm. ACM Computer Graphics 1987;21(4):163–170.
32. Han, X.; Xu, C.; Prince, JL. Topology preserving geometric deformable models for brain reconstruction. In: Osher, S.; Paragios, N., editors. Geometric Level Set Methods in Imaging, Vision and Graphics. Springer-Verlag; New York: 2003.
33. Kong TY, Rosenfeld A. Digital topology: Introduction and survey. CVGIP: Image Understanding 1989;48:357–393.
34. Strang, G. Introduction to Applied Mathematics. Wellesley Cambridge Press; 1986.
35. Press, WA.; Teukolsky, SA.; Vetterling, WT.; Flannery, BP. Numerical Recipes in C. Vol. 2. Cambridge University Press; New York, NY: 1995.
36. Ivanenko, SA. Harmonic mappings. In: Thompson, JF.; Soni, BK.; Weatherill, NP., editors. Handbook of grid generation. Vol. 8. CRC Press; Boca Raton: 1999. p. 1-43.
37. Ushakova OV. Conditions of nondegeneracy of three-dimensional cells. A formula of a volume of cells. SIAM J Sci Comput 2001;23:1274–1290.
38. Karaçali B, Davatzikos C. Simulation of tissue atrophy using a topology preserving transformation model. IEEE Trans Med Imag 2006;25:649–652.

39. Han, X.; Xu, C.; Prince, JL. ECE technical report. Johns Hopkins University; Baltimore, MD: 2006. A moving grid framework for geometric deformable models. available at http://iacl.ece.jhu.edu/~xhan/MMGDM.pdf
40. Tsai, YR.; Cheng, L.; Osher, S.; Zhao, H. Technical Report UCLA-CAM-01-27. Institute for Pure and Applied Mathematics (IPAM), UCLA; 2001. Fast sweeping algorithms for a class of Hamilton-Jacobi equations.
41. Kao, CY.; Osher, S.; Tsai, YR. Technical Report UCLA-CAM-02-66. Institute for Pure and Applied Mathematics (IPAM), UCLA; 2002. Fast sweeping methods for a class of static Hamilton-Jacobi equations.
42. Sethian JA, Vladimirsky A. Ordered upwind methods for static Hamilton-Jacobi equations. Proc Natl Acad Sci 2001;98(20):11069–11074. [PubMed: 11572970]
43. Osher S, Shu C. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. J Comput Phys 1989;83:32–78.
44. Cao W, Huang W, Russell RD. A moving mesh method based on the geometric conservation law. SIAM J Sci Comput 2002;24:118–142.
45. Khoo B, Wang S, Lim K, Wang M. An extended level set method for shape and topology optimization. J Comput Phys 2007;221:395–421.
46. Xie, X.; Mirmehdi, M. Implicit active model using radial basis function interpolated level sets. Proc. 17th British Machine Vision Conf.; 2007. p. 1040-1049.
47. Oishi, T.; Takamatsu, J.; Zheng, B.; Ishikawa, R.; Ikeuchi, K. 6-DOF pose estimation from single ultrasound image using 3D IP models. Proc. IEEE Comput. Vision Patt. Recog. Workshop 2008; 2008. p. 1-8.

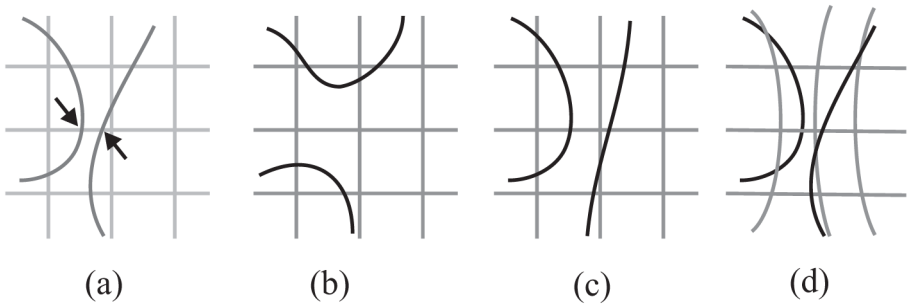


Figure 1.
Resolution problem of level set methods. (a) Contours irrepresentable due to implicit embedding; (b) GDM changes topology; (c) TGDM keeps the contours separated by grid nodes; (d) An adaptive grid correctly resolves the desired contours.

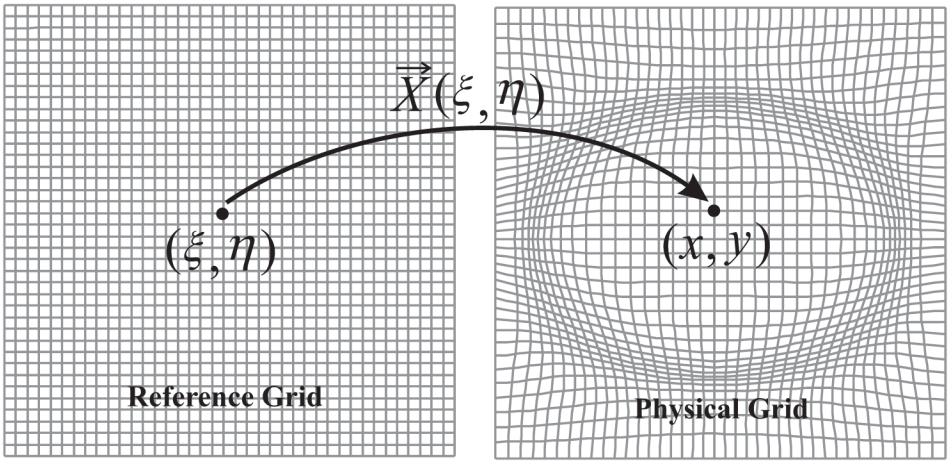


Figure 2.
Adaptive grid generation with moving grid methods: the adaptive grid in the physical domain can be described as a mapping of the uniform reference grid.

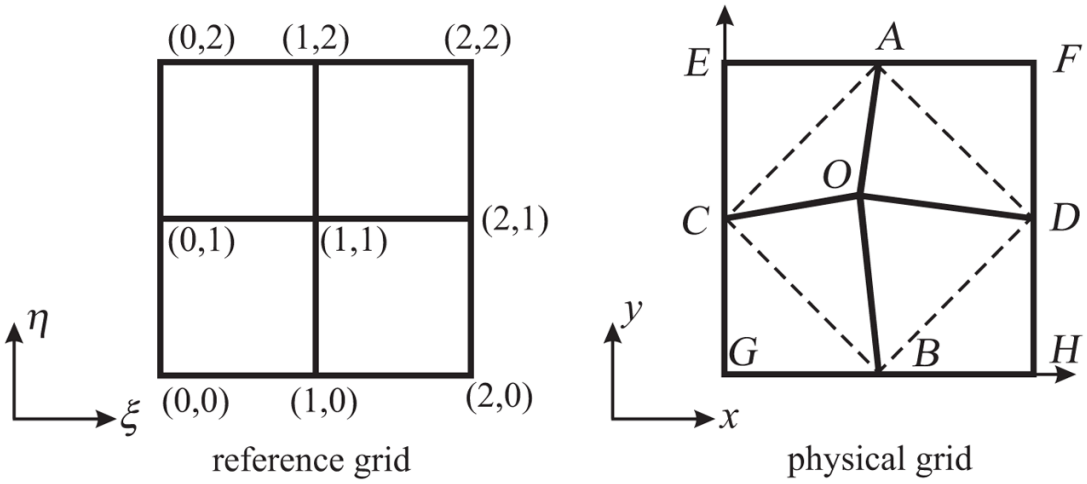


Figure 3.
2D reference and physical grid cells.

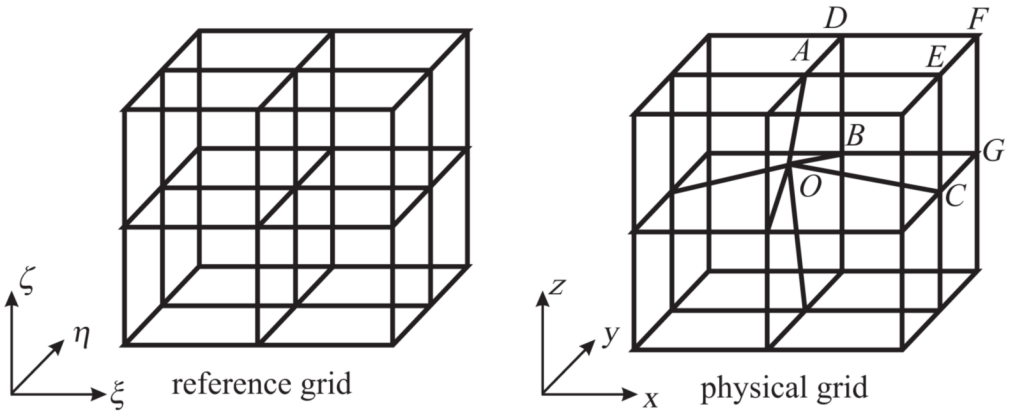


Figure 4.
3D reference and physical grid cells.

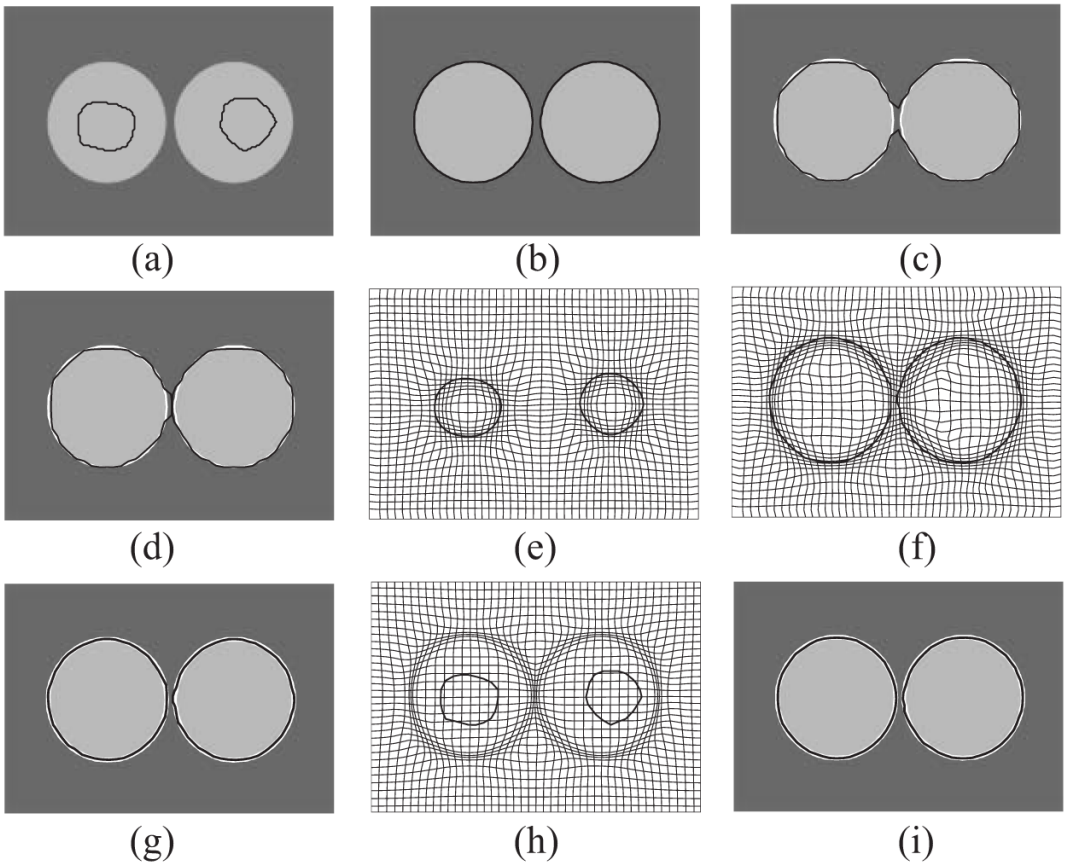


Figure 5.
Segmentation of a phantom image. See text for details.

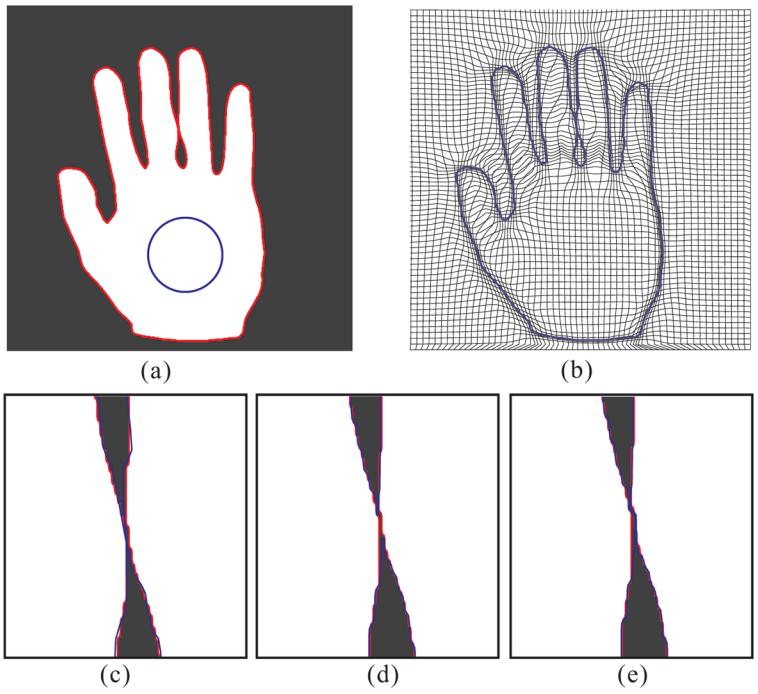


Figure 6.

(a) A phantom image with a hand-shaped object. Red contour shows the true object boundary and the blue circle is the initial contour for the GDMs. (b) The final adaptive grid of size 64×64 generated using (18). The overlaid blue curve is the final result of the adaptive grid GDM (both MG-SGDM and MG-TGDM give same results on this grid). (c) Magnified view of the result in (b) around the touching-finger area. The underlying red contour is the truth. (d),(e) Magnified views of the corresponding results of the 256 × 256 uniform grid SGDM and TGDM respectively.

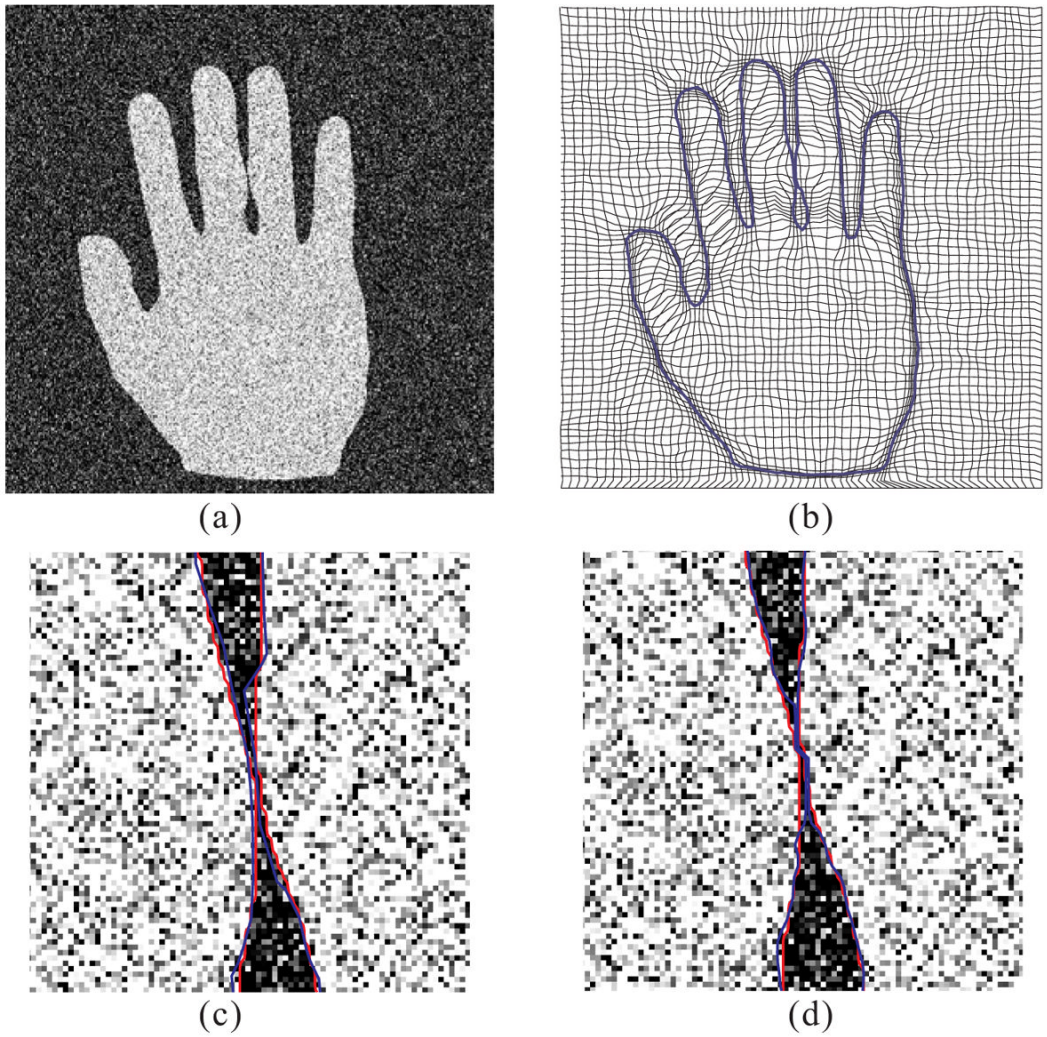


Figure 7.
(a) The hand phantom image contaminated by white Gaussian noise. (b) The final adaptive grid of size 64 × 64 generated using (18); the overlaid blue curve shows the adaptive grid TGDM result. (c) Magnified view of the result in (b). The underlying red contour indicates the truth. (d) Magnified view of the corresponding result of the 256 × 256 uniform grid TGDM.

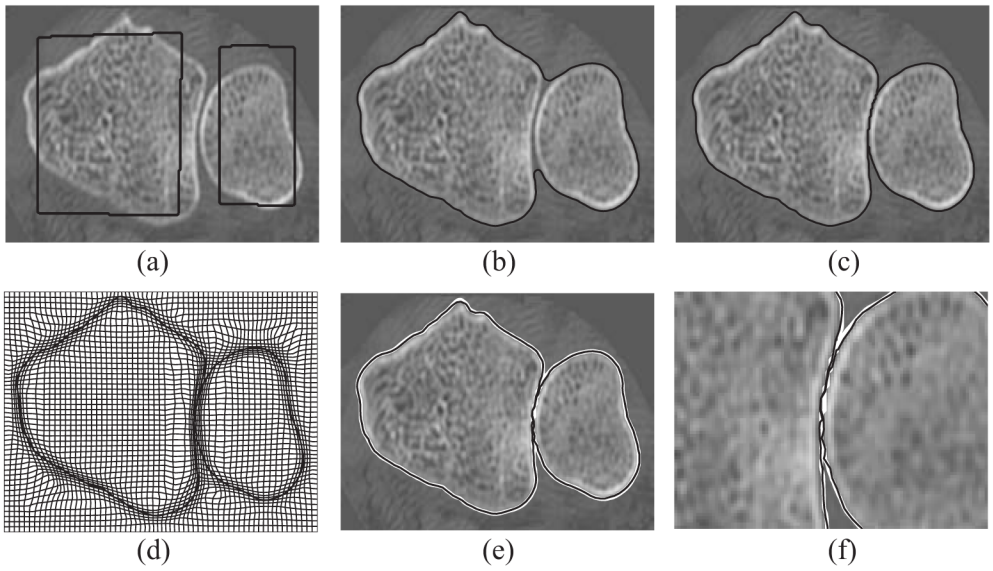


Figure 8.
Segmentation of a carpal bone CT image. See text for details.

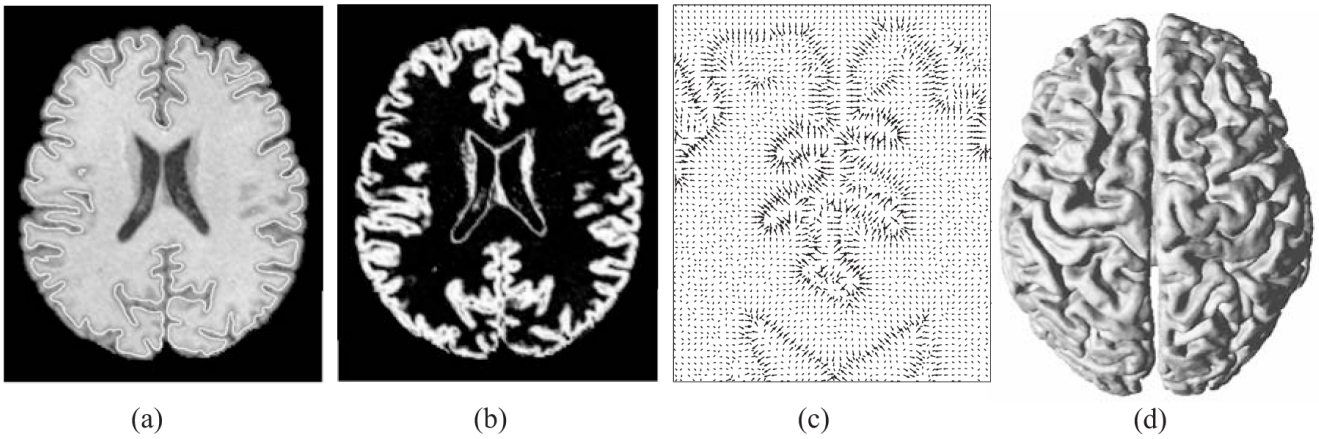
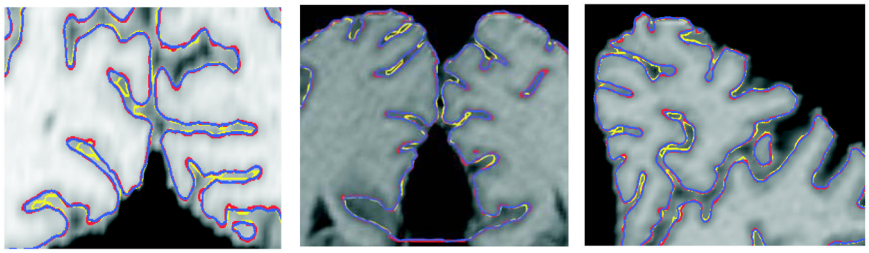


Figure 9.
(a) 2D cross-section of a 3D MR brain image (the overlaid white curve is the extracted cortical surface); (b) 2D cross-section of the GM membership function; (c) the GVF image force projected on the same 2D slice (zoomed view); and (d) the reconstructed central cortical surface using the method described in [10].



(a) (b) (c)

Figure 10.
Magnified views of the central surface reconstruction results from three different TGDM implementations. Red: TGDM with fine uniform grid; yellow: TGDM with coarse uniform grid; blue: TGDM with adaptive grid.

Table I

Comparison of SGDM and TGDM on grids of different type and size. Entries with maximum error < 2 or mean error ≤ 0.3 are highlighted in bold.

	Grid Size	Time (sec)	# Contours	# Vertices	Maximum Error	Mean Error
Uniform Grid SGDM	256 × 256	7.72	2	1344	2.13	**0.27**
	128 × 128	3.62	2	668	4.33	0.61
	64 × 64	1.47	2	330	10.99	1.50
Uniform Grid TGDM	256 × 256	9.04	1	1349	**1.28**	**0.27**
	128 × 128	4.26	1	673	2.06	0.60
	64 × 64	1.72	1	337	6.44	1.41
Adaptive Grid SGDM	256 × 256	11.66	1	1737	**0.61**	**0.06**
	128 × 128	5.12	1	894	**1.10**	**0.12**
	64 × 64	2.11	1	477	**1.87**	**0.30**
Adaptive Grid TGDM	256 × 256	13.62	1	1737	**0.61**	**0.06**
	128 × 128	5.93	1	894	**1.10**	**0.12**
	64 × 64	2.40	1	477	**1.87**	**0.30**

Table II

Comparison of uniform and adaptive grid TGDM on a noisy image.

	Grid Size	Time (sec)	# Contours	# Vertices	Maximum Error	Mean Error
Uniform Grid TGDM	256 × 256	9.04	1	1359	1.62	0.28
	128 × 128	4.26	1	675	3.53	0.77
	64 × 64	1.72	1	347	8.51	1.46
Adaptive Grid TGDM	256 × 256	13.62	1	1669	1.37	0.24
	128 × 128	5.93	1	837	1.54	0.26
	64 × 64	2.40	1	419	2.20	0.34

Table III

Performance comparison of TGDM with three different computational grids.

	Fine Uniform Grid	Coarse Uniform Grid	Moving Grid
Average Landmark Error	0.62 mm	0.94 mm	0.71 mm
Computation Time	45 minutes	4 minutes	26 minutes (14 for grid)
Memory Usage	1.2 GB	0.2 GB	0.5 GB
Mesh Size (vertices)	1,416,000	338,000	437,000