Int J Comput Vis (2009) 85: 223–236
DOI 10.1007/s11263-009-0214-4

3D Topology Preserving Flows for Viewpoint-Based Cortical Unfolding

Kelvin R. Rocha · Ganesh Sundaramoorthi · Anthony J. Yezzi · Jerry L. Prince

Received: 1 February 2008 / Accepted: 13 January 2009 / Published online: 21 February 2009

Abstract We present a variational method for unfolding of the cortex based on a user-chosen point of view as an alternative to more traditional global flattening methods, which incur more distortion around the region of interest. Our approach involves three novel contributions. The first is an energy function and its corresponding gradient flow to measure the average visibility of a region of interest of a surface with respect to a given viewpoint. The second is an additional energy function and flow designed to preserve the 3D topology of the evolving surface. The third is a method that dramatically improves the computational speed of the 3D topology preservation approach by creating a tree structure of the 3D surface and using a recursion technique. Experiments results show that the proposed approach can successfully unfold highly convoluted surfaces such as the cortex while preserving their topology during the evolution.

Keywords Visibility · Visibility maximization · Topology preservation · Cortex · Surface flattening · Surface unfolding · Active polyhedron · Area preservation · Variational method

This work was supported by the grants NSF CCR-0133736 and NIH/NINDS R01-NS-037747.

K.R. Rocha (✉) · G. Sundaramoorthi · A.J. Yezzi
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
e-mail: krocha@ece.gatech.edu

G. Sundaramoorthi
e-mail: ganeshs@ece.gatech.edu

A.J. Yezzi
e-mail: ayezzi@ece.gatech.edu

J.L. Prince
Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, USA
e-mail: prince@jhu.edu

1 Introduction

In this paper, we are interested in maximizing the visibility of a local region of a surface from a given viewpoint. A wide variety of visibility-related problems appear in many different research areas such as computer graphics, computer vision, computational geometry, and robotics, just to mention a few (Durant 1999). Visibility applications include shadow computations (Woo et al. 1990), rendering (Hertzmann and Zorin 2000), etching (Sethian and Adalsteinsson 1997), and object recognition (Pope 1994) among many others. Even though a large amount of research has been done on the visibility phenomenon, to the best of our knowledge a surface evolution technique for increasing visibility, in a local manner, with respect to a user's external viewpoint has never been proposed. This is particularly applicable to the cortex of the brain. Unfolding the cortex has received much attention (Fischl et al. 1999; Carman et al. 1995; Wandell et al. 2000) in the medical imaging community, since the cortex is highly convoluted and therefore difficult to visualize in folded regions. In this work, we construct a visibility measure and a technique for gradually increasing a surface's visibility from a given viewpoint.

Several surface flattening methods that evolve a surface for the purpose of visualization have been proposed in the past. However, traditional surface flattening techniques are global, with no dependence on the user's viewing point external to the surface itself, and most of these global techniques (with a few notable exceptions such as Hermosillo et al. 1999) do not produce an evolution of the surface itself that can be interactively halted as soon the desired level of visibility is achieved (which typically does not require full flattening of the surface). As such, due to both the global nature and the full flattening effects of traditional techniques, much more distortion is incurred than necessary, especially if the user is interested in viewing only a localized part of the surface. An example in which a surface is unfolded is the work of Hermosillo et al. (1999), which proposes area and volume preserving normalized mean curvature flows together with a tracking framework that can be used to unfold the cerebral cortex via level set methods. This work, which is a 3D extension of the work by Sapiro and Tannenbaum (2005), provides the smoothing of a closed surface without shrinkage. Recently, a more general approach for surface unfolding was proposed by Pons et al. (2004) in which an application-specific normal motion is used to evolve a cortical surface while a tangential motion is used to preserve its area.

It is important that our surface visibility technique preserve the topology of the surface, e.g., for cortex unfolding this is crucial since the cortex is known to have a simple topology. Since the topology is not automatically preserved during our visibility maximization evolution, we design and add a 3D topology-preserving term into the evolution.

A number of researchers have endeavored to incorporate various topology preservation constraints into their evolution models for the purpose of segmentation. Some authors such as Han et al. (2001, 2003) have proposed discrete representation dependent constraints that kick in at the moment and at the location where a topology change is about to occur in order to enforce the original topology. Others, such as Unal et al. (2005), Slabaugh and Unal (2005) have directly added continuous evolution forces that increase toward infinity as the contour or surface configuration approaches a change in topology. Sundaramoorthi and Yezzi (2005) recently introduced a variational method for topology preservation in active contours based on knot energies (O'Hara 1991; Abrams et al. 2003).

For many segmentation applications, the manner in which the topology constraints are introduced is often unimportant since only the final configuration of the contour matters. Here, however, we consider an application of cortical unfolding in which the evolution itself is important to the end user who will typically wish to stop the unfolding process at any given time to obtain the desired level of unfolding. Therefore, the nature of the topology preservation should go hand-in-hand with the desired unfolding evolution and not yield undesirable transient geometric configurations that are often common when using mere topology enforcement.

The extension of the global knot-energy based topology regularizers proposed in Sundaramoorthi and Yezzi (2005) to three dimensions is conceptually straight-forward but mathematically and computationally much more involved than the original 2D formulation. However, our effort seems to have been well justified since these types of topology preserving energies are ideally suited to our cortical unfolding application. The resulting evolution forces, on their own, induce an unfolding effect that tends to drive the initial cortical surface towards a final spherical configuration that globally minimizes most knot energies (see Abrams et al. 2003 for the case of curves). This renders a very natural and visually pleasing global unfolding effect. Since knot-energy based topology preservation forces already produce a global unfolding effect on their own, they combine very naturally with our visibility-based flows to maintain a constant topology without introducing undesirable artifacts into the evolution.

Our goal is a viewpoint dependent unfolding of the cortex in which the user is able to select an area of interest on the cortical surface for visualization. By focusing the unfolding on a region of interest with respect to a chosen viewpoint, distortion effects may be significantly reduced compared with global flattening techniques (Hermosillo et al. 1999; Angenet et al. 1999; Pons et al. 2004) commonly used in brain mapping. To accomplish this, we introduce a novel energy functional and gradient flow to measure and improve the average visibility of the selected region. Without topology preservation, however, this flow is not useful for the purpose of cortical visualization and unfolding.

The remainder of this article is organized as follows. In Sect. 2 we review the two dimensional knot-energy based topology preservation method introduced in Sundaramoorthi and Yezzi (2005). In Sect. 3 we outline the extension of this method to three dimensions. In Sect. 4 we propose a method that uses a tree structure of the faces of a triangulated surface and a recursion technique to significantly improve the computational speed of the 3D topology preservation method presented in Sect. 3. In Sect. 5 we present our viewpoint-based visibility energy functional and its corresponding gradient flow to which this 3D topology preservation method will be applied. It is important to note, however, that without the topology constraint, the visibility-based flow often undergoes intermediate topology changes during the resulting cortical unfolding process. As such, while the visibility energy provides the driving force behind our flow, the topology forces are indispensable to this application. Finally, in Sect. 6 we show simulations on both synthetically created surfaces as well as cortical surfaces extracted from real data.

2 Background on Topology Preservation

In many active surfaces applications it is very important that the topology of the object does not change during the evolution. For instance, when the cortex of the brain is being segmented it is necessary to keep its topology during the evolution because the cortex is homeomorphic to a two-dimensional sphere (Han et al. 2001, 2003).

Several topology preservation methods have been proposed in the past. Han et al. (2001, 2003), in their work on cortical segmentation, presented a technique to prevent topology changes when the active contour evolution is implemented via level sets methods. In this work, changes in topology at grid points are detected by deriving a condition based on the configuration of the level set function in a small neighborhood of the grid points. This method has the disadvantage of being highly dependent on the grid spacing used in the level set function. In addition, when this method is used the resulting motion may be abrupt and look unnatural. Segonne (2008) designs an extension to Hans et al., based on the same principles, that produces more geometrically accurate segmentation.

Unal et al. (2005) proposed a novel approach for topology preservation for active polygons. In this work, it is assumed that the polygon consists of a uniform charge distributed along its perimeter. Each vertex is then moved in the direction of the electrostatic force, which is computed numerically. Even though this method may prevent some topology changes, it does not prevent two adjacent sides from touching. Moreover, the flow is unstable as the number of vertices increases and the length of the segments decreases (Unal et al. 2005).

Sundaramoorthi and Yezzi (2005) have proposed a robust topology preservation technique in which an special geometric flow is added to the original image-based curve evolution to avoid intersections. This geometric flow, which is derived from the minimization of an energy based on electrostatic principles, affects significantly the original evolution only when the contour is close to a change in topology. Unlike a curvature regularizer, when the regularizer proposed in Sundaramoorthi and Yezzi (2005) is applied to a point the resulting force depends globally on all other points of the curve. This technique, which is based on the work in O'Hara (1991), Abrams et al. (2003), has the advantage over the one proposed in Han et al. (2001, 2003) of changing the original evolution in a gradual manner. Moreover, it is not restricted to level sets and can be used on any active contour implementation.

Other variational approaches for topology preservation are found in the work by Shi and Karl (2004), which only favors the repulsion of different connected components of the evolving curves, and in the work of Alexandrov and Santosa (2005) and the recent work of Le Guyader and Vese (2007), both of which are designed specifically for level set methods (Osher and Sethian 1988).

Since the 3D topology preservation method we are proposing in this paper is an extension of the work in Sundaramoorthi and Yezzi (2005), in the rest of this section we present a quick review of this work.

2.1 Differentiable Contour Case

Let $C \in \mathbb{R}^2$ be a twice-differentiable contour of length L and let $E_{2D,R}$ be the energy of an uniform charge distributed along C defined by

$$E_{2D,R}(C) = \frac{1}{2}\int\int_{C\times C}\left(\frac{1}{\|\mathbf{C}(s)-\mathbf{C}(\hat{s})\|} - \frac{1}{d_C(s,\hat{s})}\right)d\hat{s}ds, \qquad (1)$$

where $d_C(s,\hat{s})$, the geodesic distance along the curve C from point $\mathbf{C}(s)$ to point $\mathbf{C}(\hat{s})$, is used to eliminate the infinite component of the first term, thereby making the energy finite. However the gradient of this energy has the property of still becoming infinitely large whenever the curve becomes close to self-intersection.

Using the Calculus of Variations, it is shown in Sundaramoorthi and Yezzi (2005) that the gradient of (1) is given by

$$\mathbf{R}_{2D}(s) = \lim_{\varepsilon\to 0^+}\left[\int_{B_C(\varepsilon,s)}\frac{\mathbf{C}(s)-\mathbf{C}(\hat{s})}{\|\mathbf{C}(s)-\mathbf{C}(\hat{s})\|^3}\cdot\mathbf{N}(s)d\hat{s} + \int_{B_C(\varepsilon,s)}\frac{d\hat{s}}{\|\mathbf{C}(s)-\mathbf{C}(\hat{s})\|}\kappa(s) - \ln\left(\frac{L}{2\varepsilon}\right)\kappa(s)\right]\mathbf{N}(s), \qquad (2)$$

where $B_C(\varepsilon,s) = \{\mathbf{C}(\hat{s}) : d_C(\hat{s},s) > \varepsilon\}$ represents the set of all points in C except for those within a small neighborhood of $\mathbf{C}(s)$, and $\kappa(s)$ and $\mathbf{N}(s)$ represent the curvature and the inward normal of C at the point $\mathbf{C}(s)$, respectively. The first term in (2) can be regarded as the projection of the electric vector field of the charge distribution at the point $\mathbf{C}(s)$ onto the inward normal $\mathbf{N}$. On the other hand, the second term can be regarded as the electrostatic potential of the charge distribution at the point $\mathbf{C}(s)$.

Now, let us suppose that C is evolved according to the image based flow $\mathbf{C}_{t,\text{original}}(s)$ that is uniformly bounded. Sundaramoorthi and Yezzi (2005) show that if the flow $\mathbf{R}_{2D}(s)$ in (2) is added to $\mathbf{C}_{t,\text{original}}(s)$, then the topology of C will be preserved during the evolution. That is, the resulting flow

$$\mathbf{C}_{t,\text{new}}(s) = \mathbf{C}_{t,\text{original}}(s) + \mu_R\mathbf{R}_{2D}(s), \qquad (3)$$

where t is the artificial time variable and μ_R is a positive constant, preserves the topology of C. Moreover, since (3) is a geometric flow, this method of topology preservation is suitable for both parametric particle-based and level set implementations.

2.2 Polygon Case

Let P be a polygon with N edges C_i for $i \in \{1,\dots,N\}$, each one of length $|C_i|$ and going from vertex $\mathbf{v}_i$ to vertex $\mathbf{v}_{i+1}$, both in $\mathbb{R}^2$. In addition, consider the electrostatic energy

$$E_{2D,R}(P) = 2\sum_i (|C_i|\ln|C_i| - |C_i|) + \frac{1}{2}\sum_{i\neq j}\iint_{C_i\times C_j} \frac{d\hat{s}ds}{\|\mathbf{C}_i(s) - \mathbf{C}_j(\hat{s})\|}, \qquad (4)$$

where the first term results from taking just the finite part of the integral $\iint_{C_i\times C_j}\frac{d\hat{s}ds}{\|\mathbf{C}(s)-\mathbf{C}(\hat{s})\|}$ and discarding the infinite component. Like the energy for the differentiable contour case (1), this energy only becomes infinitely large when the polygon approaches a topology change.

Let $\mathbf{R}_{2D,k}(t)$be the gradient descend flow of (4) for vertex $\mathbf{v}_k$ at time t that is computed by the procedure outlined in Sundaramoorthi and Yezzi (2005) and let $(d\mathbf{v}_k/dt)_{\text{original}}$ be the original image based vertex flow. Sundaramoorthi and Yezzi (2005) show that the new vertex flow

$$\left(\frac{d\mathbf{v}_k}{dt}\right)_{\text{new}} = \left(\frac{d\mathbf{v}_k}{dt}\right)_{\text{original}} + \mu_R R_{2D,k}(t) \qquad (5)$$

has the property of preserving the topology of P during the evolution.

3 Topology Preservation in 3D

We now present the extension of the work in Sundaramoorthi and Yezzi (2005) to both active surfaces and active polyhedrons, making more emphasis on the latter since we will apply this in the next section.

3.1 Differentiable Surface Case

Let $S : [0,1]\times[0,1]\to\mathbb{R}^3$ be a parameterization of a differentiable, closed, compact and orientable surface, then the natural 3D extension of (1) is

$$E_{3D,R}(S) = \iint_{S\times S}\left[\frac{1}{\|\mathbf{S}(u,v)-\mathbf{S}(\hat{u},\hat{v})\|^2} - \frac{1}{d_S^2((u,v),(\hat{u},\hat{v}))}\right]d\mathbf{S}d\mathbf{S}, \qquad (6)$$

where $d_S((u,v),(\hat{u},\hat{v}))$ is the geodesic distance along the surface S from point $\mathbf{S}(u,v)$ to point $\mathbf{S}(\hat{u},\hat{v})$. However, unlike the case of curves, the second term is not straightforward to compute numerically nor is the variation easy to compute. Therefore, we consider the cut-off energy

$$E_{3D,R}(S) = \iint_{S\times B_\varepsilon}\frac{1}{\|\mathbf{S}(u,v)-\mathbf{S}(\hat{u},\hat{v})\|^2}d\mathbf{S}d\mathbf{S}, \qquad (7)$$

where $B_\varepsilon = \{(\hat{u},\hat{v})\in[0,1]^2 : |u-\hat{u}|\geq\varepsilon \vee |v-\hat{v}|\geq\varepsilon\}$ represents the set of all the points of S except for a small neighborhood around the point $\mathbf{S}(u,v)$.

A formal computation using the Calculus of Variations shows that the limit of the gradient of (7) converges and that it is equal to

$$\mathbf{R}_{3D}(u,v) = \lim_{\varepsilon\to 0^+}\left\{\iint_{B_\varepsilon(S)}\left[\frac{\mathbf{S}(u,v)-\mathbf{S}(\hat{u},\hat{v})}{\|\mathbf{S}(u,v)-\mathbf{S}(\hat{u},\hat{v})\|^4}\cdot\mathbf{N}(u,v) + \frac{H(u,v)}{\|\mathbf{S}(u,v)-\mathbf{S}(\hat{u},\hat{v})\|^2}\right]d\mathbf{S}(\hat{u},\hat{v})\right\}\mathbf{N}(u,v), \qquad (8)$$

where $H(u,v)$ and $\mathbf{N}(u,v)$ represent the mean curvature and the inward normal of S at $\mathbf{S}(u,v)$. We believe that $\mathbf{R}_{3D}$ becomes infinite as the surface approaches self-intersection and points in a direction opposite to self-intersection, as in the case of curves. We offer experimental evidence in Sect. 6.

3.2 Triangulated Surface Case

Topology preservation methods can also be applied to active polyhedrons, that is, a polyhedral surface whose vertices evolve to minimize some energy functional. In this sense, Slabaugh and Unal (2005) have proposed a 3D extension of the work in Unal et al. (2005) by adding an electric force to each vertex flow. This force is computed by creating an electric field that goes to infinity as a vertex moves towards the surface. Unfortunately, this method does not guarantee topology preservation between non-adjacent triangular faces and becomes unstable as the triangular mesh becomes finer. This is especially true for our novel cortical unfolding application that we present in the next section, which needs topology preservation to work properly. Therefore, we decided to choose a direct energy-based approach based on Sundaramoorthi and Yezzi (2005), which although slower, provides a more powerful topology preservation factor.

Let S be a triangulated surface with N faces S_i for $i\in\{1,\ldots,N\}$. Let also $\mathbf{v}_a$, $\mathbf{v}_b$, and $\mathbf{v}_c$ be the vertices of S_i, ordered counterclockwise, as shown in Fig. 1. Now consider the new energy

$$E_{3D,R} = \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} E_{3D,R_{i,j}}, \qquad (9)$$

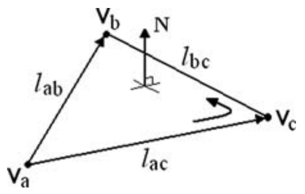

Fig. 1 Sample triangle S_i for a triangular mesh

where $E_{3D,R_{i,j}}$ represents the electrostatic energy between the faces S_i and S_j. More specifically, $E_{3D,R_{i,j}}$ is defined by

$$E_{3D,R_{i,j}} = \iint_{S_i \times S_j} \frac{1}{\|\mathbf{S}_i - \mathbf{S}_j\|^2} dS_i dS_j. \tag{10}$$

Accordingly, the gradient of the proposed energy in (9) becomes infinitely large when any two faces become infinitely close.

Taking the derivative of $E_{3D,R}$ with respect to the vertex $\mathbf{v}_a$ of S_i gives us

$$\frac{\partial E_{3D,R}}{\partial \mathbf{v}_a} = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{\partial E_{3D,R_{i,j}}}{\partial \mathbf{v}_a}. \tag{11}$$

Furthermore, if we use the parameterization

$$\mathbf{S}_i(u, v) = \mathbf{v}_a + u(\mathbf{v}_b - \mathbf{v}_a) + v(\mathbf{v}_c - \mathbf{v}_a), \tag{12}$$

for $u \in [0, 1]$ and $v \in [0, 1-u]$, then it can be shown that the derivative of $E_{3D,R_{i,j}}$ with respect to $\mathbf{v}_a$ in (11) when S_i and S_j are non-adjacent becomes

$$\begin{aligned}\frac{\partial E_{3D,R_{i,j}}}{\partial \mathbf{v}_a} &= -8A_iA_j \int_0^1\int_0^{1-u}\int_0^1\int_0^{1-\hat{u}} (1-u-v)\mathbf{F}_{i,j}\, d\hat{v}d\hat{u}dvdu \\ &\quad + 4A_j \int_0^1\int_0^{1-u}\int_0^1\int_0^{1-\hat{u}} \frac{1}{\|\mathbf{S}_i(u,v) - \mathbf{S}_j(\hat{u},\hat{v})\|^2} d\hat{v}d\hat{u}dvdu \frac{\partial A_i}{\partial \mathbf{v}_a}, \quad \text{where}\end{aligned} \tag{13}$$

$$A_i = \frac{1}{2}\|(\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)\| \tag{14}$$

is the area of S_i and the force vector $\mathbf{F}_{i,j}$ is given by

$$\mathbf{F}_{i,j} = (\mathbf{S}_i(u,v) - \mathbf{S}_j(\hat{u},\hat{v}))/\|\mathbf{S}_i(u,v) - \mathbf{S}_j(\hat{u},\hat{v})\|^4. \tag{15}$$

If S_i and S_j are adjacent, then the derivative of $E_{3D,R_{i,j}}$ with respect to $\mathbf{v}_a$ is different from (13). Without loss of generality, let us assume that S_i and S_j share at least the vertex $\mathbf{v}_a$ in Fig. 1. It can be verified that

$$\begin{aligned}\frac{\partial E_{3D,R_{i,j}}}{\partial \mathbf{v}_a} &= -8A_iA_j \int_0^1\int_0^{1-u}\int_0^1\int_0^{1-\hat{u}} (\hat{u}+\hat{v}-u-v)\mathbf{F}_{i,j}\, d\hat{v}d\hat{u}dvdu \\ &\quad + 4A_j \int_0^1\int_0^{1-u}\int_0^1\int_0^{1-\hat{u}} \frac{1}{\|\mathbf{S}_i(u,v) - \mathbf{S}_j(\hat{u},\hat{v})\|^2} d\hat{v}d\hat{u}dvdu \frac{\partial A_i}{\partial \mathbf{v}_a} \\ &\quad + 4A_i \int_0^1\int_0^{1-u}\int_0^1\int_0^{1-\hat{u}} \frac{1}{\|\mathbf{S}_i(u,v) - \mathbf{S}_j(\hat{u},\hat{v})\|^2} d\hat{v}d\hat{u}dvdu \frac{\partial A_j}{\partial \mathbf{v}_a}, \quad \text{for } \mathbf{F}_{i,j} \text{ defined as in (15).}\end{aligned} \tag{16}$$

Although the computation of (16) implies the numerical solution of quadruple integrals, we can reduce the number of computations by solving it explicitly just when the two faces S_i and S_j are close enough, that is, when they are within a certain thresholded distance from each other, which is when it matters the most. On the other hand, when the faces are not considered to be close enough we can then use their centroids, which we called $\bar{\mathbf{S}}_i$ and $\bar{\mathbf{S}}_j$, in (10). The result is the much simpler estimate

$$\frac{\partial E_{3D,R_{i,j}}}{\partial \mathbf{v}_a} = -\frac{2}{3}A_iA_j \frac{\bar{\mathbf{S}}_i - \bar{\mathbf{S}}_j}{\|\bar{\mathbf{S}}_i - \bar{\mathbf{S}}_j\|^4} + \frac{A_j}{\|\bar{\mathbf{S}}_i - \bar{\mathbf{S}}_j\|^2}\frac{\partial A_i}{\partial \mathbf{v}_a}. \tag{17}$$

Again, if we add the vertex motion as defined in (11), to a surface-based evolution then the topology of the surface will be preserved during the evolution.

4 Recursive Computational Implementation for Topology Preservation

The topology preservation method presented in the previous section is very robust, but at the same time is computationally intensive as it involves a significant amount of computations for each possible pair of triangles of the evolving surface. That is, for each vertex, we must compute the force contribution from each triangle of the entire surface mesh. Even if we use the simplified expression in (17) to estimate the topology preservation forces when triangles are far apart, the resulting number of computations would still be large, especially considering that 3D triangulated surfaces usually have thousands of triangles.

We could increase the speed of the computation of the topology preservation forces of a triangle with respect to the rest of the surface by grouping triangles that are far away and then using (17) to estimate the topology preservation forces with respect to the group of triangles as a single unit. A natural way of doing this grouping is by creating a tree data structure, where each node of the tree has a unique identification number, a centroid, an area, a list of neighbor nodes, a list of the identification number of its child nodes, and the identification number of its parent. We can then use a recursion technique to approximate the topology preservation forces for each triangle of the evolving surface.

One way we can create this tree data structure is by grouping neighbor nodes (i.e., nodes that share at least one vertex) two at a time, prioritizing those neighbor nodes that are closer to each other. Every time we merge two nodes we then create a new parent node whose area is the total area of its child nodes and whose centroid is the weighted average of the centroids of its child nodes. A more detailed explanation of the merging process is described below

Merging Algorithm

[STEP 1] Set all triangles as nodes, assign them a unique node identification number, and store the corresponding centroids and areas in each node. Tag all the nodes as UNMERGED. This set of nodes form the bottommost level of the tree.

[STEP 2] For each node tagged as UNMERGED, find the UNMERGED nodes that are neighbors and store their node identification numbers in a list of neighbors contained in the node.

[STEP 3] Compute the distance between each node and each one of its neighbors and put the information in a heap sorted by the distances.

[STEP 4] Grab the pair of neighbor nodes from the top of the current heap, that is, the two closest neighbors. If any of the nodes is tagged as MERGED, then go to [STEP 4], otherwise tag them as MERGED and merge them by creating a parent node whose area is the sum of the two node areas and whose centroid is the weighted average of the two centroids. Create a list of child nodes in the parent node and store the parent identification number in each of the child nodes. Tag the parent node as UNMERGED. If the end of the current heap has been reached go to [STEP 5], otherwise go to [STEP 4].

[STEP 5] Create a new heap sorted by distances with the elements of the previous heap that have one node tagged as UNMERGED.

[STEP 6] Grab the pair of neighbor nodes from the top of new heap and add the node tagged as UNMERGED to the list of child nodes of the parent node of the node tagged as MERGED. Tag the UNMERGED node as MERGED and update the area, the centroid, and the list of child nodes of the parent node. Store the number identification of the parent node in the child node. If all the nodes in the heap has been tagged as MERGED go to [STEP 7], otherwise go to [STEP 6].

[STEP 7] Stop if the number of UNMERGED nodes is equal to one; else go to [STEP 2].

Storing the parent identification number in its child nodes is very useful when finding the neighbor nodes in [STEP 2]. That is, if we want to find neighbors of a given parent node we just need to find the identification number of the parent nodes of the neighbor nodes of the child nodes of the given parent node.

Figure 2(a) depicts a tree structure for a surface containing nine triangles. As can be seen, in the bottommost level of the tree we have the original nine triangles of the surface. After going from [STEP 1] to [STEP 7] we have the first level of new parent nodes (*nodes* 10 to 13). We can see that even though most of the parent nodes have only two child nodes, it is still possible to have more child nodes. That is the case of *node* 11, which has three child nodes (*nodes* 3 to 5). The reason for this is that even though we want to have two child nodes per parent node, this is not always possible due to the structure of the triangulation of the surface. After going from [STEP 1] to [STEP 7] another time we have only two new parent nodes (*nodes* 14 and 15). Finally, after going from [STEP 1] to [STEP 7] for the last time, the algorithm produces only one new parent node, *node* 16, which is called the root node. It comprises all the triangles of the surface.

We can exploit the tree structure for the estimation of the topology preservation forces by using recursion as follows. Say for instance that we want to compute the topology preservation forces for the vertices of a particular triangle with respect to a node that does not contain the triangle and that has two or more child nodes and many more nodes below it. One thing we can do is to estimate the topology preservation forces by using (17) and the centroids and areas of both the triangle and the parent node. We can do exactly the same thing with each one of the child nodes. We can then compare the results to determine whether or not more accuracy is needed. If the average relative error of the result obtained using only the parent node with respect to the sum of the results obtained by using each child node is less than or equal to some chosen value, then we can say that the approximation is good enough and, as a result, we can assume that the sum the topology preservation forces with respect to the child nodes is a good approximation of the

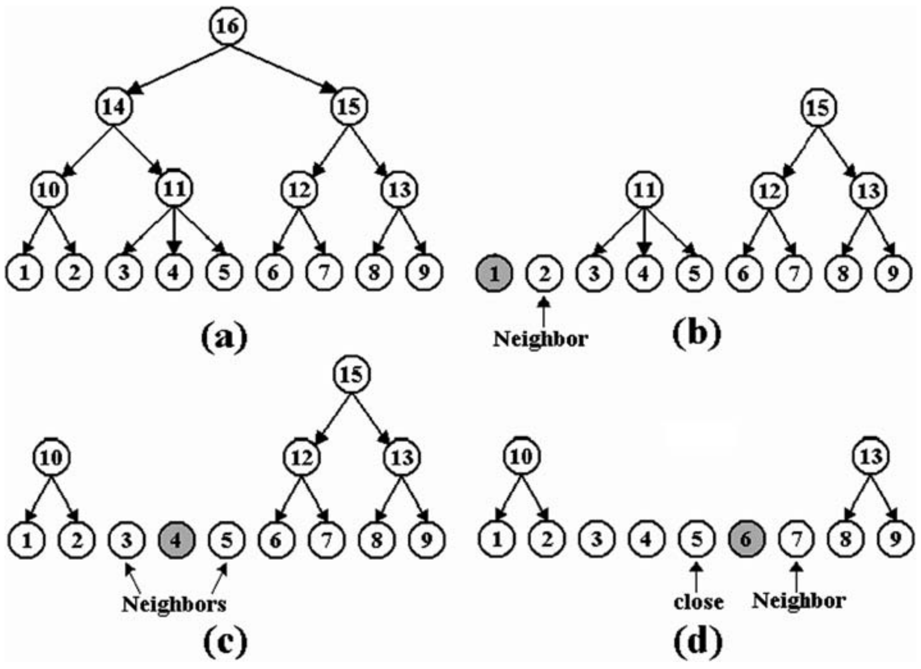

Fig. 2 Changes in the tree structure. (**a**) Original tree structure. (**b**)–(**c**) Resulting tree structures when computing the topology preservation forces with respect the *nodes* 1, 4, and 6, respectively

topology preserving forces with respect to all the triangles below the parent node. If instead, the average relative error is greater than the chosen value, then we have to repeat the process again with each one of the child nodes, where this time they would be consider as parent nodes of some other child nodes.

This recursion technique gives us a very powerful resource as we are estimating topology preservation forces with respect to a group of triangles at a time instead of a triangle at a time. Moreover, it automatically decides when more accuracy is needed.

We can also increase the accuracy of the computation of the topology preservation forces by explicitly computing them when they matter the most, that is, when two triangles are very close or when and they are neighbors. For those cases we would use (13) and (16). The resulting algorithm, which is fast and at the same time accurate enough, is describes in more detail below.

Fast Algorithm to Compute Topology Preserving Forces

For each triangle of the surface do the following:

[STEP 1] Tag all nodes in the tree as UNVISITED, except for the selected triangle, which is in the bottommost level of the tree.

[STEP 2] For each one of the nodes in the bottommost level of the tree that are within a thresholded distance from or are neighbors of the selected triangle compute the accurate topology forces for its vertices by using (13) and (16), respectively. Tag those nodes as VISITED.

[STEP 3] For each one of the nodes tagged as VISITED follow their paths to the root node, the topmost node, and tag as VISITED all the parent nodes in the paths.

[STEP 4] Select the UNVISITED node with the highest identification number and compute the topology forces for the vertices of the selected triangle with respect to this node and all the nodes below it by using recursion and the estimation formula in (17). Add the results to the overall topology forces for the vertices of the selected triangle. Tag all these nodes as VISITED.

[STEP 5] Stop if all nodes are tagged as VISITED; else go to [STEP 4].

Figure 2 shows several examples of how this algorithm works. In Fig. 2(b) it is shown the resulting tree when computing the topology preservation forces of the surface with respect to *node* 1, which is shown in gray. In this case we just need to compute the topology preservation forces accurately once, for *node* 2, which is the only neighbor node that *node* 1 has. We then have to use recursion to compute the topology preservation forces starting with *node* 15. Of course, in the case of *node* 11, it makes more sense just to compute the topology preservation forces with respect to each of the *nodes* 3, 4, and 5 by using (17). Figs. 2(c) and 2(d) show the resulting tree structures when *node* 4 and *node* 6 are selected. As can be seen, different tree structures can result. However, they can be easily obtained from Fig. 2(a) by deleting the nodes that are in the paths from the neighbor nodes of the selected node to the root node (*node* 16) and from the nodes that are close enough to the selected node to the root node.

As it is going to be seen Sect. 6, the improvement in speed by using a tree structure and a recursion technique can be enormous.

5 Application to Cortical Unfolding

In this section we present a novel viewpoint-based visibility energy as the basis for a class of flows with appropriate geometric constraints that can be used to maximize the visibility of a surface with respect to a fixed external viewpoint. The proposed energy—which was previously presented in Rocha et al. (2007) and is described in more detail here—cannot work by itself, as it requires topology preservation.

The main purpose of the method we propose here is to allow one to inspect a surface interactively from different viewpoints and perform unfolding locally with a continuous surface evolution that depends upon the user's current external viewpoint rather than global unfolding or flattening of the entire surface. This approach has the advantage of producing less distortion in the specific area of interest as well as allowing the user to interactively determine the exact level of unfolding desired. To achieve this goal, we propose a viewpoint-dependent visibility energy from which we derive a gradient surface evolution with appropriate geometric constraints. The result is a continuous unfolding of the surface with respect to the user's external viewpoint which allows the user to view the deeper self-occluded structures of the surface. This result can have several applications in the area of medical imaging. For instance, it can be used in human brain mapping to unfold a specific part of the cerebral cortex while introducing little distortion or to visually explore and validate the deep sulcul structures extracted by cortical surface segmentation algorithms. In addition, it can have a number of applications in computer graphics. For example, it can be used for the visualization of complicated surfaces and for visual inspection of texture mappings onto complex geometries.

For the sake of simplicity and better insight into the 3D case, we begin by describing our framework for visibility maximization in 2D (i.e. planar contours viewed from an external point by someone positioned in the same plane). We then extend the framework in a straight-forward manner to the 3D case.

5.1 New Visibility Energy Functional for Viewpoint-Based Unfolding: The Flux Model

The problem of maximizing the visibility of a contour with respect to a fixed viewpoint can be thought as the problem of maximizing the flux of light that is being absorbed by a contour due to a light source at a given viewpoint. In Fig. 3 the contour C is absorbing the light coming from the viewpoint **P**, which is acting as the source of light. The part of C that goes from point a to point b and contains the points c, d, e, f, and g is absorbing all the rays coming from **P**. From now on, we will refer to this region as the *region of interest of the contour*. In the figure we have used a natural criteria for choosing this region by finding the two points of C that are intersected by the extremities of the viewing region and then selecting the portion of C that passes through these two points. This procedure can be done automatically, but it is out of the scope of this work.

In the proposed flux model the objective is to evolve the region of interest of the contour in such a way that the average flux it receives is maximized. In order to do this, we first need a way to quantify the amount of flux received at any point in the contour. For instance, in Fig. 3 both the points g and f receive light from **P**. However, we might argue that points in the neighborhood of point g are receiving more light than points in the neighborhood of point f. This occurs because the rays arriving at point g are almost perpendicular to the contour, whereas the rays arriving at point f are almost tangent to the contour. We define the flux at any point in the contour as the Euclidean dot product between the unit ray that is coming from the viewpoint and the unit inward normal of the contour at the given point. Accordingly, the value of the flux at any point will always be between -1 and 1. This approach provides a physical interpretation of how illuminated any point in the contour is. If a point is illuminated, that is, if it is visible from the viewpoint, a positive flux indicates the degree of perpendicularity of the incoming ray. For instance, in Fig. 3 the flux at point g is greater than the one at point c because the ray of light is more perpendicular to the contour at the former than it is at the latter. On the other hand, if a point is not receiving any light because the ray is being blocked by the contour, a positive flux indicates how perpendicular the ray is going to come in if the part of the contour that is blocking the ray moves away. This is the

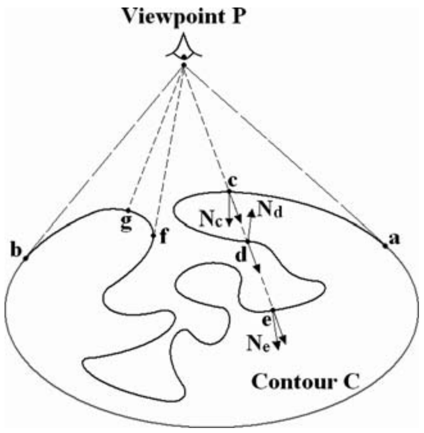

Fig. 3 Visibility of a contour C with respect to a viewpoint **P**

case of point e in Fig. 3, because even though the point does not receive any light, it has a positive flux. When a point has a negative flux, like point d in the figure, it means that the point is not receiving any light. The more negative the flux is, the more the point will have to move to receive light.

The average flux, which we call $E_{2D}(C)$, can be computed by integrating the flux at all the points in the region of interest of the given contour and then dividing by the total length. That is,

$$E_{2D}(C) = \frac{1}{L}\int_0^L \frac{\mathbf{C}(s) - \mathbf{P}}{\|\mathbf{C}(s) - \mathbf{P}\|} \cdot \mathbf{N}(s) ds, \tag{18}$$

where L is the length of the region of interest of the contour C. From now on, we will use the terms "average flux" and "average visibility" interchangeably. We need to point out a very important feature of the energy in (18). If we evolve the contour according to the gradient ascent along this energy, then the points where the flux is negative would be forced to move in such a way that the flux they receive increases. By doing so, these points make visible other points that already have a positive flux, but are not visible from the viewpoint. This generates the unfolding motion that is desired. Since the flux at any point can be at most equal to 1, then $E_{2D}(C)$ also has a maximum of 1. This would occur when the viewing surface coincides with a circular arc with the viewpoint $\mathbf{P}$ as its center. In this case, the rays coming from the viewpoint would have the same direction as the unit inward normal at each point and, consequently, the flux at every point in the region of interest of the contour would be equal to 1. Moreover, in this case all points would have the same flux. Therefore, maximizing the average visibility will, by itself, promote an equal distribution of light along the region of interest of the contour.

Using the Calculus of Variations it is not difficult to show that the gradient ascent for the energy in (18) is

$$\mathbf{C}_t(s) = \frac{1}{L}\left(\kappa(s)E_{2D}(C) + \frac{1}{\|\mathbf{C}(s) - \mathbf{P}\|}\right)\mathbf{N}(s), \tag{19}$$

where $\kappa(s)$ is the curvature of C at the point $\mathbf{C}$(s). This gradient shows that the problem is well-posed and that we can use our model to maximize the visibility of piece of contour *only* if the initial average visibility with respect to $\mathbf{P}$ is positive (otherwise a backwards heat flow results).

Now we construct the average visibility measure for a polygon (illustrated in Fig. 4). As can be seen in the figure, the region of interest of the polygon comprises N edges and goes from vertex $\mathbf{v}_1$ to vertex $\mathbf{v}_{N+1}$. By applying (18) we obtain that the average visibility of the region of interest of the polygon, $E_{2D,P}$, is

$$E_{2D,P} = \frac{1}{L_T}\sum_{i=1}^{N} E_{2D,i}, \tag{20}$$

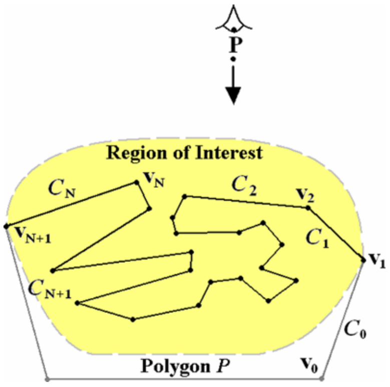

Fig. 4 Visibility of a polygon with respect to a viewpoint **P**

for

$$E_{2D,i} = \frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{|C_i|} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\mathbf{v}_i - \mathbf{P}) \cdot \ln\left(\frac{|C_i| + \|\mathbf{v}_{i+1} - \mathbf{P}\| + \phi_i}{\|\mathbf{v}_i - \mathbf{P}\| + \phi_i}\right), \tag{21}$$

where $L_T = |C_1| + \cdots + |C_1|$ is the total length of the region of interest of the polygon, $\phi_i = 1/|C_i|(\mathbf{v}_{i+1} - \mathbf{v}_i) \cdot (\mathbf{v}_i - \mathbf{P})$, and $E_{2D,i}$ is the total flux received by the edge C_i.

The motion of the vertices to maximize the average visibility in (21) can be computed by taking the partial derivatives of $E_{2D,P}$ with respect to each of the vertices and taking into consideration that the edges C_k and C_{k-1}, as well as the energies $E_{2D,k}$ and $E_{2D,k-1}$, are the only expressions in (21) that depend on the vertex $\mathbf{v}_k$ for $k = \{1, \dots, N+1\}$. Thus we get

$$\frac{d\mathbf{v}_k}{dt} = \mu_{2D}\frac{\partial E_{2D,P}}{\partial \mathbf{v}_k}, \tag{22}$$

for

$$\frac{\partial E_{2D,P}}{\partial \mathbf{v}_k} = -\frac{E_{2D,P}}{L_T}\left(\frac{\mathbf{v}_k - \mathbf{v}_{k+1}}{l_k} + \frac{\mathbf{v}_k - \mathbf{v}_{k-1}}{l_{k-1}}\right) + \frac{1}{L_T}\left(\frac{\partial E_{2D,k}}{\partial \mathbf{v}_k} + \frac{\partial E_{2D,k-1}}{\partial \mathbf{v}_k}\right), \tag{23}$$

where μ_{2D} is a positive constant, and t denotes an artificial time variable. Since the visible region of the polygon goes from $\mathbf{v}_1$ to $\mathbf{v}_{N+1}$, then the value of the partial derivative of $E_{2D,k-1}$ with respect to $\mathbf{v}_k$ is 0 for $k = 1$. Similarly, the value of the partial derivative of $E_{2D,k}$ with respect to $\mathbf{v}_k$ is also 0 for $k = N + 1$.

When the vertices of the region of interest of a given polygon are evolved according to (23) so that the average visibility is increased, we have to make sure that the topology of the polygon does not change throughout the evolution. In other words, we have to restrain the edges from intersecting with each other. To do this, we add the gradient

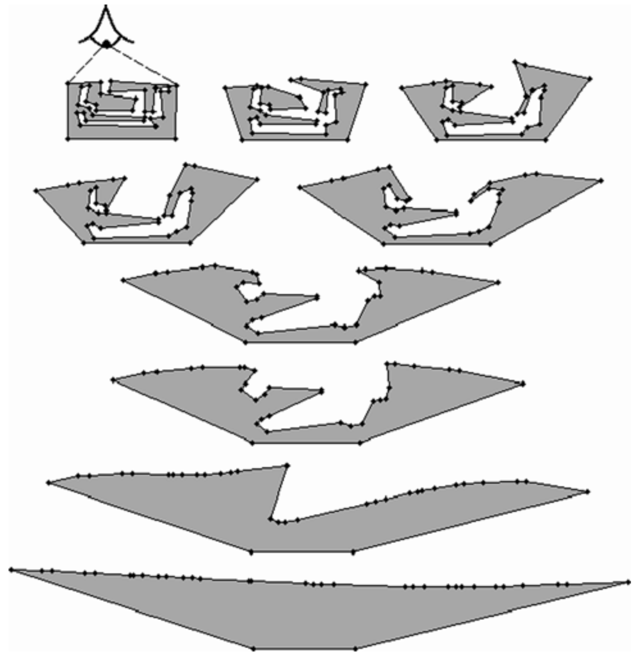
Fig. 5 Visibility maximization evolution for a highly convoluted polygon

descend of the electrostatic energy (4) to the result in (23), just like it is done in (3).

In order to have an average visibility equal to 1, the region of interest of the contour of a polygon has to have a visibility equal to 1 at every point of every one of its edges. It is easy to see that this only occurs when all the edges collapse to a single point. This is exactly what would occur if we evolve the vertices of the polygon according to (23). Of course, this result is undesirable. To overcome this issue we need to maximize the average visibility and, at the same time, maintain the length of the edges constant. This can be done by considering only the component of the gradient in (23) that does not change the length of the edge. The procedure to compute the constrained gradient is very similar to the one used in Witkin and Baraff (1997), Cantarella et al. (2004), Iben et al. (2006). Details of this procedure will be discussed in Sect. 5.4, when we show how to preserve the area of an evolving surface.

Figure 5 shows the proposed visibility maximization evolution when the initialization is a convoluted 34-edge polygon. Initially the polygon has an average visibility of just 0.17. As the proposed algorithm was applied, the polygon unfolded to make visible sections that were previously not visible. At the end of the simulation the average visibility was almost equal to one, and the total length of the region of interest was virtually preserved.

The importance of length preservation can be inferred from the results shown in Fig. 6. In this case the proposed approach was applied to the same polygon as the one in Fig. 5, but this time without enforcing length constraints.

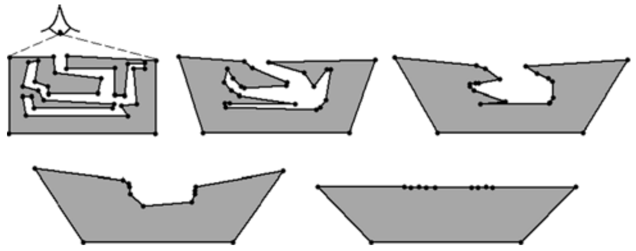
Fig. 6 Visibility maximization without length preservation

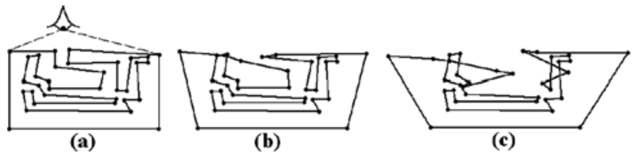

Fig. 7 Visibility maximization without topology preservation. (**a**) Initial polygon. (**b**) Polygon after 10 iterations. (**c**) Polygon after 100 iterations

The result is the region of interest of the contour shrinks itself conveniently to increase the average visibility. Indeed, the length of the visible region decreased from 723.70 pixels to just 205.41 pixels as the number of vertexes decreased from 34 to 13 and the average visibility increased from 0.17 to almost 1. As a result, even though the visibility was maximized, we did not get the unfolding effect that allows the occluded sections in the visible part of the contour to be seen.

Figure 7 shows the necessity of topology preservation. In this simulation we applied the proposed algorithm, without the topology preserving forces, to the highly convoluted polygon of Fig. 5. As can be seen, the edges intersect each other after a few iterations. The changes in topology were even larger as the number of iterations increased.

5.2 Differentiable Surface Case

The proposed method of maximizing the visibility of a contour with respect to a viewpoint can be easily extended to 3D. That is, we can use the same approach to maximize the visibility of a surface with respect to a viewpoint.

Let S be a differentiable surface and let S_v be a selected region of interest on S that the user would like to unfold for visualization. Consider the energy

$$E_{3\mathrm{D}}(S_v) = \frac{1}{A_{S_v}} \iint_{S_v} \frac{\mathbf{S}-\mathbf{P}}{\|\mathbf{S}-\mathbf{P}\|} \cdot \mathbf{N}\, dS, \tag{24}$$

where A_{S_v} is the area of S_v, $\mathbf{S} \in \mathbb{R}^3$ is a point in the surface, and $\mathbf{N} \in \mathbb{R}^3$ is the unit inward normal at $\mathbf{S}$. This energy represents the average visibility S_v with respect to $\mathbf{P}$.

If we evolve S_v according to the gradient ascent of (24), then the points where the flux is negative would be forced to move in such a way that the flux they receive increases. By doing so, these points would make visible other points that already have a positive flux, but are not visible from **P**. This would generate the unfolding motion that we are looking for.

Using the Calculus of Variations, it can be shown that a maximizing gradient flow for (24) is given by

$$\mathbf{S}_t = \frac{1}{A_{S_v}}\left(H E_{3\mathrm{D}}(S_v) + \frac{1}{\|\mathbf{S}-\mathbf{P}\|}\right)\mathbf{N}, \tag{25}$$

where H represents the mean curvature of S_v at the point **S**. Like in the 2D case, this result gives us the main formulation in case we want to implement the proposed method for differentiable contours. In addition, it provides us with a mathematical criterion that tells us which part of the surface we can flatten. Specifically, this stability condition is that we must choose the portion S_v of the surface to have a positive initial average visibility with respect to **P**. This may be done by enlarging or reducing the initial region S_v until this condition is satisfied. As the surface unfolds, the average visibility of S_v would increase, thereby allowing a user to select a smaller subset of S_v at later stages.

5.3 Triangulated Surface Case

Let S be now a triangulated mesh and let S_v be a section of S with N triangles and a positive average visibility with respect to the viewpoint **P**. By applying (24) to S_v we get that the average visibility of S_v, $E_{3\mathrm{D},P}$, is

$$E_{3\mathrm{D},P}(S) = \frac{1}{A_{S_v}}\sum_{i=1}^{N} E_{3\mathrm{D},i}, \tag{26}$$

where $A_{S_v} = A_{S_1} + \cdots + A_{S_N}$ is the total area of S_v and $E_{3\mathrm{D},i}$ represents the total flux being received by the triangular face S_i of area A_i. Accordingly, $E_{3\mathrm{D},i}$ is

$$E_{3\mathrm{D},i} = \iint_{S_i} \frac{\mathbf{S}_i - \mathbf{P}}{\|\mathbf{S}_i - \mathbf{P}\|}\cdot \mathbf{N}\,dS_i, \tag{27}$$

where the inward normal **N** (Fig. 1) is given by

$$\mathbf{N} = -\frac{(\mathbf{v}_b - \mathbf{v}_a)\times(\mathbf{v}_c - \mathbf{v}_a)}{\|(\mathbf{v}_b - \mathbf{v}_a)\times(\mathbf{v}_c - \mathbf{v}_a)\|}. \tag{28}$$

If we use the parameterization in (12), then it can be shown that (27) becomes

$$E_{3\mathrm{D},i} = \frac{\alpha}{l_{ac}}\int_0^1 \ln\left(\frac{l_{ac}^2(1-u)+\beta+\gamma l_{ac}}{\beta+\delta l_{ac}}\right)du, \tag{29}$$

where l_{ac} is the length from vertex $\mathbf{v}_a$ to vertex $\mathbf{v}_c$,

$$\alpha = -(\mathbf{v}_a - \mathbf{P})\cdot(\mathbf{v}_a\times\mathbf{v}_b + \mathbf{v}_b\times\mathbf{v}_c + \mathbf{v}_c\times\mathbf{v}_a), \tag{30}$$

$$\beta = (\mathbf{v}_c - \mathbf{v}_a)\cdot((1-u)\mathbf{v}_a + u\mathbf{v}_b - \mathbf{P}), \tag{31}$$

$$\gamma = \|u\mathbf{v}_b + (1-u)\mathbf{v}_c - \mathbf{P}\|, \tag{32}$$

and

$$\delta = \|(1-u)\mathbf{v}_a + u\mathbf{v}_b - \mathbf{P}\|. \tag{33}$$

Taking the derivative of $E_{3\mathrm{D,i}}$ with respect to the vertex $\mathbf{v}_a$ we get

$$\begin{aligned}\frac{\partial E_{3\mathrm{D},i}}{\partial \mathbf{v}_a} &= \frac{\alpha}{l_{ac}}\int_0^1\left[\frac{(2(1-u)+\gamma/l_{ac})(\mathbf{v}_a-\mathbf{v}_c)+\partial\beta/\partial\mathbf{v}_a}{l_{ac}^2(1-u)+\beta+\gamma l_{ac}}\right.\\ &\qquad\left.-\frac{\partial\beta/\partial\mathbf{v}_a + l_{ac}\partial\delta/\partial\mathbf{v}_a + \delta/l_{ac}(\mathbf{v}_a-\mathbf{v}_c)}{\beta+\delta l_{ac}}\right]du\\ &\quad + E_{3\mathrm{D},i}/\alpha(\partial\alpha/\partial\mathbf{v}_a - \alpha/l_{ac}^2(\mathbf{v}_a-\mathbf{v}_c)),\end{aligned} \tag{34}$$

where the partial derivatives of α, β, and δ with respect to vertex $\mathbf{v}_a$ can be obtained from (30), (31), and (33), respectively. On the other hand, if we take the derivative of (26) with respect to $\mathbf{v}_a$ we get

$$\frac{\partial E_{3\mathrm{D},P}(S)}{\partial \mathbf{v}_a} = \frac{1}{A_{S_v}}\sum_{i=0}^{N-1}\left(\frac{\partial E_{3\mathrm{D},i}}{\partial \mathbf{v}_a} - E_{3\mathrm{D},P}(S)\frac{\partial A_i}{\partial \mathbf{v}_a}\right). \tag{35}$$

Using these results together with those of the previous section we have that the vertex motion that maximizes the average visibility of S_v (26), while preserving its topology, is given by

$$\frac{d\mathbf{v}_a}{dt} = \mu_{3\mathrm{D}}\frac{\partial E_{3\mathrm{D},P}}{\partial \mathbf{v}_a} + \mu_R\frac{\partial E_{3\mathrm{D},R}}{\partial \mathbf{v}_a}, \tag{36}$$

where $\mu_{3\mathrm{D}}$ and μ_R are positive constants and the second term is the topology preserving term computed by using (11).

5.4 Area Preservation

In order to have an average visibility equal to 1, the region of interest S_v in the triangulated mesh S has to have a visibility equal to 1 at every point of every one of its faces. Similar to the 2D case, it is easy to see that this only occurs when all the faces collapse to a single point. To overcome this problem we need to maximize the average visibility and, at the same time, maintain the area of each one of the triangular faces of S_v constant. This can be done by applying a similar technique such as the one employed in Witkin and Baraff (1997), Cantarella et al. (2004), Iben et al. (2006) in which only the component of the gradient that does not change the area is used to evolve the vertices. This procedure is described below.

Let $\Psi = \{\mathbf{v}_1,\ldots,\mathbf{v}_M\}$ be the ordered set of the M 3D vertices comprising S_v. In order to maintain a constant area for each of the triangular faces S_i in S_v during the evolution

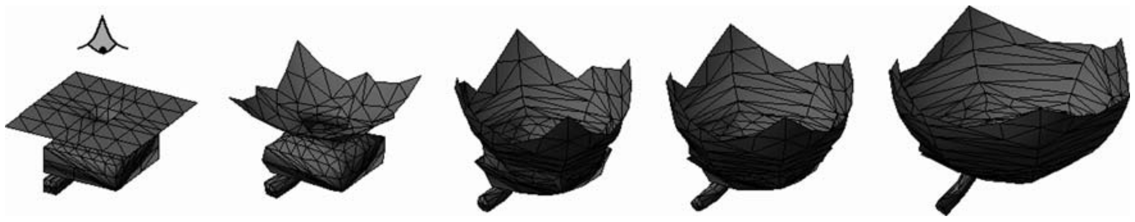

Fig. 8 Unfolding of a synthetic triangulated surface using the proposed visibility maximization approach

we need to satisfy the following N constraints

$$\frac{1}{2}\|(\mathbf{v}_{i,b}-\mathbf{v}_{i,a})\times(\mathbf{v}_{i,c}-\mathbf{v}_{i,a})\|^2 - A_i^2 = 0, \tag{37}$$

where $i \in \{1,\dots,N\}$ and the vectors $\mathbf{v}_{i,a}$, $\mathbf{v}_{i,b}$, and $\mathbf{v}_{i,c}$ represent the corresponding vertices of S_i (see Fig. 1) in Ψ. Using Lagrange multipliers one can obtain

$$V_{t,c} = V_{t,u} - J^{\mathrm{T}}\mathbf{I}, \tag{38}$$

where $V_{t,u}$ is the vector of unconstrained gradient flow obtained by applying (36) to each vertex of S_v, J is the Jacobian matrix of (37), $V_{t,c} = \frac{d}{dt}[\mathbf{v}_1^{\mathrm{T}} \ \dots \ \mathbf{v}_M^{\mathrm{T}}]^{\mathrm{T}}$ is the vector of constrained gradient flow, and the vector $\mathbf{I}$ is the minimum norm solution to the system

$$JJ^{\mathrm{T}}\mathbf{I} = JV_{t,u}. \tag{39}$$

Since the matrix JJ^{T} is symmetric positive definite, then $\mathbf{I}$ can be quickly computed using the conjugate gradient method. Moreover, since J is sparse, the matrix multiplication JJ^{T} can be computed and stored efficiently.

6 Simulation Results

Figure 8 depicts the evolution of a 3D synthetic surface when the visibility is maximized and, at the same time, the topology is preserved. The initial visibility of this surface was 0.2, whereas by the end of the simulation it was very close to 1.

Topology preservation plays a very important role when maximizing surface visibility for highly convoluted surfaces. This is the case of the evolutions shown in Fig. 9, where two regions of a cortex are evolved over 150 iterations so that the visibility with respect to a viewpoint located just in front of the regions is maximized. Each of these regions consisted of about 1,500 triangles. Using the method of topology preservation as described in Sect. 3 took about 20 minutes per iteration on a 2:53 GHz computer running Windows. However, using the fast algorithm described in Sect. 4 reduced the computational time to just 5 second per iterations, which is really promising.

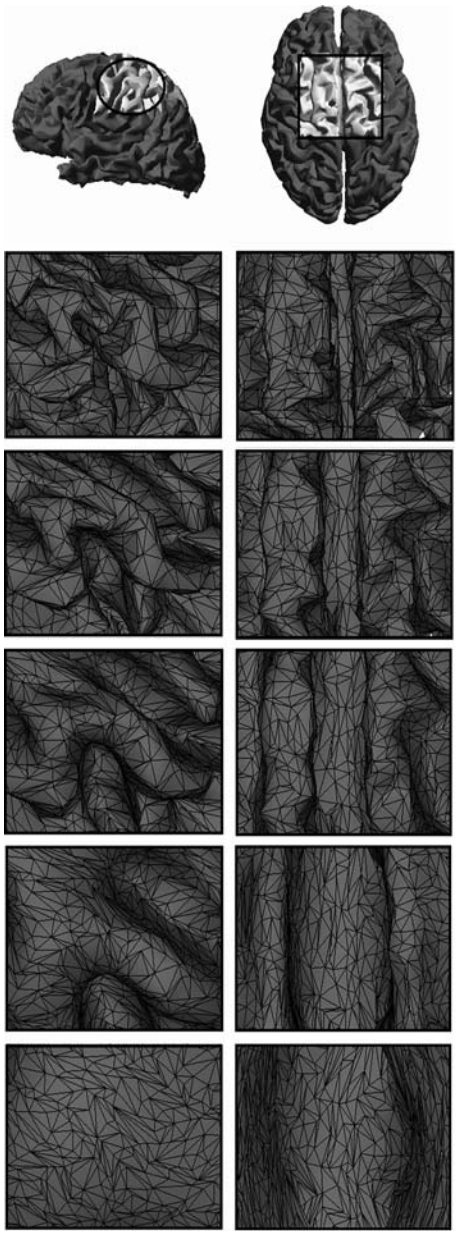

Fig. 9 Unfolding of for two different regions of a cortex using visibility maximization

Figure 10 shows an additional evolution in which one can clearly see how the selected region unfolds to become more visible from the external viewpoint.

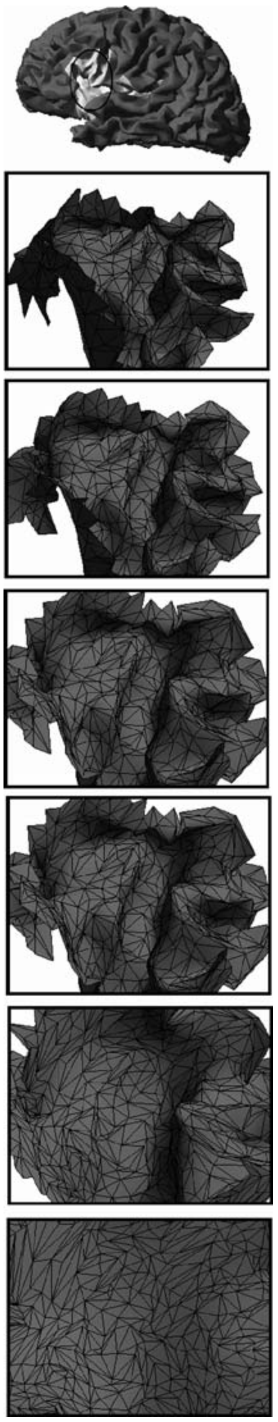
Fig. 10 Unfolding of a region of the cortex using viewpoint-based visibility maximization

7 Conclusion

We have presented a novel method for cortical surface unfolding based on a viewpoint-based visibility energy and a three-dimensional generalization of knot energy type forces for topology preservation. Simulation results show that the gradient flow of these combined energy terms yields a localized unfolding that can be used for cortical visualization and exploration tailored to the user's current viewpoint. We believe this method, compared with more traditional global flattening techniques, may be very useful for more customized inspection of cortical surface segmentations.

We have also been able to significantly increase the speed of the computations for the 3D topology preservation forces by using a tree structure and a recursion technique. We expect to use this same method for improving the speed of the visibility computations. In addition, we are exploring other ways of reducing the number of computations in order to make our unfolding approach more user-interactive in real-time.

Finally, we point out that since what we are proposing is new and since there is no "ground truth" regarding the "correct" local unfolding, it is neither clear how we should compare our approach to other approaches nor what is the proper way to apply or formulate any quantitative evaluation metrics.

References

Abrams, A., Cantarella, J., Fu, J., Ghomi, M., & Howard, R. (2003). Circles minimize most knot energies. *Topology*, *42*, 381–394.

Alexandrov, O., & Santosa, F. (2005). A topology-preserving level set method for shape optimization. *Journal of Computational Physics*, *204*, 121–130.

Angenet, S., Haker, S., Tannenbaum, A., & Kikinis, R. (1999). On the Laplace-Bertrami operator and brain surface flattening. *IEEE Transactions on Medical Imaging*, *18*(80), 700–711.

Cantarella, J. H., Demaine, E. D., Iben, H. N., & O'Brian, J. F. (2004). An energy-driven approach to linkage unfolding. In: *Proceedings of the 20th symposium on computational geometry*, pp. 134–143.

Carman, G. J., Drury, H. A., & Van Essen, D. C. (1995). Computational methods for reconstructing and unfolding the cerebral cortex. *Cerebra Cortex*, *5*, 506–517.

Durant, F. (1999). *3D visibility: analysis, study and applications*. Ph.D. thesis, University J. Fourier, Grenoble, France.

Fischl, B., Sereno, M., & Dale, A. (1999). Cortical surface-based analysis. *Neuroimage*, *9*, 195–207.

Han, C., Xu, D., Tosun, D., & Prince, J. L. (2001). Cortical surface reconstruction using a topology preserving geometric deformable model. In *Proceedings of fifth IEEE workshop on mathematical methods in biomedical image analysis (MMBIA'01)* (pp. 213–220).

Han, X., Xu, C., & Prince, J. L. (2003). A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(6), 755–768.

Hermosillo, G., Faugeras, O., & Gomes, J. (1999). *Cortex unfolding using level set methods*. Technical report 3663, INRIA.

Hertzmann, A., & Zorin, D. (2000). Illustrating smooth surfaces. In *Proceedings SIGGRAPH*, pp. 517–526.

Iben, H. N., O'Brien, J. R., & Demaine, E. D. (2006). Refolding planar polygons. In: *Proceedings of the 22nd symposium on computational geometry*, pp. 71–79.

Le Guyader, C., & Vese, L. (2007). *Self-repelling snakes for topology segmentation models*. Technical report, UCLA.

O'Hara, J. (1991). Energy of a knot. *Topology*, *30*, 241–247.

Osher, S., & Sethian, J. (1988). Fronts propagating with curvature-dependent speed: algorithms based on the Hamilton-Jacobi equations. *Journal of Computational Physics*, *79*, 12–49.

Pons, J.-P., Keriven, R., & Faugeras, O. (2004). Area preserving cortex unfolding. In *Proceedings of international conference on medical image computing and computer assisted intervention (MICCAI'04)* (pp. 376–383).

Pope, A. R. (1994). *Model-based object recognition: a survey of recent research*. Technical report TR-94-04, University of British Columbia, Department of Computer Science.

Rocha, K., Yezzi, A., Mennucci, A., & Prince, J. L. (2007). Viewpoint-based visibility maximizing flows. In *Proceedings of international conference on medical image computing and computer assisted intervention (MICCAI'07)* (pp. 499–506).

Sapiro, G., & Tannenbaum, A. (2005). Area and length preserving geometric invariant scale-spaces. *Pattern Analysis and Machine Intelligence*, *17*(1), 67–72.

Segonne, F. (2008). Active contours under topology control genus preserving level sets. *International Journal on Computer Vision*, *79*(2), 107–117.

Sethian, J. A., & Adalsteinsson, D. (1997). An overview of level set methods for etching, deposition, and lithography development. *IEEE Transactions on Semiconductor Manufacturing*, *10*(1), 167–184.

Shi, Y., & Karl, W. C. (2004). Differentiable minimin shape distance for incorporating topological priors in biomedical imaging. In: *IEEE international symposium on biomedical imaging*, pp. 1247–1250.

Slabaugh, G., & Unal, G. (2005). Active polyhedron: surface evolution theory applied to deformable meshes. In *Proceedings of IEEE Computer Society conference on computer vision and pattern recognition (CVPR'05)* (pp. 84–91).

Sundaramoorthi, G., & Yezzi, A. (2005). More-than-topology-preserving flows for active contours and polygons. In *Proceedings of the tenth IEEE international conference on computer vision (ICCV'05)* (pp. 1276–1283).

Unal, G., Yezzi, A., & Krim, H. (2005). Information-theoretic active polygons for unsupervised texture segmentation. *International Journal of Computer Vision*, *62*(3), 199–220.

Wandell, B. A., Chial, S., & Backus, B. T. (2000). Visualization and measurement of the cortical surface. *Journal of Cognitive Neuroscience*, *12*(5), 739–752.

Witkin, A., & Baraff, D. (1997). *Physically based modeling: principles and practice*. SIGGRAPH Course Notes, pp. 1–13.

Woo, A., Poulin, P., & Fournier, A. (1990). A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, *6*(10), 13–32.