MORPHING ACTIVE CONTOURS: A GEOMETRIC APPROACH TO TOPOLOGY-INDEPENDENT IMAGE SEGMENTATION, TRACKING, INTERPOLATION AND MORPHING

Master's Thesis of *Marcelo Bertalmío*

July 21, 1998

Contents

1 Introduction.

There is a large number of applications in which we can use information from one or more images to perform a certain operation on another. Examples of this are given in figure 1. On the top row we have two consecutive slices of neural tissue obtained from electronic microscopy: given a segmentation on the first slice, the image on the left, one would want that information to drive the segmentation on the second slice, the image on the right. On the bottom row we see two consecutive frames of a video sequence: once again, given a certain object in the first image, one would like to track that object finding its location on the second image.

In both these examples what we want is to find an *object* on the second image that suitable corresponds to a given object on the first one. We shall attempt to define these terms later, but here we are using *images* as n-dimensional arrays of data, while we can imagine objects to be sets of closed curves (surfaces) on 2D(3D) images, corresponding to certain edges on the image.

In other cases we have the corresponding objects in two images, and we want to find a transformation that continuously takes one into the other. Examples of this are given in figure 2. On the top row we see the contour of a synaptic button on two consecutive slices of neural tissue. As these images were obtained with electronic microscopy, the resolution in the XY plane is much higher than that on the Z axis. Therefore it is essential for a good 3D reconstruction of the button -from its edges on the slices- to interpolate the edges in between the slices. This can be accomplished if we can continuously drive the edge in one slice to the edge on the next. On the bottom row we see two 3D objects, each of them a set of closed surfaces: here we want a metamorphosis between them, what is usually called *morphing.*

We see from the kind of applications abovementioned that topological changes in the objects may and do occur often: due to occlusions in a video sequence, due to the non-convexity of the sliced volume in a series of 2D slices of a volume, due to the topological nature of the objects involved in the morphing, etc.

The goal of this work is to introduce a novel framework for the solution of these and other problems. It is based on deforming one image toward another via a Partial Differential Equation (PDE) while tracking the deformation of the curves of interest in the first image with an additional, coupled PDE: hence its name of Morphing Active Contours (MAC). The topology of the deforming curve can change without any special topology handling procedures added to the scheme.

This work is organized as follows:

- Section 2 covers the fundamentals and basic background on Curve Evolution Theory and its numerics. It is intended to be a non-exhaustive review, with references to

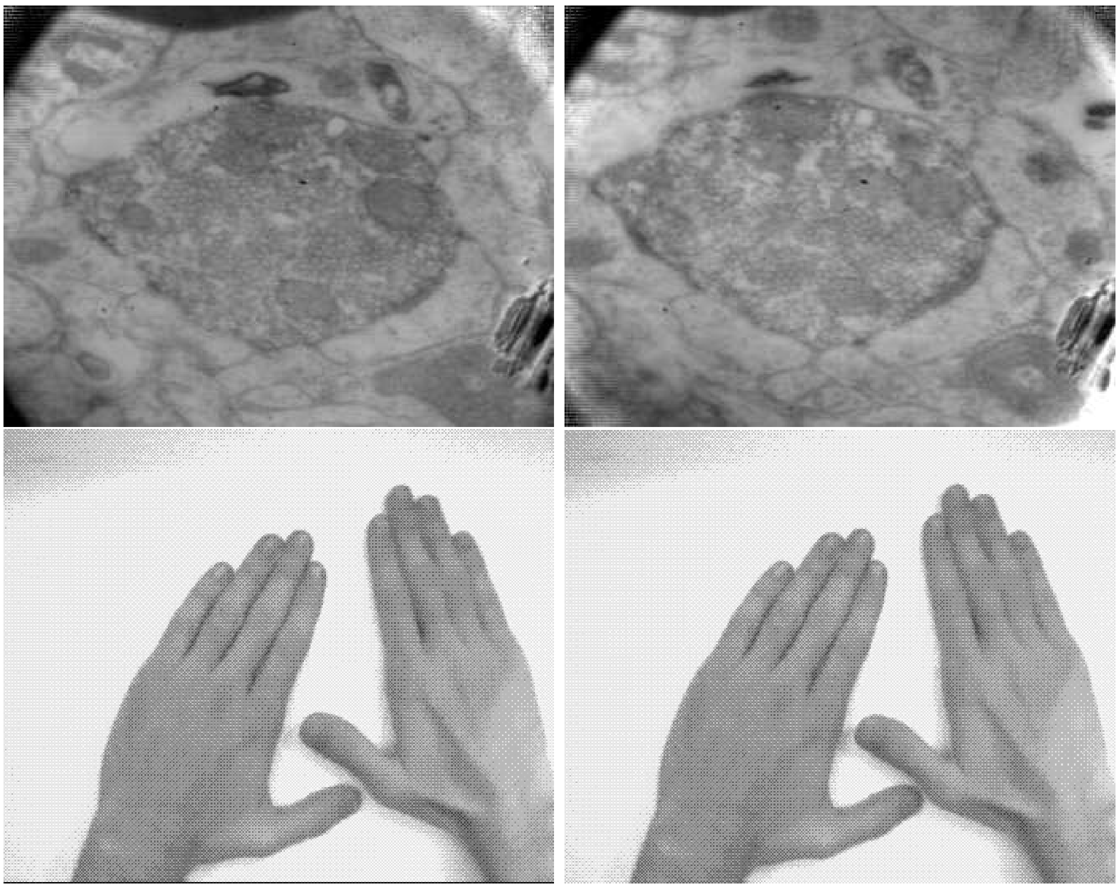

Figure 1: Examples of the problems addressed in this work. Top row: two consecutive slices of neural tissue obtained with electronic microscopy. Bottom row: two consecutive frames of a video sequence.

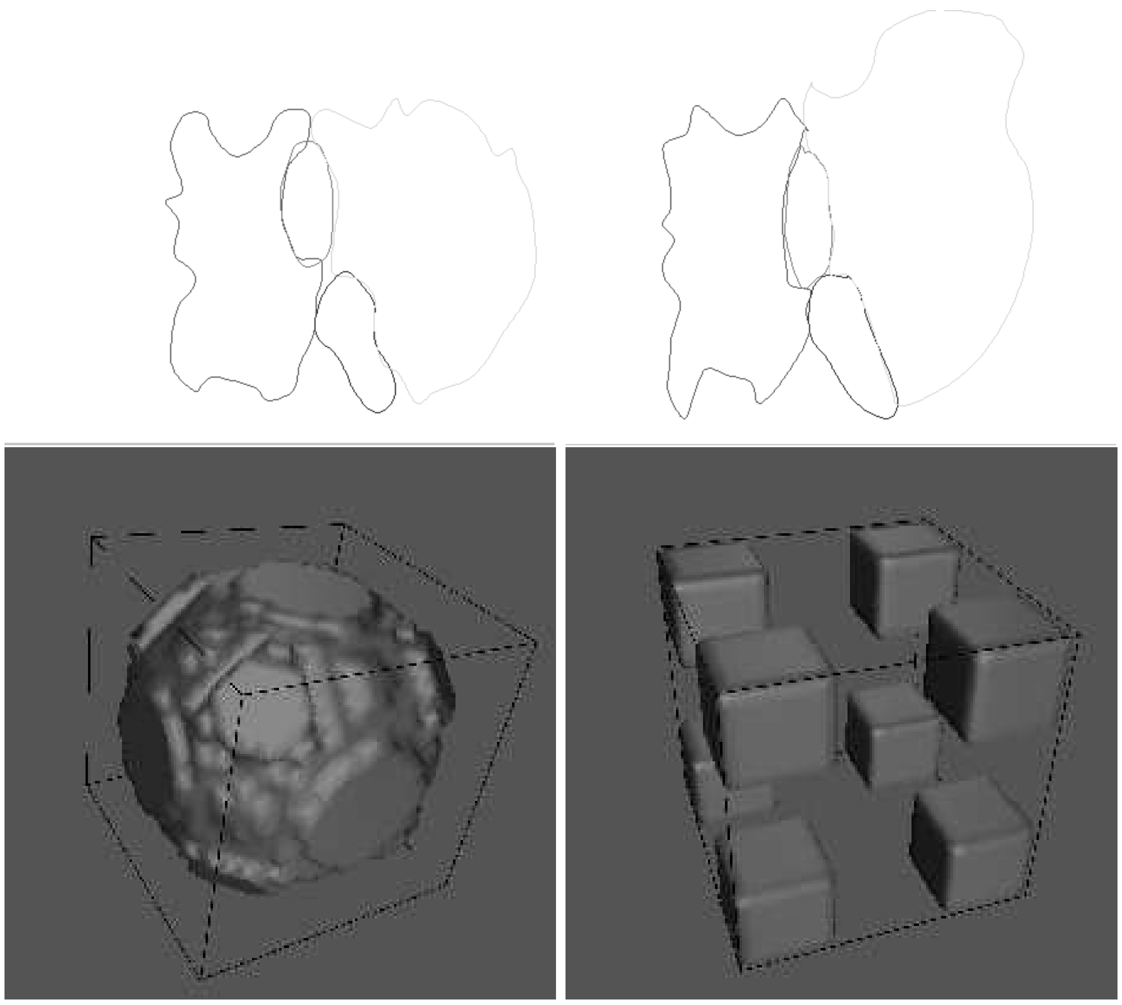

Figure 2: Examples of the problems addressed in this work. Top row: edges of a certain object in two consecutive slices of neural tissue obtained with electronic microscopy. Bottom row: two 3D objects, each one a set of closed surfaces.

in-depth studies on several areas of the subject.

- Section 3 introduces the general formulation of the MAC algorithm, regarding its validity, stability and convergence.
- Section 4 covers the application of MAC to segmentation and tracking. Segmentation is focused on the case of Electronic Microscopy (EM) neuronal images, and results are compared with the Geodesic Active Contours algorithm, but the analysis given is general.
- Section 5 covers the application of MAC to interpolation and morphing. Again, interpolation is focused on the case of EM neuronal images, but the validity of the algorithm is general.
- Section 6 introduces the concept and necessity of flattening in 3D cortical images. A novel algorithm for the tracking of curves on evolving 3D surfaces is presented. The MAC algorithm is adapted to accelerate convergence of the tracking algorithm.
- Section 7 covers conclusions and future work.

2 Fundamentals of Curve Evolution Theory.

2.1 Introduction.

The use of partial differential equations (PDE's) for image processing became a major research topic in the past years. The idea is not to think of image processing in the discrete domain but in the continuous one, combined with efficient numerical implementations. In general, let $\Phi_0 : R \times R \rightarrow R$ represent a gray-level image, where $\Phi_0(x, y)$ is the gray-level value. The algorithms we describe are based on the formulation of PDE's of the form:

$$\Phi_t = F(\Phi(x, y, t))$$

where $\Phi(x, y, t) : R^2 \times [0, \tau) \rightarrow R$ is the evolving image, $F : R \rightarrow R$ is a given function wich depends on the algorithm, and the image Φ_0 is the initial condition. The solution $\Phi(x, y, t)$ of the PDE gives the processed image.

Most of the use of PDE's for image processing was done for image deblurring or denoising. PDE's were recently used as well for a number of problems in computer vision as shape analysis, shape from shading, segmentation, invariant shape smoothing, and mathematical morphology. For references see [4], a recent book on the topic.

A great advantage of this methodology is the accuracy achieved when efficient numerical implementations are used: the accuracy relies on the implementaion, and not on the theory, as it happens with Discrete Mathematical Morphology or classical image processing algorithms. The use of PDE's complies with the increasing effort of mathematical formalization of image processing algorithms, and while numerical implementations may be computationally expensive, the ever-increasing speed and power of computers makes computation time less and less a subject, thus helping to the spreading of these techniques, as pointed out in [16].

The Theory of Curve Evolution is based on PDE's, and its applications include invariant shape recognition, segmentation, measurement, shape from shading and stereo and surface reconstruction. The purpose of this section is to introduce the basic background on this theory to understand the rest of this work.

2.2 Basic Curve Evolution.

In this subsection we shall include excerpts from [49]. We will introduce the subject in the two dimensional case and later on, when introducing Osher and Sethian's algorithm, we will see how to extend these concepts to an arbitrary dimension.

A geometric set or shape can be defined by its boundary. In the case of bounded planar shapes, for example, this boundary consists of closed planar curves. A curve may be regarded as a trajectory of a point moving in space (the plane, for our case). Formally, we define a curve $C(\bullet)$ as the map:

$$C(p) : S^1 \rightarrow R^2$$

C can be written using Cartesian coordinates, that is, $C(p) = [x(p), y(p)]^T$, where $x(\bullet)$ and $y(\bullet)$ are maps from S^1 to R. We assume throughout that all of our mappings are sufficiently smooth, so that all the relevants derivatives may be defined. We also assume that our curves have no self intersections, that is, are embedded. Moreover, since the curve $C(p)$ is assumed to be closed, the functions $x(p)$ and $y(p)$ are periodic.

We now consider plane curves deforming in time. Let $C(p,t) : S^1 \times [0, \tau) \rightarrow R^2$ denote a family of closed embedded curves, where t parametrizes each curve. Assume that this family evolves (changes) according to the following evolution equation, a PDE:

$$\begin{cases} \frac{\partial C}{\partial t} = \alpha \overrightarrow{T} + \beta \overrightarrow{N} \\ C(p,0) = C_0(p) \end{cases} \tag{1}$$

where $\overrightarrow{N}$ is the inward Euclidean normal, $\overrightarrow{T}$ is the unit tangent, and α and β are the tangent and normal components of the evolution velocity $\overrightarrow{v}$, respectively. See [17],[28].

In order to separate the geometric concept of a planar curve from its formal algebraic description, it is useful to refer to the planar curve described by *C(p,t)* as the image of *C(p,t)*, denoted by *Img[C(p,t)]*. Therefore, if the curve *C(p,t)* is parameterized by a new parameter w such that $w{=}w(p,t)$, $\frac{\partial w}{\partial p} > 0$, we obtain *Img[C(p,t)]=Img[C(w,t)]*.

It can be proven (see [21]) that if the normal component of the velocity is a geometric function of the curve (i.e., does not depend on the parameterization) then $Img[\bullet]$ (which represents the "geometry" of the curve) is only affected by its normal component. The tangential component affects only the parameterization and not $Img[\bullet]$, which is independent of the parameterization by definition. Therefore, assuming that the normal component β of the curve evolution velocity $\overrightarrow{v}$ in equation 1 does not depend on the curve parameterization, we can consider the evolution equation

$$\frac{\partial C}{\partial t} = \beta \overrightarrow{N} \tag{2}$$

where $\beta = \overrightarrow{v} \cdot \overrightarrow{N}$, that is, the projection of the velocity vector on the normal direction. The evolution in equation 2 was studied by numerous researchers for several functions β. Different evolution equations can model different physical phenomena, such as crystal growth, the Hüygens Principle or curve shortening processes. The theory has been well studied in areas such as computational physics, differential geometry, numerical analysis, parabolic equations and viscosity solutions (see references in [49]). This type of evolution was introduced into vision by Kimia et al. (see [32]).

One of the most studied evolution equations is the one obtained by $\beta = \kappa$, where κ is the Euclidean curvature:

$$\frac{\partial C}{\partial t} = \kappa \overrightarrow{N} \tag{3}$$

Equation 3 has its origins in physical phenomena and it is called the *Euclidean shortening flow*, since the perimeter shrinks as fast as possible when the curve evolves according to 3 [25]. Gage and Hamilton [22, 23, 24] proved that a planar embedded convex curve converges to a round point when evolving according to 3. (A round point is a point that, when the curve is normalized in order to enclose an area equal to π, is equal to the unit disc). Grayson [27] proved that a planar embedded nonconvex curve converges to a convex one, and from there to a round point from Gage and Hamilton result.

Sapiro et al. [49] introduced the equation where $\beta = \kappa^{\frac{1}{3}}$. This is called the *Affine Shortening Flow*:

$$\frac{\partial C}{\partial t} = \kappa^{\frac{1}{3}} \overrightarrow{N} \tag{4}$$

This flow is the affine analogue of the euclidean one. The evolution defines an affine-invariant fully geometric scale-space. It is also numerically more satble than the euclidean shortening flow, and preserves more structure up to a higher degree of smoothing, as pointed out in [42].

Other examples are the *Grassfire Flow*, for $\beta = 1$, and the evolution equation introduced by Kimia et al., where $\beta = 1 + \varepsilon\kappa$, being ε a positive constant:

$$\frac{\partial C}{\partial t} = (1 + \varepsilon\kappa) \overrightarrow{N} \tag{5}$$

This last equation models flame propagation and crystal growth [51].

It is interesting to note that, despite the fact that equation 5 is reduced to the grassfire flow when $\varepsilon \to 0$, an initially smooth surface may develop singularities when evolving according to the grassfire flow but never when evolving according to 5, as seen in [51]. This fact leads us to the question of how to handle singularities, when PDE's require smooth curves, and will be addressed in the following subsection.

2.3 Handling of singularities: The Entropy Condition and the Theory of Viscosity Solutions.

As pointed above, the differential equation in its classical form is no longer valid from the point where a singularity forms in the solution. Nonlinear equations will frequently lead to singularities, even when the initial data are smooth.

In the case where a PDE can be written in this form:

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \tag{6}$$

it can be shown (see [54]) that this equation represents a hyperbolic conservation law of the scalar quantity u with a flux f, given by the integral equation:

$$\frac{\partial}{\partial t}\int_V u dv = -\int_{\partial V} f(u) \cdot \overrightarrow{n} da \tag{7}$$

where V is any arbitrary volume, with border ∂V, volume element dv and surface element da. This implies the existence of a single physically-realized solution, but there are too many functions in the space of generalized functions ([54]) satisfying 7. It is necessary to impose certain conditions to choose one single solution. The following concepts are extracted from [32].

The characteristics of equation 6 are trajectories in the space-time domain, over which u satisfying equation 6 remains constant. For an initially smooth function u there may be a time t in the evolution where characteristics clash: two characteristics enforce two possible values for u. The physically meaningful value is chosen by enforcing conservation at the shock, the so-called *jump condition.*

Another problem arises when there exist diverging characteristics: there will be points whose value can not be determined. The gap may be filled using the jump condition, however we find that many solutions exist in conjunction. The Entropy Condition determines the solution by imposing that characteristics always flow into the discontinuity.The Entropy Condition, as proposed by Sethian ([51])for the propagation of flame fronts states that a propagating front satisfies the entropy condition if "once a particle burns, it remains burnt". Since in the process of deformation of a shape, local portions of the boundary may cross over each other, this condition rules out those solutions where the front passes through itself.

Lax proved in [33] that a generalized solution of 6 which has only shocks satisfying both the jump condition and the entropy condition, exists and is unique.

Recapping, a smooth initial condition deforms according to equation 6 up to the point where a singularity forms in the solution: after that, the PDE in its classical form is no longer valid, yet the principle of conservation as represented by its integral form in equation 7 remains in force. We can then look for a non-necessarily continuous generalized or "weak" solution, imposing certain conditions to pick only one from the possible infinite number of them. Lax's theorem says that the weak solution thus chosen is unique, and if a classical solution exists it is identical to the weak one.

If we add to equation 6 a viscosity term

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = \varepsilon g(u, u_x, u_{xx}) \tag{8}$$

it can be proven that no singularity will appear in the evolution. This is a reaction-diffusion equation: the second term of the left-hand side is the reaction term and tends to the formation of singularities, while the right-hand side is the diffusion term and tends to smooth the front. It would be logical to impose that when ε tended to 0 the limit of the continuous solution would be the solution we choose from among the weak ones: this solution is the same we obtain if we impose the entropy condition, as Barles proved in [3].

What happens if the original curve is not smooth? And when our equation does not have a formulation as in 5? When dealing with arbitrary β functions and/or non-smooth initial conditions, we must resort to the Theory of Viscosity Solutions [13]. This theory was developed in the early eighties, and started to be used to address problems in Computer Vision and Image Processing in the early nineties. Informally, this theory deals with a curve with singularities by surrounding it with two smooth curves, one in the inside and the other in the outside, taking them to the limit and using the causality principle in the evolution (by which enclosed curves will not cross each other when evolving).

While the abovementioned formulations solve the problem of the formation of singularities, this is not the only problem that may invalidate our curve evolution as the solution of a PDE. A curve may split into two in a finite time, thus having to deal with a topologically diferent entity beyond some time in the evolution, as shown for instance in the dumbbell example in [51]. To address this problem (both conceptually and numerically) we will introduce Osher and Sethian's Algorithm [43].

2.4 Osher and Sethian's Algorithm.

2.4.1 The algorithm.

The algorithm devised by Osher and Sethian in 1988 deals with topological changes in the evolving curve by embedding it in a higher dimensional surface. What in the evolution of a 2D closed planar curve appears as the splitting in two, in a 3D surface does not have a special appearance and is handled automatically, without special procedures. Besides, the algorithm proposes a reliable numerical solution of the abovementioned singularity problems, using a discretization scheme that can be associated with the pixel grid of digital images, as we shall see later.

Consider the general formulation of the evolution equation, equation 2. We construct an arbitrary continuous and Lipschitz function $\Phi : R^2 \times [0, \tau) \rightarrow R$ such that the initial zero-level set of its graph is the initial curve $C_0(p) = C(p, 0)$. In the following we assume that Φ is negative in the interior and positive in the exterior of the zero-level set. Consider the zero-level set, defined by

$$\left\{ \overrightarrow{X}(t) \in R^2 : \Phi(\overrightarrow{X}, t) = 0 \right\} \tag{9}$$

We have to find an evolution equation of Φ, such that the evolving curve $C(t)$ is given by the evolving zero level $\overrightarrow{X}(t)$, that is,

$$C(t) = \overrightarrow{X}(t), \forall t \tag{10}$$

By differentiation of 9 with respect to t we obtain

$$\nabla \Phi(\overrightarrow{X}, t) \cdot \overrightarrow{X}_t + \Phi_t(\overrightarrow{X}, t) = 0 \tag{11}$$

Note that for the zero-level set, the following relation holds:

$$\frac{\nabla \Phi}{\|\nabla \Phi\|} = -\overrightarrow{N} \tag{12}$$

In this equation, the left side uses terms of the surface Φ while the right side is related to the curve C. The combination of equations 9 to 12 with 2 gives

$$\Phi_t = \beta \, \|\nabla \Phi\| \tag{13}$$

and the curve C, evolving according to 2, is obtained as the zero-level set of the function Φ, which evolves according to 13.

This method gives us a strightforward way of extending a curve evolution algorithm to an arbitrary dimension. For instance, if we wanted to extend the grassfire flow to 3D surfaces, and we had an initial surface S_0, all we need to do is to construct a continuous Lipschitz function $\Psi : R^3 \times [0, \tau) \rightarrow R$, such that the initial zero-level set of Ψ is S_0, and the evolution of Ψ would be

$$\Psi_t = \|\nabla \Psi\|$$

This is easily extended to an arbitrary dimension, provided we can compute β (this could be tricky if, for example, β were a function of the euclidean curvature in 2D, because in 3D there are two main curvatures: see section 6.3.2).

Numerical implementation, surface constructing and contour finding are discussed in the following subsections.

2.4.2 Numerical implementation.

The numerical implementation must aproximate the evolution equation, and it must be robust, i.e., it must be accurate and stable. Sethian [51] proved that a simple, Lagrangian, difference approximation, requires an impractically small time step in order to achieve stability, while in some cases stability implies a time step equal to zero. In the Lagrangian formulation (from the field of fluid dynamics) equations of motion are based on the flow of particles. The basic problem with this is that the marker particles on the evolving curve may come very close during the evolution. This could be solved by a redistribution of the marker particles, altering the equations of motion in a nonobvious way. As an alternative, Osher and Sethian proposed an Eulerian formulation, in which quantities are a function of their position, the front propagation is written in terms of a fixed coordinate system in the physical domain. As pointed out above, there are three advantages to this approach. First, since the underlying coordinate system is fixed, discrete mesh points do not move and the stability problems that plagued the Lagrangian approximations may be avoided. Second, it clearly works in any number of space dimensions. Third, the *rectangular* grid of the Eulerian formulation is naturally associated with the pixel/voxel grid of digital images.

It can be easily shown [51] that an Eulerian formulation using central differences in space may be intrinsically unstable in some cases. The reason for this is that a central difference approximation does not handle correctly the fact that corners may develop in the evolution of a smooth curve. An entropy condition must be imposed on the numerical scheme, analogous to the one we impose to pick the correct weak solution of the evolution. Osher and Sethian propose such a scheme: first the scheme must approximate the hyperbolic conservation law, and then they impose monotonicity on it to guarantee that the scheme picks out the correct entropy-satisfying weak solution. Besides, theirs is an *upwind* scheme, one that calculates derivatives in the direction of the outward flowing normals: thus, necessary numerical boundary conditions far from the region of interest do not flow backwards and create spurious solutions.

The stability analysis of this scheme can be found in [4], where stability conditions for the time step and the space step are given for several β functions, in arbitrary dimensions.

2.4.3 Embedding function.

The algorithm by Osher and Sethian requires that we construct an arbitrary continuous Lipschitz function Φ such that its zero-level set is a given curve C_0. A simple way to do so is to construct the signed-distance function for C_0: this is a continuous Lipschitz function with negative values in the inside of C_0, positive in the outside and with value equal to zero over C_0. Morel [38] proposed an evolution equation to approximate this function:

$$\Phi_t = \delta_0(1 - |\nabla\Phi|)\,|\nabla\Phi|$$

where δ_0 has a value of 1 outside C_0 and -1 inside it, and $\Phi(t = 0)$ can be an arbitrary smooth function. The steady state of this equation is the desired function.

Nevertheless, for our numerical implementations a simple binary function such as δ_0 works well as the initial state of our embedding function, since for the computer "every function is Lipschitz", so to speak.

2.4.4 Contour finding algorithms.

Since the evolution of a curve has been transformed by Osher and Sethian's algorithm in the evolution of the zero-level set of a 3D surface (a graph, actually), whenever we want to know what is the solution of the 2D evolution equation we must find the zero-level set of the corresponding graph.

A very simple contour finding algorithm for the 2D case is described in [38], but we developed another one that shows very good results with simpler computations. The algorithm works as follows. For each pixel *(i,j)* on the image, we look at its neighbors *(i,j+1)* and *(i+1,j)*. If there is a change of sign between *(i,j)* and *(i,j+1)*, we find the first order approximation of the zero-crossing, and we mark as "on the border" that pixel which is nearer to the zero crossing. We procceed in an analogous way for *(i,j)* and *(i+1,j)*. What we have at the end is another image, a binary one, with each pixel being either "on the border" or not: we do not get a parametrized curve, but this is not the point, as we work with the images of the curves. Furthermore, for display purposes is easier to have a description of the curve (or set of curves) in that way, rather than a list of consecutive nodes.

For the 3D case, we use the classic *marching cubes* algorithm [36]. This algorithm basically works in two steps. First, it locates the surface corresponding to a user-specified value and linearly interpolates it by triangles. Then, to ensure a quality image of the surface, it calculates the normals to the surface at each vertex of each triangle (the rendering algorithms use this normal to produce Gourad-shaded images). Surface finding is done as follows: for each voxel in the volume, we consider a cube of side one with neighbor voxels as vertexes. Since there are eight vertexes in each cube and two states, inside and outside, there are 256 ways a surface can intersect the cube. These 256 states can be reduced to 14 patterns by simmetry, each of them with one to four triangles interpolating the surface. For each cube, its corresponding pattern is found and the interpolating triangles' vertex computed. Finally, the unit normal for each vertex is estimated by computation of the gradient using central differences.

For our purposes, we adapted a C implementation of this algorithm to our C++ programs, producing a file that can be rendered using public domain visualization software.

2.5 Anisotropic diffusion.

A number of algorithms for image smoothing have been developed based on the Euclidean and affine shortening flows described in section 2.2. For a detailed analyisis of the subject as well as a complete set of references see [4]. Here we shall only give a brief description of the smoothing algorithm used for the images in this work.

The basic idea of anisotropic diffusion, first proposed by Perona and Malik [44], consists in performing a selective smoothing that reduces noise while preserving edges. This is not what happens when when we diffuse by convolution with a Gaussian, for example. Assuming that edges are related to high gradient, anisotropic diffusion means diffusion in the direction perpendicular to the gradient:

$$\Phi_t = \Phi_{\zeta\zeta}$$

where Φ is a 2D image and $\zeta \perp \nabla\Phi$. But it can be shown that:

$$\Phi_{\zeta\zeta} = |\nabla\Phi|\, div \left(\frac{\nabla\Phi}{|\nabla\Phi|} \right) = \kappa\, |\nabla\Phi|$$

Thus the anisotropic diffusion of the image Φ is achieved with the evolution equation

$$\Phi_t = \kappa\, |\nabla\Phi|$$

Note that this evolution is such that the level sets of Φ move according to the Euclidean shortening flow given in equation 3. Affine invariant smoothing can be achieved using the affine shortening flow of equation 4, as explained in [42]. We pointed out above that affine smoothing is more stable and preserves more structure than the Euclidean one.

Anisotropic diffusion was performed on all the images used in this work, prior to any processing, for noise-removal purposes. This also applies to 3D images, though we did not use the straightforward extension of the algorithm:

$$\Psi_t = \kappa \left|\nabla\Psi\right|$$

where $\kappa = 0.5(\kappa_1 + \kappa_2)$, and $\kappa_1$and κ_2 are the main curvatures. Instead, smoothing is performed with another equation also dependent on the main curvatures that preserves more the topological characteristics of the volumes (see section 6.3.2).

3 Morphing Active Contours.

3.1 Introduction.

This section presents the Morphing Active Contours (MAC) algorithm, making use of the background on Curve Evolution Theory introduced in section 2. Before we write the evolution equations of the algorithm, a heuristic fundamentation of the method is given. After that, two fundamentations follow. The fisrt one for the case in which the object is a level set of the image. The second one is the general one, for an *almost* arbitrary object that behaves locally as a level set of the image. A stability condition for the time step and the space step is found, using Von Neumann's stability criterion. Finally, the problem of non convergence to the second image is addresed.

3.2 Heuristics.

The MAC algorithm was developed while working on automatic segmentation of 2D consecutive images of neural tissue. While automatic segmentation will be addressed in the next section, a basic knowledge of the problem is required to understand the motivation of the algorithm. Hence we will give the *heuristics* of the algorithm by explaining what was that we wanted it *to do* in the first place.

Briefly (as we shall go over this in deep in the next section) automatic segmentation of sequences of Transmission Electron Microscope (TEM) images of neural tissue is very hard to achieve. The main reasons for this are:

1. These are very noisy images, with artifacts and non-uniform illumination.
2. To find the object of interest in an image the help of a trained neurobiologist is required, since several objects may look similar in terms of shape and structure while only some are objects of interest.

The second problem is the most difficult to solve. It clearly invalidates clasical edge detectors, and shows poor results with the Geodesic Active Contour model, as will be discussed in section 4.2.3. Nevertheless, if an expert traces the border of the objects of interest in an image, any person could tell what are the corresponding objects in the following image, since we are assuming slight transitions along the sequence. No biological knowledge is required, just a "mental deformation" of the curves, from one image to the following.

This is what we tried to simulate with our algorithm. We assume that an expert has traced the curves of interest on the first image of the sequence, and we construct a surface (a graph) that has this set of curves as its zero-level set. Then we evolve the first image, deforming it continuously into the second image of the sequence, and while we do so, we deform accordingly the surface we constructed. That is, the first image evolves and we make our surface evolve in the same way. Hopefully, as the evolving first image reaches the steady state (the second image), the evolving surface has turned into a surface that has as its zero-level set the curves of interest of the second image. We will see in the following subsections that it is actually so for a wide range of sequences, that we will try to characterize.

3.3 The algorithm.

Let $I_{1_0} : R^n \rightarrow R$ and $I_2 : R^n \rightarrow R$ be two n-dimensional gray-level images.We construct an arbitrary continuous Lipschitz function $U_0 : R^n \rightarrow R$ with C_1 as its zero-level set, being C_1 a given set (a set of closed curves if $n = 2$, a set of closed 3D surfaces if $n = 3$, etc.). Then we define the evolution:

$$\begin{cases} I_{1_t} = (I_2 - I_1)\,|\nabla I_1| \\ U_t = sg(\nabla I_1 \cdot \nabla U)(I_2 - I_1)\,|\nabla U| \end{cases} \tag{14}$$

for the images $I_1 : R^n \times [0, \tau) \rightarrow R$ and $U : R^n \times [0, \tau) \rightarrow R$, with initial conditions:

$$\begin{cases} I_1(t = 0) = I_{1_0} \\ U(t = 0) = U_0 \end{cases} \tag{15}$$

Here, as well as in the remaining of this work, we have omitted the variables of the functions for simplicity. We will note $I_1(\overrightarrow{X}, t)$ as I_1, $U(\overrightarrow{X}, t)$ as U, $I_2(\overrightarrow{X})$ as I_2, and so on, with $\overrightarrow{X} \in R^n$.

We see that equation 14 is a system of coupled PDE's: the evolution of U depends on the evolution of I_1. Note that I_2 remains fixed throughout the evolution, so I_1 does not depend upon the evolution of U. Coupled PDE's have already been used in the past to address other image processing tasks, see [45] [4] and references therein.

The first of the equations in 14 makes I_1 evolve towards I_2. We can see that, $|\nabla I_1|$ being always positive, when I_1 is greater than I_2 then I_1 decreases: $I_{1_t} \leq 0$, because $(I_2 - I_1) < 0$. Conversely, when I_1 is smaller than I_2 then I_1 increases. Finally, we see that I_2 is a steady state solution of the first equation in 14: if $I_1 = I_2$, then $I_{1_t} = 0$ and the evolution stops. In other words, the steady state of equation 14 is obtained when

$$lim_{t \rightarrow \infty} I_1 = I_2 \tag{16}$$

But here we have assumed that $|\nabla I_1| \neq 0$. There are some cases where I_1 will not converge to I_2, for instance when $I_{1_0} = 0 \; \forall \overrightarrow{X}$, because $|\nabla I_1| = 0$ and therefore $I_{1_t} = 0$: the image I_1 will not *move* at all towards I_2. The same problem may happen if there are broad regions of I_1 with a flat gradient. But we will discuss this problem later on in this section.

The second of the equations in 14 is the "tracking" equation. It deforms the image U, whose zero-level set follows the original curve/surface C_1, by an amount that depends on the simultaneous deformation of I_1 towards I_2. Again, provided that $|\nabla I_1| \neq 0$, the steady state of this equation is achieved when $I_1 = I_2$, for in that case $U_t = 0$.

When the steady state of 14 is obtained, the current image I_1 has been totally deformed towards the next one, I_2, and U has evolved into a surface whose zero-level set is a curve/surface C_2. Ideally, C_2 is a curve/surface on I_2 that *corresponds* to the "user-defined" curve/surface C_1 on I_1. This correspondence is analysed in the following subsection.

Actually, both C_1 and C_2 are, more generally, sets of curves/surfaces. We use Osher and Sethian's algorithm, and the point of image U is to embed C_1 and track its evolution. Therefore, as we saw in the previous section, U is an arbitrary continuous Lipschitz function, such as the signed distance from C_1. Furthermore, topology may change from C_1 to C_2 without any special procedures needed to handle this: C_1 may be one curve and C_2 two curves, C_1 may be a sphere and C_2 eight cubes, C_1 may be a series of spheres and C_2 a series of pairs of cubes, and so on. The topology of the objects of interest is an information "included" in some way in U, and the evolution of U does not depend on this information: whenever we need to know the topology of $C_1(t)$, we just find the zero-level set of U. But again, the evolution of U is performed without explicit knowledge of its level sets. The advantages of this formulation are quite clear.

3.4 Correspondence between C_1 and C_2.

In the previous subsection we defined C_1 as a given level set, specified (by an expert) on I_1. Nevertheless, C_1 does not have to be a level set of I_1: it is just a set of curves/surfaces, that *someone* thinks corresponds to the boundary of an object of interest on I_1. Then, $C_1(t)$ evolves (as the zero-level set of the abovementioned surface U), finally taking the shape of a set of curves/surfaces that we call C_2:

$$C_2 = lim_{t \to \infty} C_1(t)$$

The point of the algorithm is, informally speaking, that the same *someone* who specified C_1 as the boundary of the object of interest on I_1, would specify this newly found C_2 as the boundary of the same object of interest on I_2. This introduces the main problem on the subject of evaluation of the MAC algorithm for segmentation and tracking, and that is that, as the definition of the original boundary is *subjective*, so has to be the performance evaluation. But more on this in the following section.

Actually, for the MAC algorithm to work on segmentation and tracking the original boundary C_1 can not be an arbitrary set of curves/surfaces: it has to be located over (or close to) points of local maxima of the gradient. In other words, C_1must be an edge locally. Given this assumption, and mantaining the hypothesis of slight difference among consecutive images, we will now demonstrate that C_2 actually *corresponds* in a natural way to C_1.

Recapping, the two hypothesis we will use are:

1. There is only a *slight* difference between I_{1_0} and I_2. This notion of *slightness* shall be clarified in brief, but let us say that corresponds to the perceptible but minor differences found in consecutive frames of a movie, or consecutive slices of a volume.
2. The initial boundary C_1 is locally an edge, in the sense that locally separates almost uniform regions of different gray values.

We will first consider the case where C_1 is a level set of I_{1_0}: this will be the case in interpolation and morphing, but seldom in segmentation and tracking. Then we will consider the general case, having understood the behaviour of the algorithm in the previous, particular case.

3.4.1 C_1 is a level set of I_{1_0}.

Let C_1 be the α-level set of the initial image I_{1_0}, that is:

$$C_1 = \left\{ \overrightarrow{X} : I_{1_0}(\overrightarrow{X}) = \alpha \right\}$$

The first of the hypothesis we formulated states a great similarity between I_{1_0} and I_2. Then, it is natural to postulate the α-level set of the final image I_2 as the correspondent of C_1 on I_2 . Let us call this set $\widehat{C_2}$.

One could argue that, if I_1 and I_2 are both images that depict the same scene but have different illumination, then the correspondent of the α-level set of the initial image will be a different β-level set of the final image. But our "similarity" hypothesis implies also a similarity in the histograms of the images. Then, in the abovementioned case I_1 and I_2 would not comply with our first hypothesis: an histogram equalization would be needed. This is what we actually use in our implementation, where illumination changes do occur often along a sequence of otherwise "similar" images. We will return to this in the next section. For now we assume that histograms are similar, in which case the natural correspondant of the α-level set of I_1 would be the α-level set of I_2. We must now prove that the arrival set of curves/surfaces C_2 is indeed $\widehat{C_2}$.

Assuming I_{1_0} to be continuous and Lipschitz, we can construct the initial tracking surface U_0 as

$$U_0 = I_{1_0} - \alpha$$

This assumption is made only in order to forget about the term $sg(\nabla I_1 \cdot \nabla U)$ (that will be equal to 1 everywhere), simplifying the analysis. The purpose of this term will be clarified very soon.

For this particular case, we could rewrite equation 14 as

$$\begin{cases} I_{1_t} = (I_2 - I_1)\,|\nabla I_1| \\ U_t = (I_2 - I_1)\,|\nabla U| \end{cases} \tag{17}$$

From the formulation of Osher and Sethian seen in the previous section, this evolution equation implies that all the level sets of both I_1 and U are evolving simultaneously with the image dependent velocity $(I_1 - I_2)$. If we call $C_1(t)$ the α-level set of I_1, and $C_u(t)$ the zero-level set of U, then equation 17 implies the following:

$$\begin{cases} C_{1_t} = (I_2 - I_1)\overrightarrow{N_1} \\ C_{u_t} = (I_2 - I_1)\overrightarrow{N_u} \end{cases} \tag{18}$$

where $\overrightarrow{N_1}$ and $\overrightarrow{N_u}$ are the inward normals on $C_1(t)$ and $C_u(t)$, respectively. We bear in mind that

$$\begin{cases} C_1(t=0) = C_1 \\ C_u(t=0) = C_1 \end{cases} \tag{19}$$

since C_1 is both the α-level set of I_{1_0} and the zero-level set of U_0.

Then, from equations 18 and 19 we get that $C_1(t)$ and $C_u(t)$ have the same initial condition and the same velocity for every t. Therefore, the two curves/surfaces have the same image for every t.

As $C_1(t)$ is obtained for every t as the α-level set of I_1, and assuming that equation 16 holds, then in the limit

$$lim_{t\to\infty} C_1(t) = \left\{\overrightarrow{X} : I_2(\overrightarrow{X}) = \alpha\right\}$$

or, using the notation introduced earlier

$$C_2 = \widehat{C_2}$$

which is the result we were trying to prove.

Now the reason for the term $sg(\nabla I_1 \cdot \nabla U)$, that we dropped for simplicity, becomes apparent: it guarantees that we take the normals $\overrightarrow{N_1}$ and $\overrightarrow{N_u}$ with the same sign, that is, both inward or both outward. This was not an issue for this case, in which I_{1_0} and U_0 are identical but for a constant value α, but if we had chosen U_0 as $-(I_{1_0}$-$\alpha)$, another valid choice, then the algorithm would have had to be

$$\begin{cases} I_{1_t} = (I_2 - I_1)\,|\nabla I_1| \\ U_t = -(I_2 - I_1)\,|\nabla U| \end{cases} \tag{20}$$

As the inward normal $\overrightarrow{N}$ to a level set C on a graph $\Phi$is computed as

$$\frac{\nabla \Phi}{\|\nabla \Phi\|} = -\overrightarrow{N} \tag{21}$$

when the interior of $C$is in lower level sets of Φ, and if not it is computed as

$$\frac{\nabla \Phi}{\|\nabla \Phi\|} = \overrightarrow{N} \tag{22}$$

then the term $sg(\nabla I_1 \cdot \nabla U)$ is equal to one if both normals are inward or outward, and equal to minus one if they have opposite directions, in which case it inverts $\overrightarrow{N_u}$ to make it match $\overrightarrow{N_1}$.

3.4.2 General case.

In the particular case that we have just seen, where C_1 is the α-level set of the initial image I_{1_0}, the hypothesis of similarity between I_{1_0} and I_2 is used only to justify that the natural correspondent to C_1 on image I_2 is the α-level set of I_2. In interpolation and morphing applications, I_{1_0} and I_2 will be binary images where C_1 and C_2 are both level sets, known *a priori*, and the MAC algorithm gives the desired continuous deformation that drives one into another. That will not be the case in segmentation and tracking, where if C_1 is a level set then C_2 can easily be found using a contour-finding algorithm on I_2, directly, instead of employing the evolution scheme. In the general case C_1 will comply with the second of our hypothesis but will not necesary be a level set: nevertheless, one can see this hypothesis as being equivalent to "C_1 must be a level set, locally". Also, we will have to specify better the notion of similarity between I_1 and I_2.

Throughout this subsection we will use 1D examples of the MAC algorithm, since the choice of dimension does not change the performance of the algorithm, and 1D examples are easier to display. In one dimension, equation 14 can be written as:

$$\left\{ \begin{array}{c} I_{1_t}(x,t) = (I_2(x) - I_1(x,t)) \left|\frac{\partial I_1(x,t)}{\partial x}\right| \\ U_t(x,t) = sg(\frac{\partial I_1(x,t)}{\partial x} \cdot \frac{\partial U(x,t)}{\partial x})(I_2(x) - I_1(x,t)) \left|\frac{\partial U(x,t)}{\partial x}\right| \end{array} \right. \tag{23}$$

But now C_1 is neither a curve nor a surface, but a set of points: those points where $U(x,0) = 0$. We can also see these 1D examples in another way: if I_{1_0} and I_2 are functions of *(x,y)* such that they have the same value for every y, i.e., $I_{1_{0_y}} = 0, I_{2_y} = 0$, then any slice of these graphs along the y-axis would look the same, as a 1D graph of the variable x, which is what we will show in the following.

Example 1. This example is depicted in figure 3. Here we see I_{1_0} is a decreasing function of x, and so is I_2, so their gradients will point "backward", toward lower values of x. Here C_1 has been chosen to be the point P shown in the x-axis. This choice of C_1 complies with our second hypothesis, since it is locally an edge, in the sense that it is located near a local maximum of the gradient. From P we constructed U_0, a continuous Lipschitz function with the only requirement that $U_0(P) = 0$. Since U_0 is also a decreasing function of x, its gradient points "backwards" too and throughout the evolution the term $sg(I_{1_x} \cdot U_x)$ will be equal to 1 for every x, for this particular case. For this reason, in those points where $(I_2 - I_1) > 0$, both I_1 and U will increase their values, since $I_{1_t} > 0$ and $U_t > 0$: the arrows show the direction of this change. Bearing in mind equations 21 and 22, this rising in the graphs of I_1 and U implies that the points $I_1(P)$ and $U(P)$ will move with the normal velocity $\overrightarrow{v}$:

$$\overrightarrow{v} = (I_2 - I_1)\frac{-\nabla I_1}{|\nabla I_1|} = (I_2 - I_1)\frac{-\nabla U}{|\nabla U|}$$

We have that for the point P, $I_1(P)$ and $U(P)$ move with the same velocity. Then, as equation 16 tells us that $I_1(P) \to I_2(P')$, then $U(P) \to U(P')$. In other words, the correspondant of P will be P'.

If we had chosen as C_1 the point Q, then the natural correspondent would have been Q'. Nevertheless, the evolution does not take Q into Q', since Q is located in a region of flat gradient of I_{1_0}: the point Q does not move in the evolution. What has happened? That this choice of C_1 does not comply with the requirement that it is located near a local maximum of the gradient of I_{1_0}.

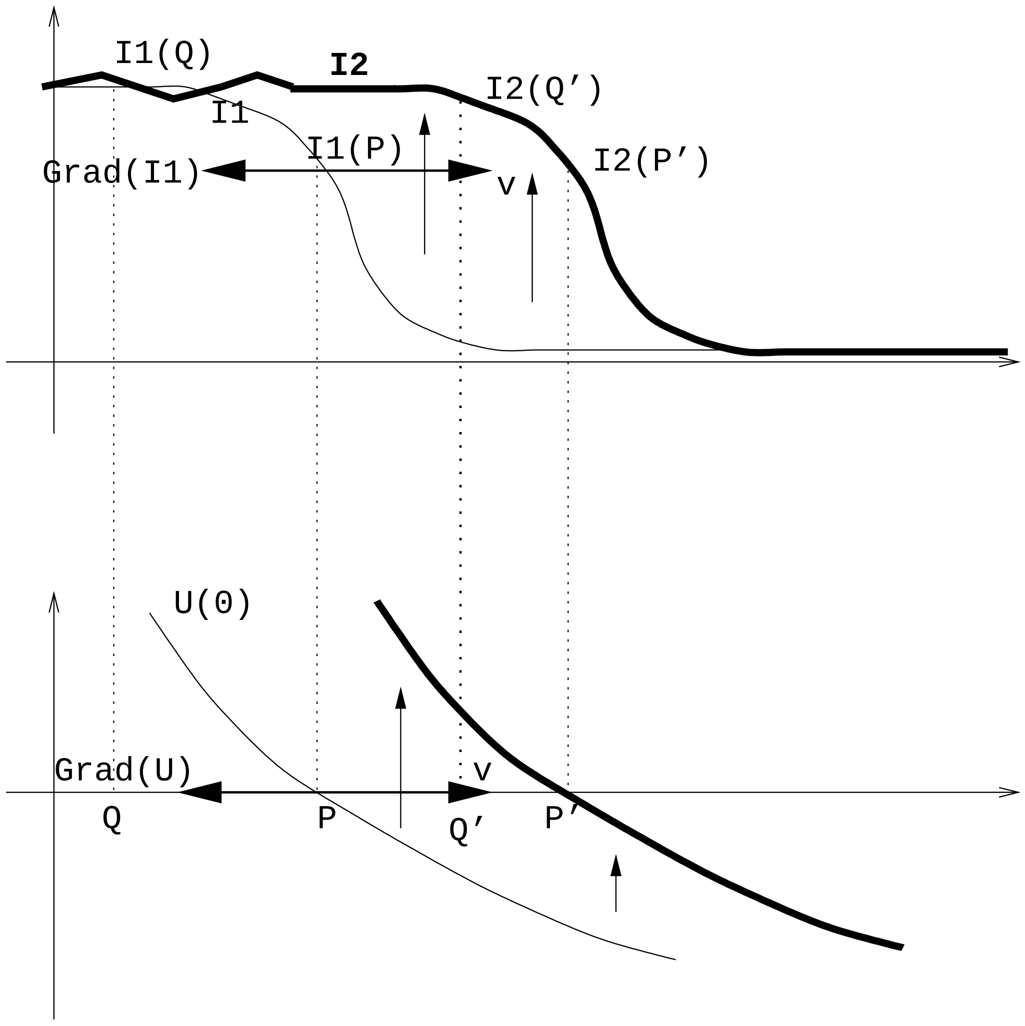

Figure 3: See example 1.

This is just a 1D example of the particular case, discussed earlier, where C_1 is a level set of I_{1_0}. But bear in mind that this example could represent a "local" view of our surfaces, and C_1 may have other points with different values than $I_{1_0}(P)$, as we will see in our third example.

Example 2. In this example (see figure 4), both $I_{1_0}$and I_2 are increasing functions of x, but we constructed U_0 as a decreasing function. As a consequence of this, the gradients of I_1 and U point in different directions and the term $sg(I_{1_x} \cdot U_x)$ will be equal to -1 for every x. Therefore, in those points where $(I_1 - I_2) < 0$ the function I_1 will decrease while U will increase, and $I_1(P)$ and $U(P)$ will move with the same velocity. Again we find that the correspondant of P is P'.

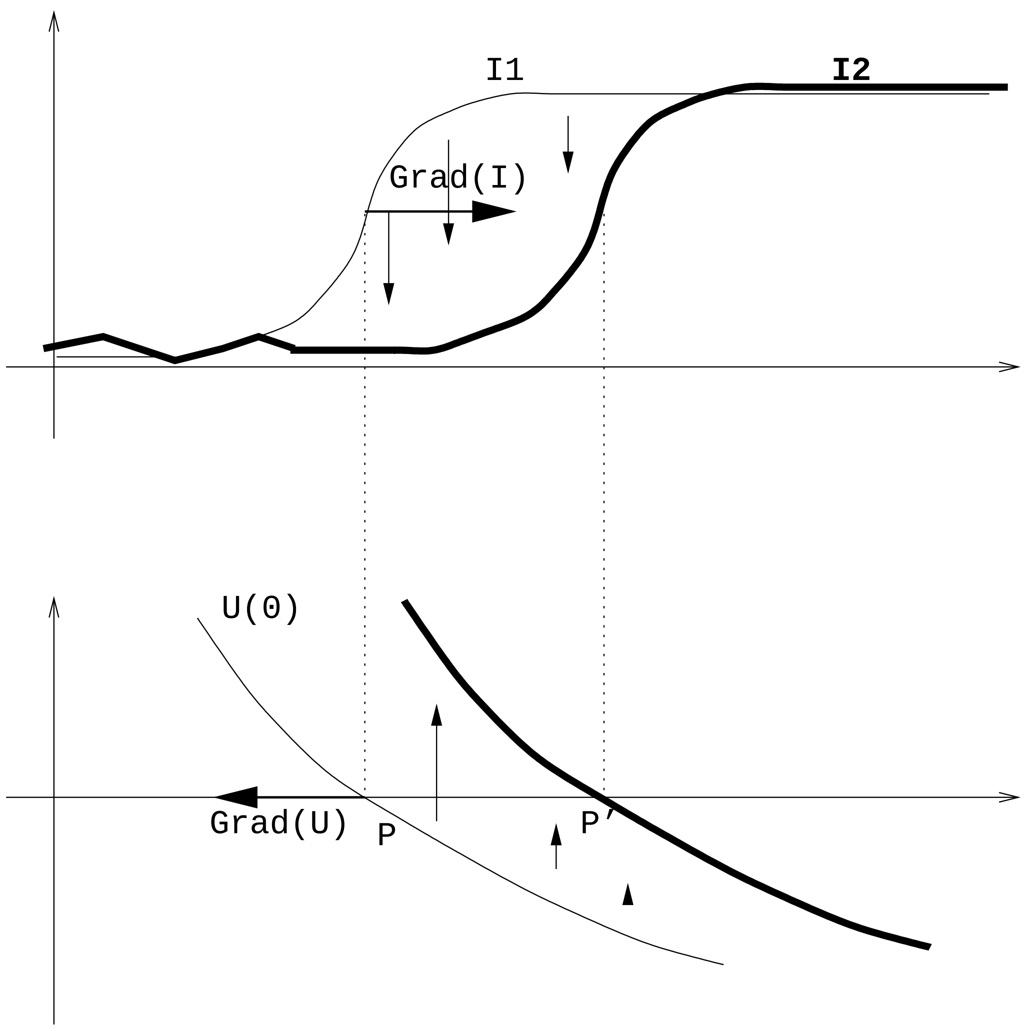

Figure 4: See example 2.

Here we see that if it were not for the term $sg(I_{1_x} \cdot U_x)$, $I_1(P)$ would move to the right while $U(P)$ would move to the left, hence the tracking of P with U would be incorrect.

Example 3. See figure 5. Here I_{1_0} is sometimes above and sometimes below I_2. Our set C_1 consists of the points P and Q, so now we are actually seeing a case where C_1 is not a level set of I_{1_0}, since $I_{1_0}(P) < I_{1_0}(Q)$. We constructed the surface U as shown, and as in example 1 the gradients of both I_{1_0} and U have the same direction for each x in the neighborhood of C_1. For this reason, in a neighborhood of P the function I_1 will increase, and so will U, while in a neighborhood of Q the function I_1 will decrease, and so will U. The final result is that when I_1 has evolved into I_2, whe obtain the set $C_2 = \{P', Q'\}$ as the correspondent of C_1, by contouring U at the zero-level. Again, C_2 is not a level set of I_2.

This example can be viewed as a "global" image with two "local" ones (one for P and another for Q): the "local" ones may be interpreted as in example 1, having as initial points "local" level sets. But the "global" image does not have a level set as an initial set of points.

Example 4. See figure 6. Again I_{1_0} is somewhere below and somewhere above I_2, but now U has its gradient pointing in the opposite direction to ∇I_{1_0} in a neighborhood of $C_1 = \{P, Q\}$. Therefore, I_1 will decrease near P but U will increase: as a result, $I_1(P)$ and $U(P)$ will move with the same velocity, in the same direction. Conversely, I_1 will increase near Q but U will decrease: again, $I_1(Q)$ and $U(Q)$ will move with the same velocity, in the same direction. The correspondant set of C_1 is $C_2 = \{P', Q'\}$.

C_2 is not a level set of I_2, but if we think of this image as composed of two smaller ones, one on the left (the neighborhood of P) and another one on the right (the neighborhood of Q) then each of them could be analysed as in example 2.

Example 5. See figure 7. This example shows an unsuccessful attempt to track the point P with our algorithm. Here I_{1_0} is a function with a "valley" near the slope where we have placed the point P. This point is near a local maxima of the gradient, so C_1 complies with our "local edge" hypothesis. I_2 is the same function I_{1_0} with a shift to the right of value S. Hence one would want the algorithm to find the correspondant of P as being the point $P' = P + S$.

Nevertheless, this is not so. P is located in the "valley" of I_2, so in a neighborhood of P the function I_{1_0} is greater than I_2 . Therefore I_1 will decrease near P, thus moving P to the left instead of moving it to the right. As a result, the algorithm tells us that the correspondant of P is P'', clearly not what we wanted.

What has happened? Here we violated our first hypothesis, that of "similarity" between I_{1_0} and I_2: both images where too much apart, the shift S was too high. The images were not similar enough, in the sense that the edge near P in I_1 was closer to the edge near P'' in I_2 than to the edge near P'. This could be seen as sensitivity of the algorithm to local maxima of the gradient, but this is not so: figure 8 shows how the high gradients near P (around the points H and J) do not affect the behaviour of the algorithm, now that the shift is smaller. The algorithm is sensitive to gray-levels, instead: the point P moves until it finds a point on I_2 , say P', such that $I_2(P') = I_{1_0}(P)$, and there it stops, since $(I_2 - I_1) = 0$ for that point.

This clarifies the notion of similarity between I_{1_0} and I_2 that we introduced in our first hypothesis: *the distance between corresponding points of C_1 and C_2 must be small enough so that between them (along the path of the evolution) there are no other points with the same gray value.* If this is not the case, the algorithm may get lost or mark as solutions points which are not. Examples of this are seen in the next section.

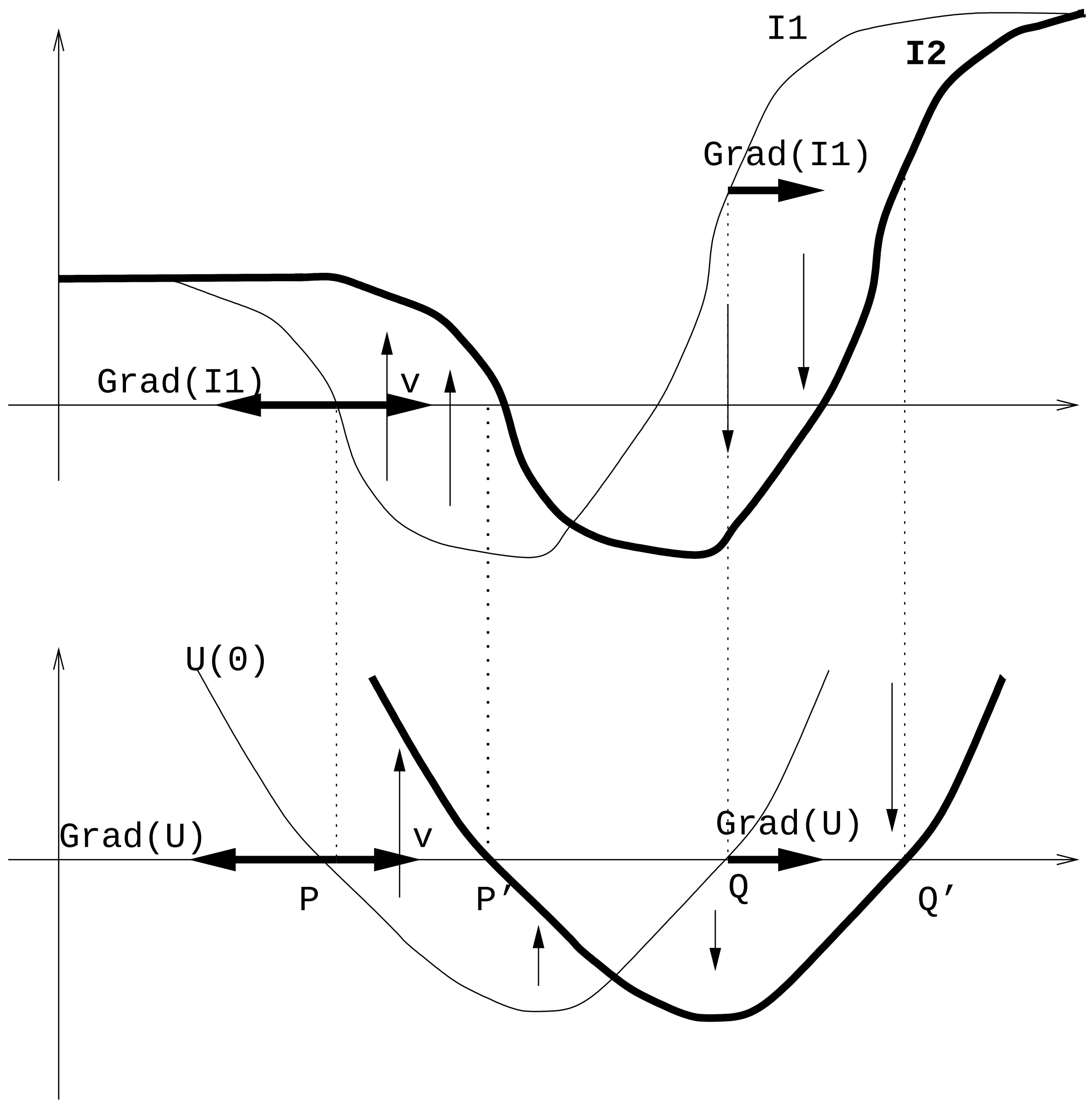

Figure 5: See example 3.

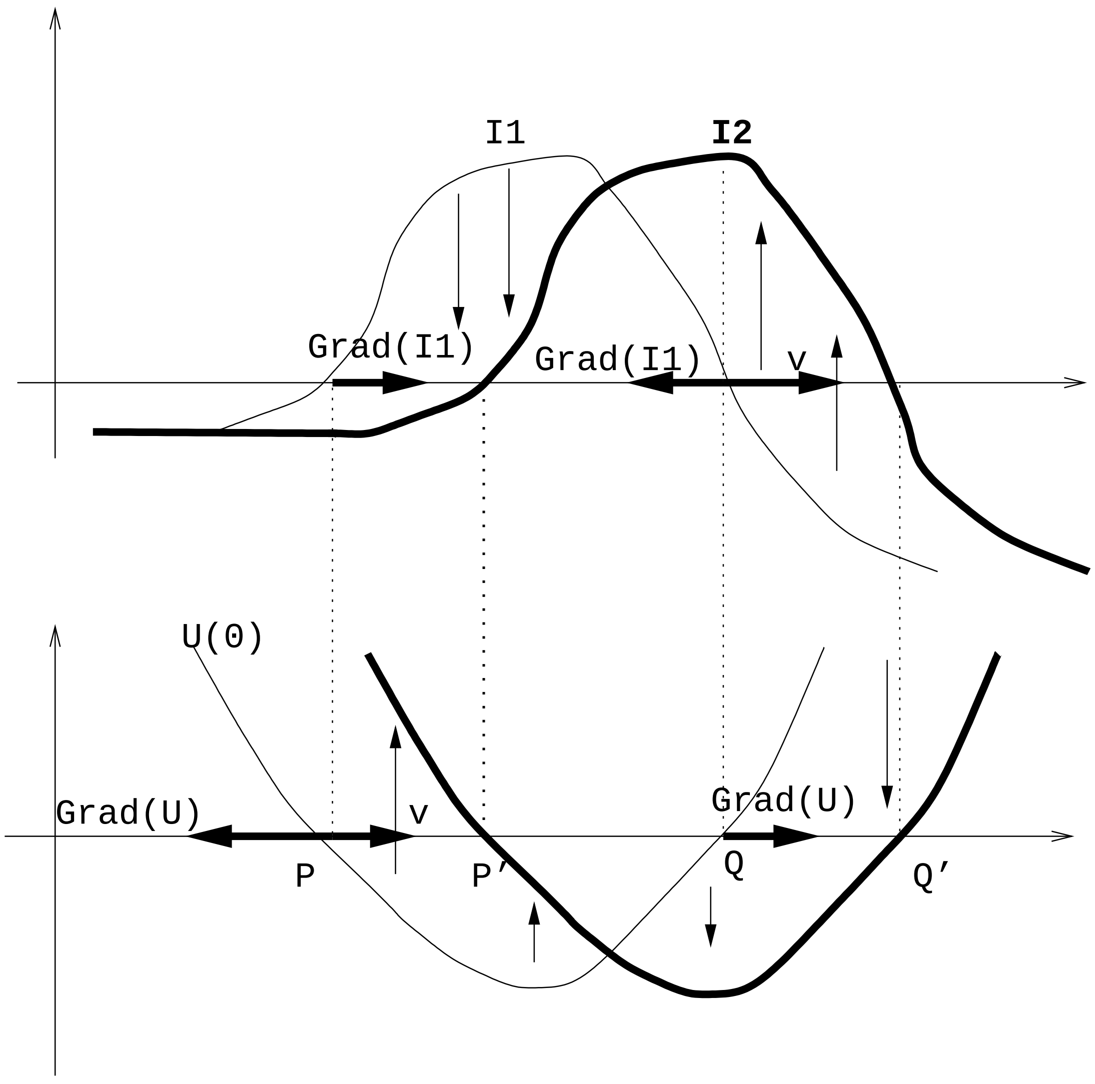

Figure 6: See example 4.

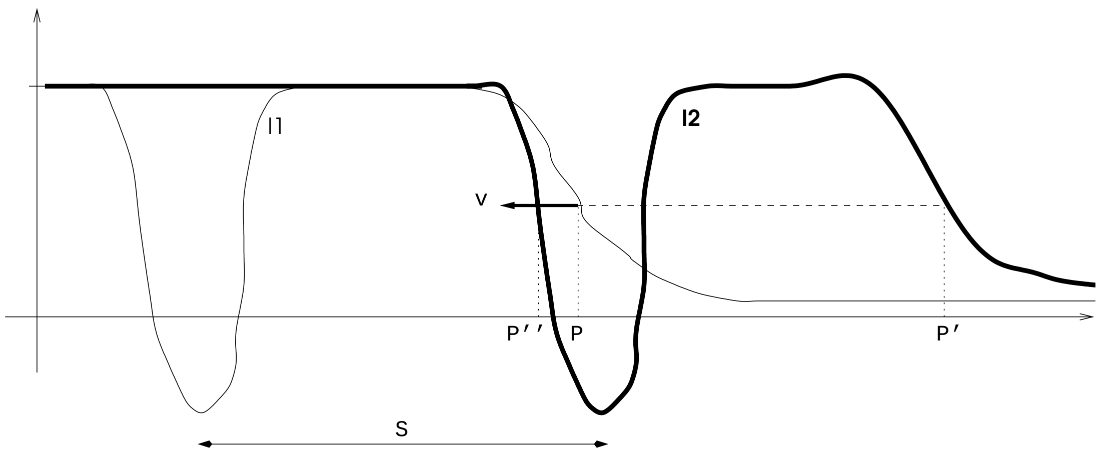

Figure 7: See example 5.

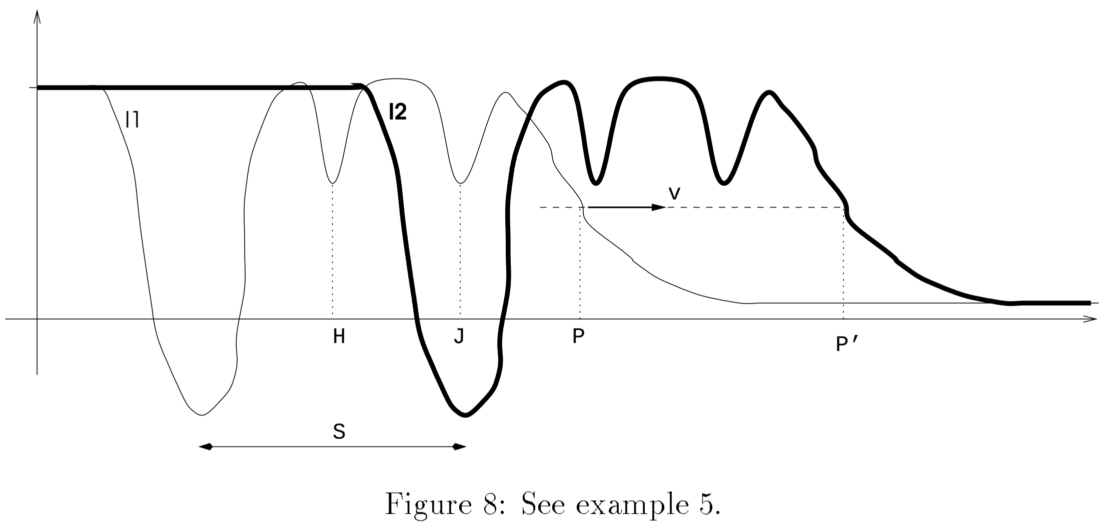

Figure 8: See example 5.

3.5 Numerical implementation.

For the numerical implementation we used Osher and Sethian's formulation for the discretization of evolution equations, as mentioned in section 2.4.2. This is an upwind scheme that satisfies an entropy condition for the discrete case, being a good approximation to the *continuous* evolution with its entropy condition (see [43]).

We now procceed to study the stablity of this scheme according to Von Neumann's criterion.

3.6 Stability of the scheme.

We will study the stability of the scheme as follows: first, we will use Von Neumann's criterion on a generic 2D evolution $I_t = c\,|\nabla I|$, with c a scalar function of time. This will give us a stability condition that vinculates Δt, Δx and c. Then we will prove that, for our equation 14, the parameters multiplying the gradient terms are always decreasing. Finally we will choose a time step Δt taking into account the previous results.

3.6.1 Stability of $I_t = c\,|\nabla I|$.

The discretization of I is performed on a grid with cells of size $\Delta x \cdot \Delta y$. The pixel with space coordinates (i,j) and time coordinate n is noted as $I^n_{i,j}$. We define the (forward) differences as follows:

$$I_x = \frac{I^n_{i+1,j} - I^n_{i,j}}{\Delta x}$$

$$I_y = \frac{I^n_{i,j+1} - I^n_{i,j}}{\Delta y}$$

$$I_t = \frac{I^{n+1}_{i,j} - I^n_{i,j}}{\Delta t}$$

This upwind scheme is only first-order accurate in the spatial derivatives. Osher and Sethian studied higher order upwind schemes [43].

For $c > 0$ and if the partial derivatives do not change sign at (i,j), the discretization of $I_t = c\,|\nabla I| = c\sqrt{I_x^2 + I_y^2}$ is then carried out as:

$$\frac{I^{n+1}_{i,j} - I^n_{i,j}}{\Delta t} = c^n_{i,j}\sqrt{\left(\frac{I^n_{i+1,j} - I^n_{i,j}}{\Delta x}\right)^2 + \left(\frac{I^n_{i,j+1} - I^n_{i,j}}{\Delta y}\right)^2}$$

The aforementioned hypothesis have the only purpose of simplifying the calculus (otherwise, the entropy condition for the discrete case would require to compute maxima and minima with zero of forward and backward differences). With the additional hypothesis of $\Delta x = \Delta y$ we have:

$$I_{i,j}^{n+1} - I_{i,j}^{n} = c_{i,j}^{n} \frac{\Delta t}{\Delta x} \sqrt{\left(I_{i+1,j}^{n} - I_{i,j}^{n}\right)^2 + \left(I_{i,j+1}^{n} - I_{i,j}^{n}\right)^2} \tag{24}$$

Von Neumann stability analysis is local and ignores boundary conditions. A difference scheme is said to be unstable if there exist initial values for which the solution $I_{i,j}^{n}$ blows up for $n \to \infty$. If we periodize the image we can express every initial value by its Fourier expansion, and for the evolution each term evolves according to:

$$I_{i,j}^{n} = \varepsilon^{n} e^{J\omega(i\Delta x + j\Delta y)} \tag{25}$$

where ω is the angular frequency, J is the complex variable and ε is a complex number which depends on ω and is called the amplification factor. If $|\varepsilon| < 1$ for all ω the equation is stable (see [1]). Inserting 25 in 24 we get:

$$(\varepsilon - 1) = c_{i,j}^{n} \frac{\Delta t}{\Delta x} \sqrt{2}(e^{J\omega\Delta x} - 1)$$

The stability requirement $|\varepsilon| < 1$ for all ω implies that:

$$\| \, c_{i,j}^{n} \frac{\Delta t}{\Delta x} \sqrt{2} \, \| < 1$$

Therefore the stability condition for the two dimensional case can be stated as:

$$\Delta t(n) < \frac{\Delta x}{\sqrt{2} max_{i,j} \left\{ \left\| c_{i,j}^{n} \right\| \right\}} \tag{26}$$

Similar criterions can be found for other dimensions: for the 1D case we have Courant-Friedrichs-Lewy stability criterion (replacing $\sqrt{2}$ by $\sqrt{1} = 1$), for the 3D case we replace $\sqrt{2}$ by $\sqrt{3}$, and so on.

This criterion applies to both the evolution of I_1 and the evolution of U, since they have the same velocity (in absolute value). In other words, it is a stability condition to the whole system 14.

Finally, let us just state the general form of the discretization equation, for c of an arbitrary sign:

$$\frac{I^{n+1}_{i.j} - I^{n}_{i,j}}{\Delta t} = \begin{cases} c_{i,j}\sqrt{I^2_{x_{b_m}} + I^2_{x_{f_M}} + I^2_{y_{b_m}} + I^2_{y_{f_M}}} & , c_{i,j}>0 \\ c_{i,j}\sqrt{I^2_{x_{b_M}} + I^2_{x_{f_m}} + I^2_{y_{b_M}} + I^2_{y_{f_m}}} & , c_{i,j}<0 \end{cases}$$

where the subindexes denote:

- b: backward difference
- f: forward difference
- m: compute the minimum between the quantity and zero.
- M: compute the maximum between the quantity and zero.

3.6.2 Velocity always decreases in the MAC algorithm.

To prove that the velocity of the evolution in any given point is a decreasing function of time, we will assume the necessary smoothnes over I_1 and I_2 so as to compute derivatives.

For the image I_1 we have:

$$I_{1_t} = (I_2 - I_1)\,|\nabla I_1| \tag{27}$$

and since I_2 is constant over time:

$$I_{2_t} = 0 \tag{28}$$

Combining 27 and 28 we get:

$$(I_1 - I_2)_t = (I_2 - I_1)\,|\nabla I_1| \tag{29}$$

Multiplying both terms of 29 by $(I_1 - I_2)$:

$$\frac{1}{2}[(I_1 - I_2)^2]_t = -(I_1 - I_2)^2\,|\nabla I_1| \tag{30}$$

which implies that

$$|I_2 - I_1|_t \leq 0, \forall t \tag{31}$$

That is, the absolute value of the velocity is always decreasing as a function of time. This is another way to see that I_1 evolves towards I_2.

3.6.3 Choice of time step.

In practice, the spacing in the spatial domain is fixed by the distance between pixels, while the time step has to be set. Equations 31 and 26 give us a way of determining the greatest value of the time step Δt that will guarantee stability for the whole evolution (since we want the numerical approximation of the evolution to be stable and as fast as possible):

$$\Delta t < \frac{\Delta x}{\sqrt{N} C} \tag{32}$$

where $C = max_{i,j} \{ \| I_{1_0}(i,j) - I_2(i,j) \| \}$ and N is the dimension of the image. In this equation we have made use of the fact that, since the difference between I_1 and I_2 is ever decreasing, the smallest time step $\Delta t(n)$ of the evolution is $\Delta t(0)$. So equation 32 guarantees stability for the whole evolution.

Nevertheless, our numerical implementation uses a variable time step instead of a fixed one. The time step $\Delta t(n)$ is inexpensive to compute, and the speed of the algorithm increases greatly if we recompute Δt after each iteration, since $\Delta t(n)$ is ever increasing.

3.7 Existence and uniqueness of the flow and the convergence problem.

3.7.1 Existence and uniqueness of the flow.

In section 2.3 we mentioned that hyperbolic equations like 14 usually develop singularities in initially smooth solutions. Besides, when working with digital images it is necessary to deal with initially non-smooth solutions. From the point where a singularity appears or when the initial condition is not smooth the classical results on existence and uniqueness of the flow are no longer valid, and the Theory of Viscosity Solutions has to be used. This is a very delicate matter, specially in our case where we have a system of coupled PDE's. Besides the works in [9] and [7], there is not much formal analysis for active contours approaches in the literature. We should also need to prove the independence of the evolution with respect to the embedding function U, as was done in [10] for other flow equations and in [9] for the Geodesic Active Contours model. These are results that require deep knowledge of the Theory of Viscosity Solutions to be achieved, and we will try to address them in our future work. Nevertheless, the stability condition found in 3.6.3, the fact that velocity decreases proved in 3.6.2 and our experimental results lead us to think that in fact our algorithm is "well-posed". This shall be the subject of further study.

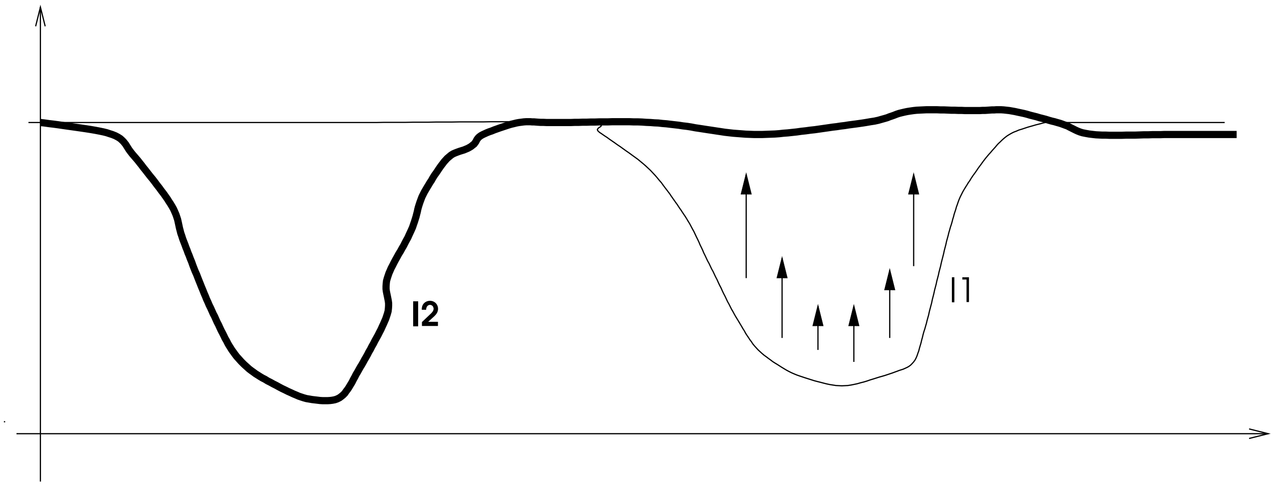

Figure 9: Illustration of the problem of non-convergence (see text).

3.7.2 The problem of non-convergence.

Another problem we have already mentioned is that the first equation in 14 does not guarantee that I_{1_0} will evolve into I_2. The simplest example is $I_{1_0} \equiv 0$, in which case $|\nabla I_1| = 0$, therefore $I_{1_t} \equiv 0$ and I_{1_0} does not change. But consider the case shown in figure 9.

Here, the "valley" in I_{1_0} will evolve to the "plain" that is I_2 over that region, since there $(I_2 - I_1) < 0$ and $\nabla I_1 \neq 0$ almost everywhere. But the "plain" on the left side of I_{1_0} will not evolve into the "valley" of I_2, because in that region $|\nabla I_1| \equiv 0$, therefore $I_{1_t} \equiv 0$ and I_{1_0} does not change. As a result, $lim_{t\to\infty} I_1 \neq I_2$. This problem would not happen if there were a slight overlapping between both "valleys", as shown in figure 10: there the deformation of I_1 propagates to the region where $|\nabla I_{1_0}| \equiv 0$, changing it and therefore making I_1 evolve in this region as well.

Since we have no formal proof concerning existence and uniqueness of the flow, we can not chacracterize the pairs of images (I_{1_0}, I_2) for which the evolution will not take the first into the second. But the studied examples suggest that non-convergence is reduced to the cases where I_{1_0} has a region with constant level α enclosing a region for whose boundary I_2 has also constant level α and different levels in its interior. This boundary where I_{1_0} and I_2 have the same level (therefore $(I_2 - I_1) = 0$) and flat gradient (therefore $|\nabla I_{1_0}| \equiv 0$) "isolates" the interior of the considered region from propagation of deformations from the outside. It is in this interior where we do not have convergence of I_{1_0} towards I_2. In other words, we do not have convergence when there are disjoint objects, objects that do not overlap, like those shown in figure 9.

This problem has different influence over segmentation and tracking applications than over interpolation and morphing applications.

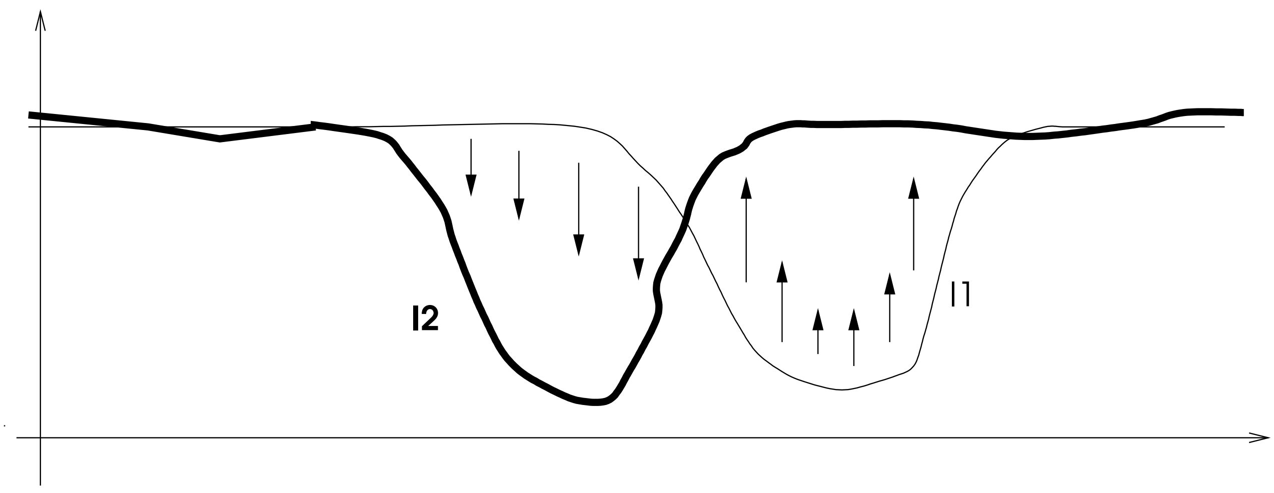

Figure 10: In this case, the image I_{1_0} converges to I_2.

3.7.3 Non-convergence in segmentation and tracking applications.

In the case of segmentation and tracking, since C_1 is over local edges (where $\nabla I_1 \neq 0$) the evolution will not "get stuck" for some region around C_1. Thus the non convergence problem will not influence the tracking operation, we think. One could argue that, since there may be regions where $(I_2 - lim_{t\to\infty} I_1) \neq 0$, for those regions the image U may diverge, having a constant limit velocity different than zero. For instance, if $U_0 > 0$ over a region where in the limit $(I_2 - I_1) < 0$, then it may happen (if $|\nabla U| \neq 0$) that $lim_{t\to\infty} U = -\infty$. This would not be a problem if it were not for the fact that our set C_2 is computed by contouring $lim_{t\to\infty} U$ at the zero-level, and the transition from $U_0 > 0$ to $lim_{t\to\infty} U = -\infty$ makes appear a "spurious" zero-level set. To avoid this, in our implementation we set that whenever $|\nabla I_1| = 0$ for a point (therefore $I_{1_t} = 0$ for that point) then the simultaneous rate of change in U for the same point is also set to zero: $U_t = 0$. This is a slight reformulation of 14 that can be expressed as:

$$\left\{ \begin{array}{c} I_{1_t} = (I_2 - I_1)\,|\nabla I_1| \\ U_t = \left\{ \begin{array}{cc} sg(\nabla I_1 \cdot \nabla I_2)(I_2 - I_1)\,|\nabla U|, & if\,|\nabla I_1| \neq 0 \\ 0 & else \end{array} \right. \end{array} \right. \tag{33}$$

Thus, for the regions of non-convergence of I_1 towards I_2 the surface U also stops its evolution.

3.7.4 Non-convergence in morphing and interpolation applications.

For interpolation and morphing applications this was not an issue since (as we will see) the tracking function U is not required. In these applications we will deal with binary images, and the problem of non-convergence appears as the non-deformation of (a subset of) C_1 towards C_2. This happens, as we mentioned above, when there is a subset of C_2 completely disjoint with C_1. To solve this problem, we can impose a minimum gradient $|\nabla I_1|$, say $G_m > 0$, so that whenever $|\nabla I_1| < G_m$ we set $I_t = (I_2 - I_1)G_m$. This guarantees the evolution of I_1 towards I_2, since the steady state is only reached when $(I_2 - I_1) \equiv 0$. And why do not we impose this condition for segmentation and tracking? Because it implies a time step of zero to ensure stability if we want to track C_1: this will be discussed in section 5.2.2.

4 Segmentation and Tracking.

4.1 Introduction.

Within the framework of Morphing Active Contours, segmentation and tracking are two closely related possible applications of our formulation. As we already mentioned in the past section, the study of methods for automated segmentation in sequences of biomedical images motivated the development of our MAC scheme. These segmentation applications required the determination of the boundary of some neurological object along several slices of it: this is analogous to the tracking of objects in a movie along several frames, so tracking applications of our scheme seemed straightforward. We will use the same set of equations from the MAC framework for segmentation and tracking, and that is why this section covers them both.

For segmentation we will specifically deal with the aforementioned sequences of neurobiological images, but just as an example, the possible use of our scheme being far more general.

4.2 Segmentation of neurobiological images.

4.2.1 The problem.

Understanding the rules commanding the operations in the nervous system is unquestionably linked to a precise knowledge of the neural circuits that ocurr at this level. Therefore, important amount of work has been devoted to explore the neuronal connections in different kinds of animals, including man. This is a laborious task that began one hundred years ago, defying the ingenuity and technical skills of numerous investigators. During several decades the main tools have been the silver impregnation methods to stain neurons and the light microscope to visualize them.

With the development of the electron microscope and its associated techniques a new frontier was reached; at present interneuron connections are studied with unexpected detail. However, the high resolution offered by the electron microscope is based on the observation of ultra-thin sections. This represents a serious handicap since the range of thickness of an electron-lucid section is only 1000 Å. Consequently, approximately 30-40 serial sections are needed to reconstruct a single interneuronal contact . The biologists have methods to obtain ordered ultra-thin sections (800 Åto 1000 Å) and observe this slices with a TEM [53]. From this images it is possible to reconstruct the celular structures.

For many years, 3D reconstruction of neural tissue using electron microscope images was an entirely manual operation. Recently, some efforts have been done to incorporate computer aided techniques to reduce this time consuming task. Carlbom et al. [6] have developed an interactive system which includes a digital "blink comparator" to allow manual registration of slices and deformable active contours for semiautomated cell segmentation. Montgomery et al. [39, 40] at the NASA-Ames Biocomputation Center have built the Reconstruction Of Serial Sections (ROSS) system which permits manual segmentation, registration and 3D reconstruction of slices.

Randall et al. [46, 47] devised a complete interactive system called Neuro3D which allows automatic enhancement and registration and manual segmentation of slices, plus 3D visualization of the reconstructed structures. This is the system we have acquired our images with, and basically it works as follows.

Firstly, a neurobiologist selects a neuron of interest from a sample of neural tissue. After the neuron is identified, it is marked via the injection of a color fluid. Then, a portion of the tissue is extracted, and after some proccessing it is cut into thin slices, observed and captured via the low-cost EM acquisition system of Neuro3D, consisting in a CCD camera placed on the binoculars of the microscope. After some artifacts are removed, noise reduced and non uniformity of the illumination compensated -automatically-, we obtain images as those shown in figure 11.

Then an automatic registration algorithm finds the euclidean transformation that better matches one image with the following, since the acquisition system occasionates displacements (rotations and traslations) between consecutive slices. This matching is performed with a novel scheme based on multirresolution, maximum correlation and maximum local entropy techniques. Then, the neurobiologist is ready to manually perform the segmentation of the sequence, drawing (with a Graphical User Interface) the boundaries of the neurological objects of interest that he sliced in the first place. Stacking in the right order and with the correct traslation+rotation these (2D) boundaries, a 3D model of the object is obtained, that can be rendered, rotated, cut, etc.

Segmentation in Neuro3D, as we said, is performed manually by a (trained) user, an expert. This choice was taken due to several facts, some already mentioned in 3.2:

1. The noisy nature of the images.
2. The wide range of types of images: an automatic segmentation algorithm that worked for all of this types seemed unlikely.
3. The presence in the same image of objects that seem of the same kind but actually are not: thus a non-trained user may choose all of them as "objects of interest" while an expert may not.

Being the point of an automatic segmentation algorithm, for this kind of images, to obtain the set of boundaries of certain neurological object along the slices so its 3D shape may be reconstructed and visualized, the aforementioned three points invalidate classical edge

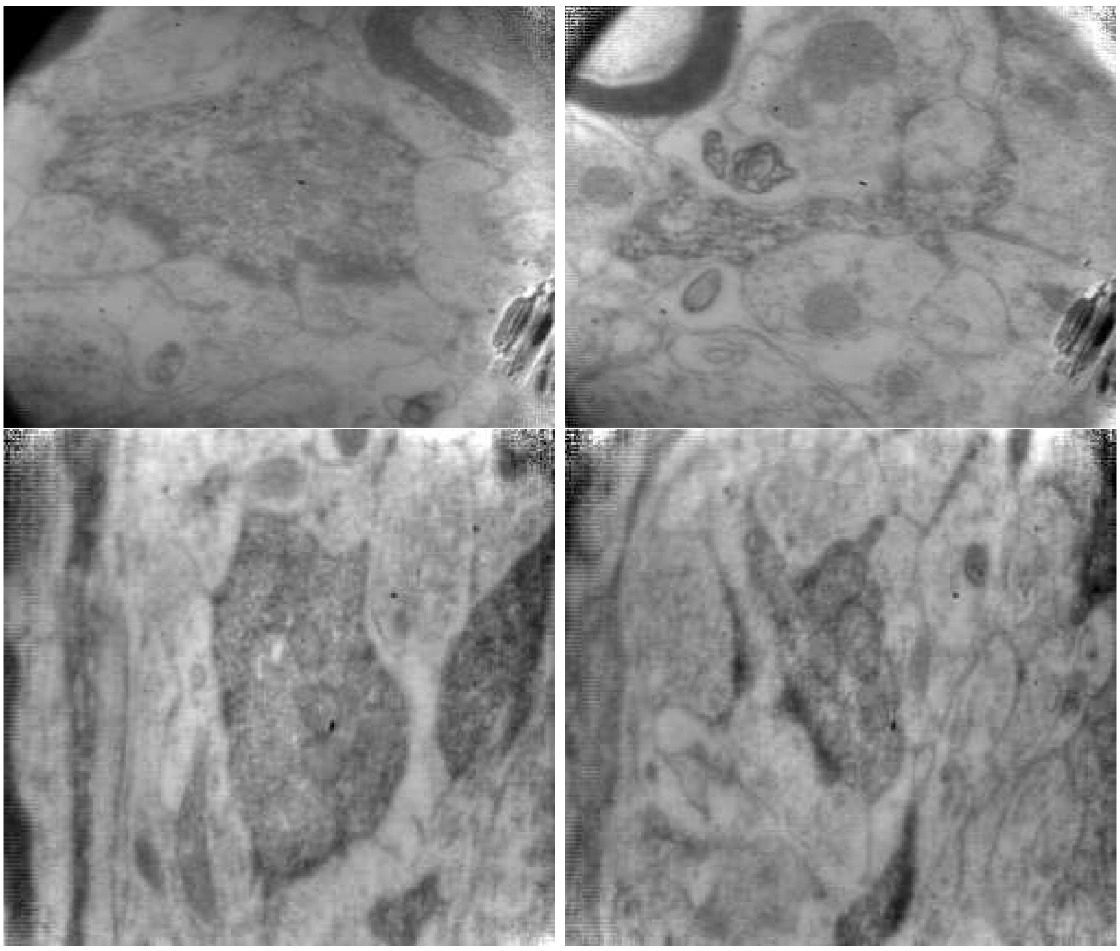

Figure 11: Neural tissue images obtained from a TEM with the Neuro3D platform. Each row shows non consecutive slices of the same volume.

detectors as tools for this purpose (here the difference between edge detection and segmentation is quite clear). In the best case they mark as edges all the boundaries of our objects of interest, but with them also a lot more edges all over the image.

To solve this problem we tried on our set of images a sophisticated edge-detector, based on the Geodesic Active Contours formulation [9].

4.2.2 Geodesic Active Contours.

In this subsection we will include excerpts from [9], where this formulation was introduced.

Since original work by Kass et al. [31], extensive research was done on "snakes" or active contour models for boundary detection. The classical approach is based on deforming an initial contour C_0 towards the boundary of the object to be detected. The deformation is obtained by trying to minimize a functional designed so that its (local) minimum is obtained at the boundary of the object:

$$E(C_0) = \alpha \int |C_0'(q)|^2 \, dq + \beta \int |C_0''(q)|^2 \, dq - \lambda \int |\nabla I(C_0(q)| \, dq$$

where α, β and λ are real positive constants and I the image in which we want to detect the objects' boundaries. The first two terms control the smoothness of the contours to be detected while the third term is responsible for attracting the contour towards the object in the image. Solving the problem of snakes amounts to finding, for a given set of constants α, β and λ, the curve C_0 that minimizes E. This formulation has two main drawbacks:

- it can not directly deal with changes in topology: the topology of the initial curve will be the same as the one of the, possibly wrong, final curve.
- it is not intrinsic: the energy functional to be minimized depends on the parameterization of the curve and it is not directly related to the objects geometry.

Novel geometric models of active contours were simultaneously proposed by Caselles et al. [7] and by Malladi et al. [37, 38]. These models are based on the theory of curve evolution and geomtric flows. In these active contours models, the curve is propagating (deforming) by means of a velocity that contains two terms as well, one related to the regularity of the curve and the other shrinking or expanding it towards the boundary:

$$u_t = g(I)(c + \kappa) \, |\nabla u|$$

where I is the image, u is a surface whose zero-level set is the deforming curve C, g is monotonically decreasing as a function of $|\nabla I|$, c is a positive real constant and κ is the euclidean curvature of C.

The model is given by a geometric flow (PDE) based on mean curvature motion. This model is motivated by a curve evolution approach and not an energy minimization one, and allows automatic changes in the topology when implemented using Osher and Sethian's algorithm.

Caselles et al. [9] improved this model with the Geodesic Active Contours formulation:

- it includes a new component in the curve velocity, based on image information, that helps to accurately track boundaries with high gradient variation, including small gaps, a task that was difficult to achieve with the previous models.
- in this framework, boundary detection can be considered equivalent to finding a curve of minimal weighted length, a geodesic curve in a Riemannian space with a metric derived from the image content: thus, the solution to this problem is intrinsic to the geometry, does not depend on a particular parameterization of the initial curve.

The Geodesic Active Contours evolution equation is:

$$u_t = g(I)(c+\kappa)\,|\nabla u| + \nabla u \cdot \nabla g \tag{34}$$

which means that the level sets of u move according to:

$$C_t = g(I)(c+\kappa)\overrightarrow{N} - (\nabla g \cdot \overrightarrow{N})\overrightarrow{N}$$

The c term moves the curve outwards or inwards, the κ term smoothes the curve, the function g tends to stop the evolution when the curve is near an edge, and the last term attracts the curve towards the boundaries of the objects, being of special help when this boundary has high variations on its gradient values.

4.2.3 Application of Geodesic Active Contours to segmentation of sequences of neurobiological images.

In the Geodesic Active Contours formulation, an initial curve that surrounds the objects of interest (or is in the interior of them) deforms with an inward (or outward) motion keeping a certain degree of smoothness and stops near high gradient regions. For our application several problems appear with this scheme, and all of them come from the dependence of the evolution equation's success on the good choice of the function $g(I)$.

The stopping function $g(I)$ takes values in the range $[0,1]$, where $g=0$ corresponds to ideal edges: thus, the evolution stops when the curve reaches an ideal edge. Typical stopping functions are:

$$g = \frac{1}{1 + \left|\nabla \widehat{I}\right|^{p}} \tag{35}$$

$$g = e^{-\alpha\left|\nabla \widehat{I}\right|} \tag{36}$$

where $\widehat{I}$ is a smoothed version of I and $p = 1$ or 2, and α is a constant.

But we mentioned above that our images may have several objects with the same appearance and only some of them being objects of interest. Thus, the abovementioned "simple" stopping functions take values close to zero near the boundaries we want to detect but also near many other boundaries. Figure 12 shows a neurobiological image, the same image with the boundary of interest drawn in black, and three stopping functions obtained from the original image with equations 35 and 36. For this image and these stopping functions, any initial curve that is not very close to the boundary of interest will certainly stop at some local minima, away from the boundary we want it to detect.

The direction of propagation (inward or outward) is given by the sign of the constant c in 34: if $c > 0$ we get inward evolution, and in the other case it is outward. The value of this constant is *heuristic*: if it is too small (in absolute value) the evolution may stop at local minima, while if it is too large (in absolute value) the evolution may not stop at the boundaries since the boundaries, being not ideal, do not have a stopping-function value of zero. And something very important: if the initial curve is neither interior nor exterior to the boundaries of interest, if it crosses them at some points, the evolution does not behave properly. Therefore it is essential to ensure that the initial curve is far enough from the boundaries of interest so as not to intersect them. But too large a distance from the initial curve to the boundaries may make the evolution stop at local minima, as we mentioned earlier.

If we allow the user-expert to draw the initial curve for every slice, a raw sketch close to the boundary of interest (and not intersecting it), then the problem is solved. But if we want automatic segmentation of the sequence, then we get into a deadlock: the initial curve must be far enough from a boundary we do not know and also close enough to it. This also implies that, even if we have a very good estimate of the boundary of interest, we will not be able to use this information as an initial curve for the next image since we must "shrink" or "expand" this curve or set of curves so as to sorround or be sorrounded by the boundaries of interest of the next image. And the amount of "shrinkage" or "expanding" depends on an estimation of the variation of the boundaries of interest from one image to the next: if the boundaries change much from one slice to the next, then we need to shrink or expand our curves a lot if we want to avoid intersections with the (unknown) boundaries of the next image. Also, the success of the evolution in detecting these boundaries depends on a good choice of the velocity c, as we mentioned above.

In other words: *automatic* segmentation of these sequences using Geodesic Active Contours

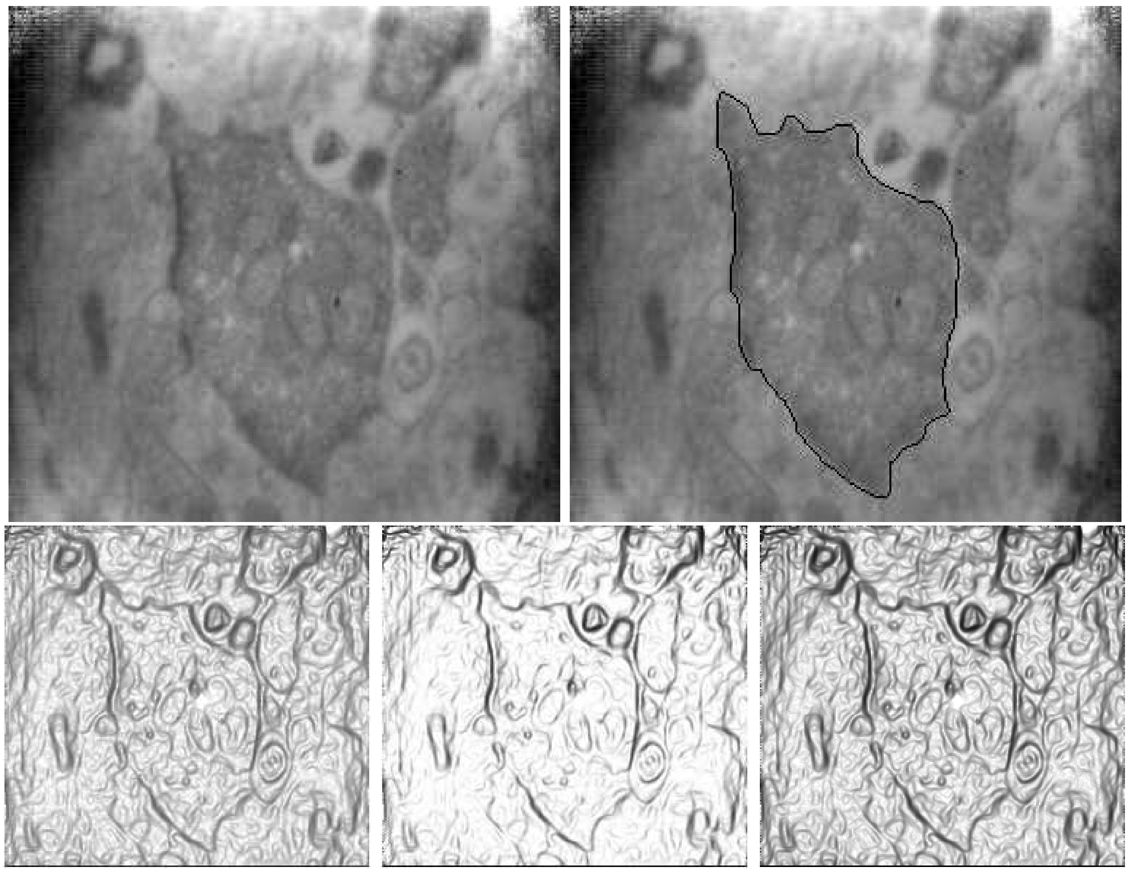

Figure 12: Top row: a neurological TEM image (left) and the same image with manual segmentation superimposed (right). Bottom row: three stopping functions; the one on the left is obtained from equation 35 with $p = 1$, the middle one from the same equation with $p = 2$, and the right one is obtained from equation 36.

seems to be an impossible task, unless we have a very good *a priori* knowledge of the image characteristics. Or if we have a very good stopping function g, one that takes values close to zero only near the boundaries we want to detect.

4.2.4 Refining g.

We tried to refine the stopping function g by two different means: by incorporating some frequency information in the boundary detection, and by deriving some statistics from a user-defined boundary.

Frequency information. Segmentation with frequency information is used when the boundaries do not represent a transition in gradient but a transition in the frequency domain (be it in phase or/and amplitude). This is tipically the case with texture segmentation. In the Gabor formulation, a family of filters (in the spatial domain but with variables concerning radial frequency and angular orientation in the frequency domain) is convolved with the "textured" image obtaining a single image which is the normalized power modulus of the filter ensemble. Each filter is (almost) localized both in the spatial and the frequency domain, and thus we get a projection of our image onto a basis that tiles the spatial frequency plane [34]. Criteria for selection of 2D Gabor filters coefficients are given in [34, 19], and for 3D Gabor filters in [5]. In a similar approach, *image pyramids* is a transform in wich the original image is decomposed into a set of subbands that are computed by convolving and subsampling: this yields a set of subband images of different sizes (hence the name pyramid) that correspond to different frequency bands. Typical pyramid schemes are the Laplacian Pyramid (with radially simmetric basis functions) and the Steerable Pyramid (that divides each frequency band into different orientation bands) [29, 52].

None of these schemes showed any amount of improvement for the sequences we worked with unless we did a manual fine-tuning of all of the parameters involved, and even then the results were not excellent. We did not go further in this line of work, prompted by the unpromising results; maybe better results are obtained for some other sequences where boundaries separate regions with clearer differences in the fequency domain.

Statistics from a user defined boundary. Our approach here was to compute a set of measures from a user-defined boundary and, again asuming that the corresponding boundary in the follwing image of the sequence will not be too different from this one, to try to tailor g (in the following image) to incorporate these measures. In practice, we computed the length of the boundary and the average (G_m) and variance (σ) of $|\nabla I|$ along it. Thus, g did not have a monotonically decreasing dependence over $|\nabla I|$ but one as shown in figure 13: g is 1 everywhere but in a region of $n\sigma$ size around G_m.

Some results of this approach are shown in figure 14. In its top row, two consecutive images are shown; the one on the left shows a user-drawn boundary. This boundary was used to compute the statistics to define the stopping function for the second image that is

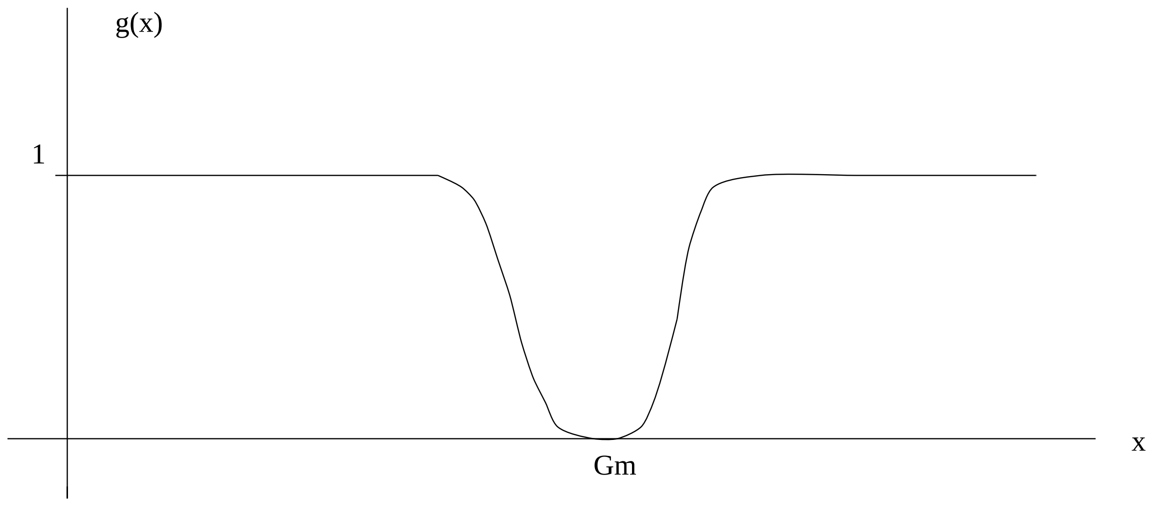

Figure 13: Plot of the stopping function g vs. $|\nabla I|$, in the case when g is computed *statistically* (see text).

shown below the top row. As we see, the boundaries of interest are better detected by g at the price of enhancing also other boundaries that lie too close to the one we are trying to detect. Regrettably, this does not seem the adequate approach either.

4.2.5 Limitations of Geodesic Active Contours in this application.

Recapping, the Geodesic Active Contour formulation requires the initial curve to sorround or be sorrounded by the boundaries to be detected. Since we do not know the exact location of these boundaries, an estimate must be made and the initial curve placed at a "safety" distance. For the type of images we have, this distance makes the evolution to stop at local minima (local maxima of the gradient), since the stopping function g can not be made to take values close to zero only near the boundaries of interest. A semi-automatic approach may solve all these problems, by requiring the user to specify the initial curve for each image.

4.2.6 Morphing Active Contours.

For this application, segmentation of sequences of 2D images, the general formulation of equation 14 can be written as:

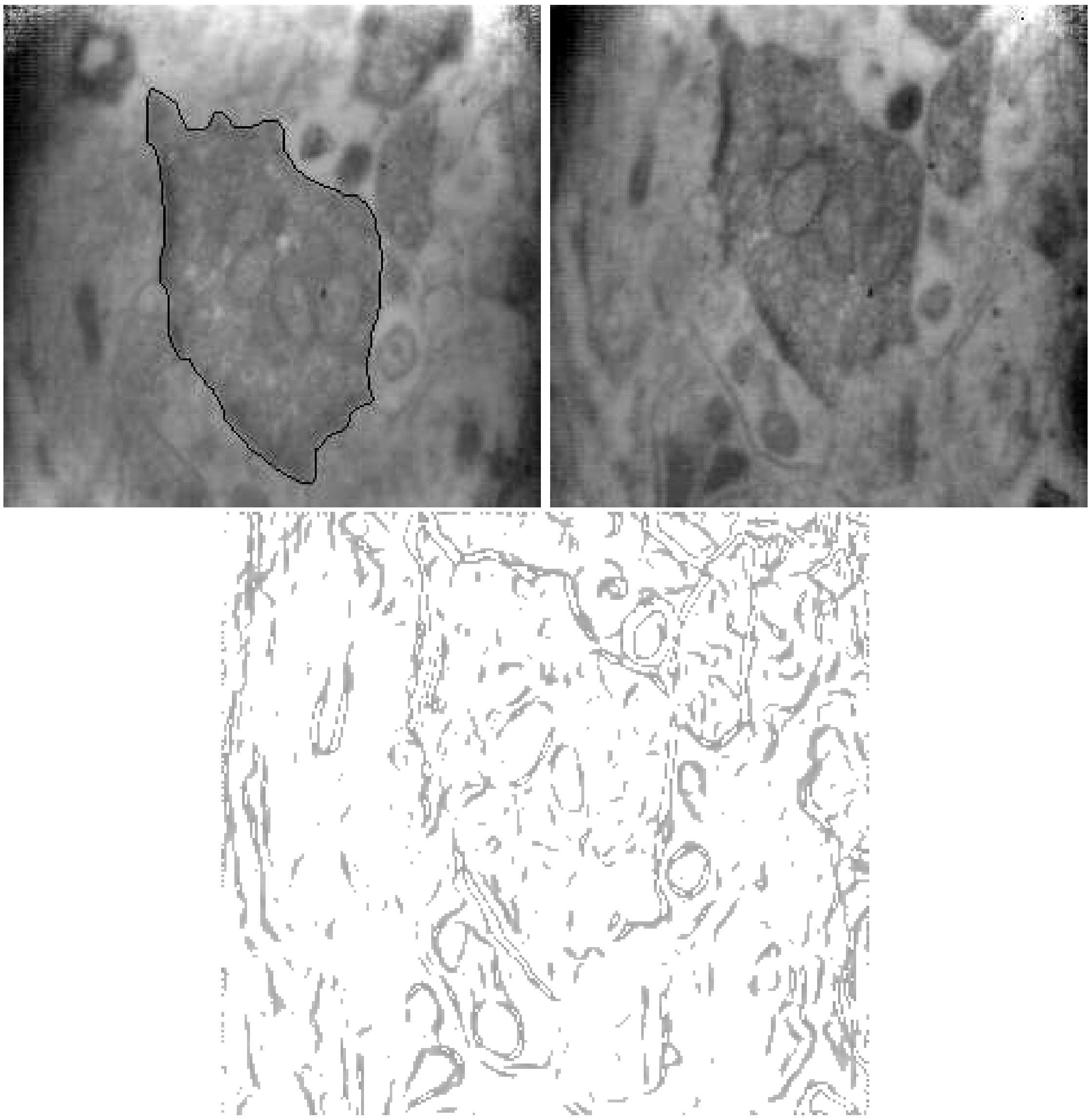

Figure 14: Top row: image of a neurological slice with superimposed (manual) segmentation (left), and the following slice, unsegmented (right). Bottom: stopping function for the second image of the top row, computed using statistics from the segmentation of the first image.

$$\left\{\begin{array}{l} I_{1_t}(x,y,t) = (I_2(x,y) - I_1(x,y,t))\,|\nabla I_1(x,y,t)| \\ U_t(x,y,t) = \left\{\begin{array}{ll} sg(\nabla I_1(x,y,t)\cdot\nabla U(x,y,t))(I_2(x,y) - I_1(x,y,t))\,|\nabla U(x,y,t)| & if\,|\nabla I_1| \neq 0 \\ 0 & else \end{array}\right. \end{array}\right. \tag{37}$$

with initial conditions

$$\left\{\begin{array}{l} I_1(x,y,0) = I_{1_0}(x,y) \\ U(x,y,0) = U_0(x,y) \end{array}\right. \tag{38}$$

These are the steps of the MAC algorithm as applied to automatic segmentation of sequences of neurobiological images:

1. We have a sequence of N images of the same size. We assume the images to be already matched.
2. Anisotropic diffusion (2D) of the images is performed.
3. The user supplies the boundaries of interest of the first image, the set C_{1_0} of (closed) curves, and the tracking surface U_0 is constructed.
4. For each pair of images (I_n, I_{n+1}), $1 \leq n \leq N-1$,with the set of curves C_{n_0} for image I_n, we want to find the set of correspondent curves $C_{(n+1)_0}$ on I_{n+1}:
   (a) The histogram of I_{n+1} is made equal to that of I_n.
   (b) The evolution equation 37 is run for a certain amount of steps.
   (c) The curve set $C_{(n+1)_0}$ is defined to be the zero-level set of U, and used as initial condition for the next evolution.

Let us look at these steps in closer detail.

1. The assumption of correct matching between consecutive images is not really a requirement of the algorithm. But a good registration implies that the distance between I_n and I_{n+1} (and also between C_n and C_{n+1}) will be minimal, therefore the evolution will require fewer steps to get close enough to the steady state. And since the registration can be achieved without previous knowledge of the curves of interest (see [46, 47]) there is no point in not using this information. Besides, unregistered slices may be too far apart and cause the evolution to stop away from the desired boundaries, when the "similarity" hypothesis is violated.

   As a future line of work, we could investigate the possibility of using the MAC framework for the aforementioned registration process, defining a feedback system with evolution equations that achieve simultaneously registration and segmentation.

2. Anisotropic diffusion is performed in order to reduce noise while preserving the edges. This is a must since our sequences are comprised from very noisy images. But in spite of the fact that we do not want the boundaries to be blurred while denoising is performed, too steep edges are very difficult to draw by hand so a little smoothing may be helpful. We will comment on boundary drawing by the user in the next point, but remembering from section 3.4.2 that the initial curve needs to be placed near local maxima of the gradient for the evolution to be successful, if the edges are too steep then it will be difficult for the user to manually draw the boundaries over these edges without "falling" to one side or the other of the edge. But instead of blurring the image with isotropic diffusion, what we do is to perform a lot of anisotropic diffusion, as a compromise between edge preserving and smoothing. Tipically, we iterate 20 steps with a time step of 0.2: these are the values used for the examples below.

3. As mentioned above, a good selection of the initial curve C_{1_0} is essential for the succesful performance of the algorithm. Since it has to be specified by the user, we assume that is as correct as it can be. But if the curve corresponds to very steep edges and is placed just a little inside or a little outside the real boundary, then the evolution may fail completely (the curve being "trapped" in a region of flat gradient) or lose track along portions of its length. The same problem may happen if later on the evolution is stopped too early and the resulting estimated set of boundaries C_{n_0} is near but not close enough to the local maxima of the gradient.

   For the initial curve, if the user draws it by hand (or rather with a GUI), some smoothing of the edges may be helpful. An alternative to this approach is the technique of geometric tracking with minimal path, devised in [12] and implemented for our case in [59]. This technique requires the user to identify just two points over the boundary, and the algorithm finds the path of minimal length (in a Riemannian metric) that conects both points. Therefore this method is more accurate and less error prone that the manual segmentation of the initial image. As for the initial surface U_0, it is convenient to make it smooth around C_{1_0}: a binary U_0 (with value -128.0 inside C_{1_0} and 128.0 outside it, for instance) may lose track of the boundaries in noisy images, and requires a more precise placement of the initial curve over the boundaries. A method for the construction of a smooth embedding function was given in section 2.4.3. It would also be convenient to ensure the smoothness of U_n, for every image in the sequence, by constructing a new surface U_n rather than using the last iteration of U_{n-1}.

4. This step is run $N-1$ times, once for every image and its following.

   (a) Histogram equalization is required to guarantee a real "similarity" between the images. As we mentioned in section 3.4.1, if I_n and I_{n+1} are both images of the same scene but with different illumination, say I_{n+1} is darker, then one would want C_{n_0} to stay put over the evolution but instead it shrinks, as it has negative normal velocity in every point. To avoid this, histogram equalization on I_{n+1} is performed, making this image have the same histogram as I_n. This is achieved as follows:

- The histograms H_n of I_n and H_{n+1} of I_{n+1} are computed.
- Then we compute the integrals of the histograms: IH_n,IH_{n+1}.
- We replace every gray-level g in I_{n+1} with another value, $\widehat{g}$, computed thus: $\widehat{g} = IH_n^{-1}(IH_{n+1}(g))$. The inversion of IH_n is always possible assuming that it is a monotonically increasing function; this is not always true (IH_n is always increasing because H_n is always positive, but not necessarily monotonically increasing), in wich case the value $\widehat{g}$ has to be chosen among several in an interval. This is a transformation by wich we obtain a new (equalized) image $\widehat{I}_{n+1}$ such that the integral of its histogram $\widehat{IH}_{n+1}$ is identical to IH_n: therefore, the histograms of I_n and $\widehat{I}_{n+1}$ are also identical.

This method is ilustrated in figure 15. This figure shows, on the left, an image; in the middle, the consecutive image with changes in the illumination; the image on the right is obtained by equalizing the histogram of the second image so as to make it the same as that of the first image. Another approach for histogram modification can be seen in [50], where the histogram can be modified to achieve any given distribution via an evolution equation.

(b) In the Curve Evolution approach the curve follows an evolution by expansion in the normal direction yet it never comes to a complete stop, and heuristic stopping procedures are used to switch off the evolution process. In our case, since the difference between I_n and I_{n+1} is always diminishing, as we proved in 3.6.2, we could choose to terminate the evolution once the maximum pixel difference among both images is lesser than a certain threshold value. Nevertheless, the problem of non convergence does not allow us to guarantee that threshold shall be reached, since in some regions the limit of I_n may be different to I_{n+1}, as we saw in section 3.7.2. We could compute this "distance" between both images as the maximum pixel difference not in the whole image but in a region (a "band") surrounding the evolving curve C_n, where we suppose that there will not be problems of non convergence, as mentioned in section 3.7.3. In practice, we set a heuristic number of iterations for the evolution: this number should depend on the type of images we deal with, but for these sequences a number of 200 iterations seems to be a safe choice. In tracking, where consecutive images are "closer", a number of 30 steps may be enough, as it is in interpolation and morphing.

As a future line of work, we think that after the evolution stops there should be a stage of "correction" of the set of curves C_{n+1} found so as to make it lie exactly on the local maxima of the gradient: this procedure would increase the accuracy of the algorithm. Since segmentation is performed automatically, our approach does not allow for self-corrections and errors add up, therefore it may

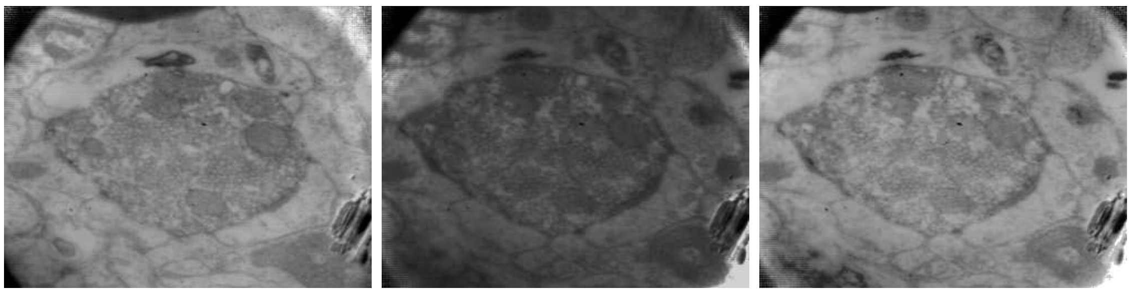

Figure 15: Histogram equalization. The image on the right is obtained by equalizing the histogram of the middle image so as to make it the same as that of the first image.

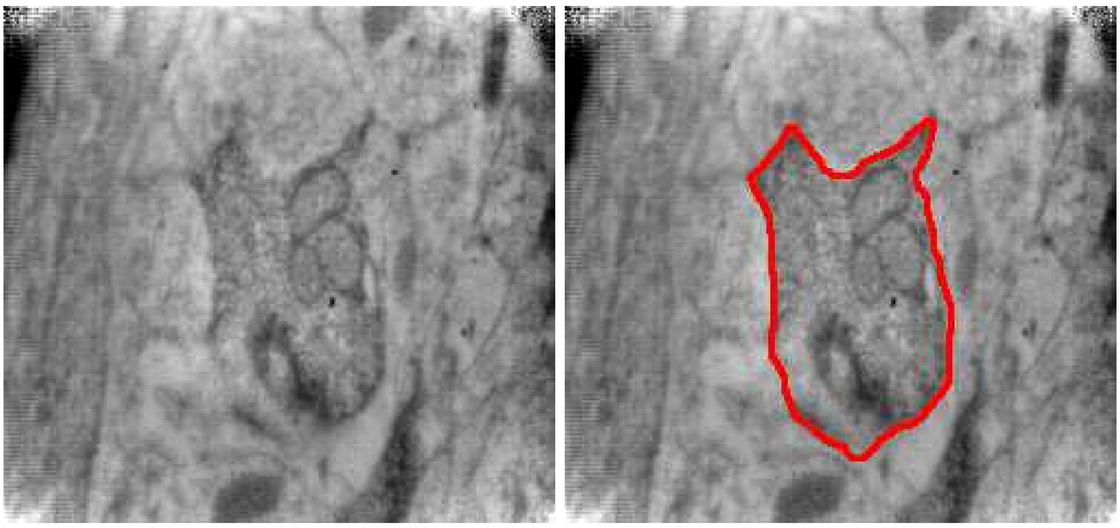

Figure 16: Left: initial image of a sequence. Right: the same image with manual segmentation superimposed.

be convenient to reinitialize after a certain amount of slices. The automatic "correction" procedure would increase the number of consecutive slices that the algorithm can handle.

4.2.7 Examples of segmentation with MAC.

Figure 16 shows, on the left, the initial image of a sequence, and on the right the same image with its correspondent manual segmentation superimposed. This curve is used as the initial condition for the automatic segmentation process, which is performed for the following twelve images of the sequence.

Results are shown by superimposing the boundaries found by the MAC algorithm on the original greyscale images, but the iterations were performed on smoothed and matched images, with equalized histograms: each image has been smoothed with 20 steps of anisotropic diffusion, with time step of 0.2. The number of iterations of the algorithm was chosen to be 200. The manual segmentation on the initial image was obtained with a very simple drawing program. Notice how the boundaries separate regions of different grey value, but in the interior of the object of interest there are small areas of gray values similar to those of the exterior. This is why some “holes” in the object appear, as the evolving curve passes over these areas on its way to the boundaries. Conversely, near the boundary of the object of interest but on its exterior there are regions of similar gray values to those of the interior: hence, if any initial curve travels inward over these areas, it may mark them as pertaining to the object. Apart from these problems, we see that the overall performance is quite good: see figures 17 and 18. A final correction step performed by the biologist with the help of a simple GUI would allow to eliminate the spurious boundaries.

Figures 19 to 22 show the performance of the algorithm for two other sequences of neurological images. For these examples the initial curve was drawn using the aforementioned technique of geometric tracking with minimal path, devised in [12] and implemented for our case in [59]. The evolution was performed on matched, scaled (50%) and smoothed (30 steps of anisotropic diffusion with a time step of 0.2) images, for 300 iterations. The scaling greatly increases the speed and does not seem to degrade the results. For the first, 15-slice sequence we reinitialized the evolution at the 8th slice.

4.3 Tracking.

It has been mentioned throughout this work that tracking of moving objects in video sequences can be seen as automatic segmentation of objects in a sequence of 2D images, the case studied in the previous subsection. Therefore, in this subsection we will not insist on previous concepts and concentrate on the examples and the (few) differences with the abovementioned segmentation formulation.

First of all, the evolution equation for tracking in the MAC framework will be equation 37, with initial conditions as given in equation 38. Here C_{1_0} corresponds to the boundary of the object to track. The algorithm is performed in the same steps as in the segmentation case: anisotropic diffusion for edge-preserving noise removal, histogram equalization to avoid illumination problems (that may be more apparent in video sequences), and so on. Let us just note a couple of things:

- As mentioned in the previous subsection, the (heuristic) number of iterations we allow for the evolution before we stop it may be smaller for video sequences. In most of the cases, the objects to track along the sequence move with a speed considerably slower than the equivalent “speed” of neurons boundaries along a sequence of TEM images: this is closely related with the interpolation problem in those biological sequences, and will be addressed in section 5.2. For the movies we have worked

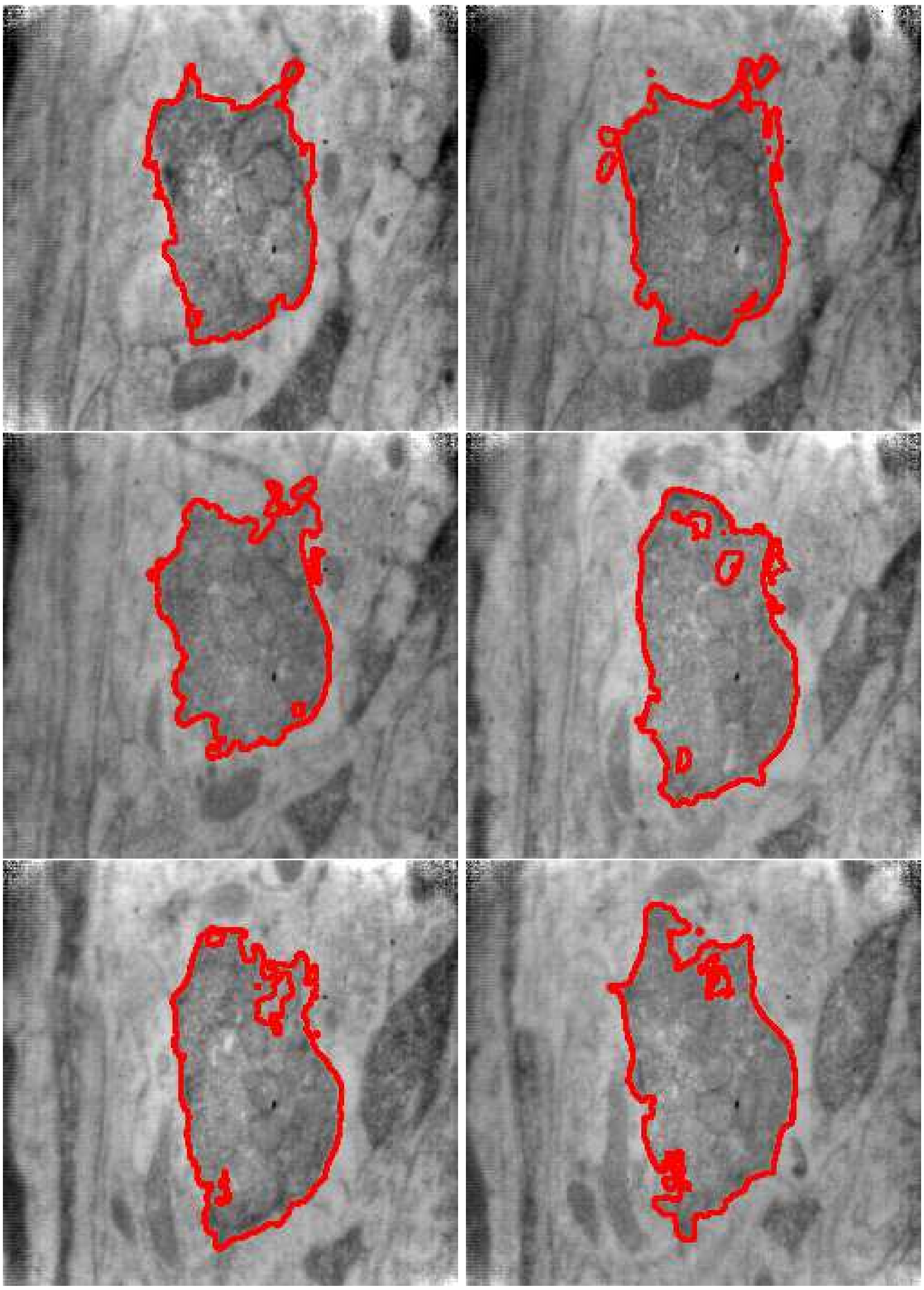

Figure 17: Automatic segmentation on a sequence of twelve consecutive images: this figure shows images 0 to 5. The boundaries found by the MAC algorithm are shown superimposed on the original images.

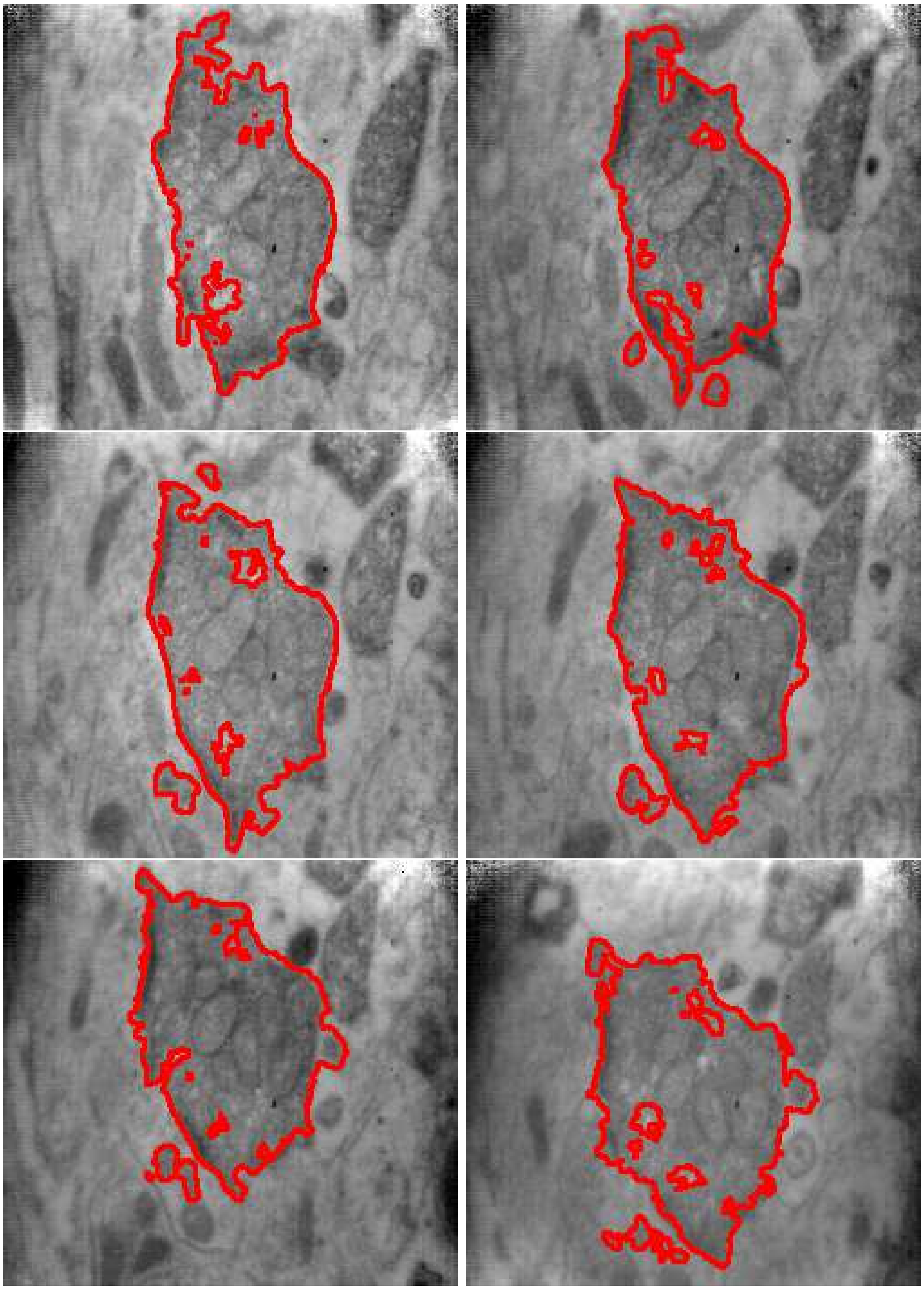

Figure 18: Automatic segmentation on a sequence of twelve consecutive images: this figure shows images 6 to 12. The boundaries found by the MAC algorithm are shown superimposed on the original images.

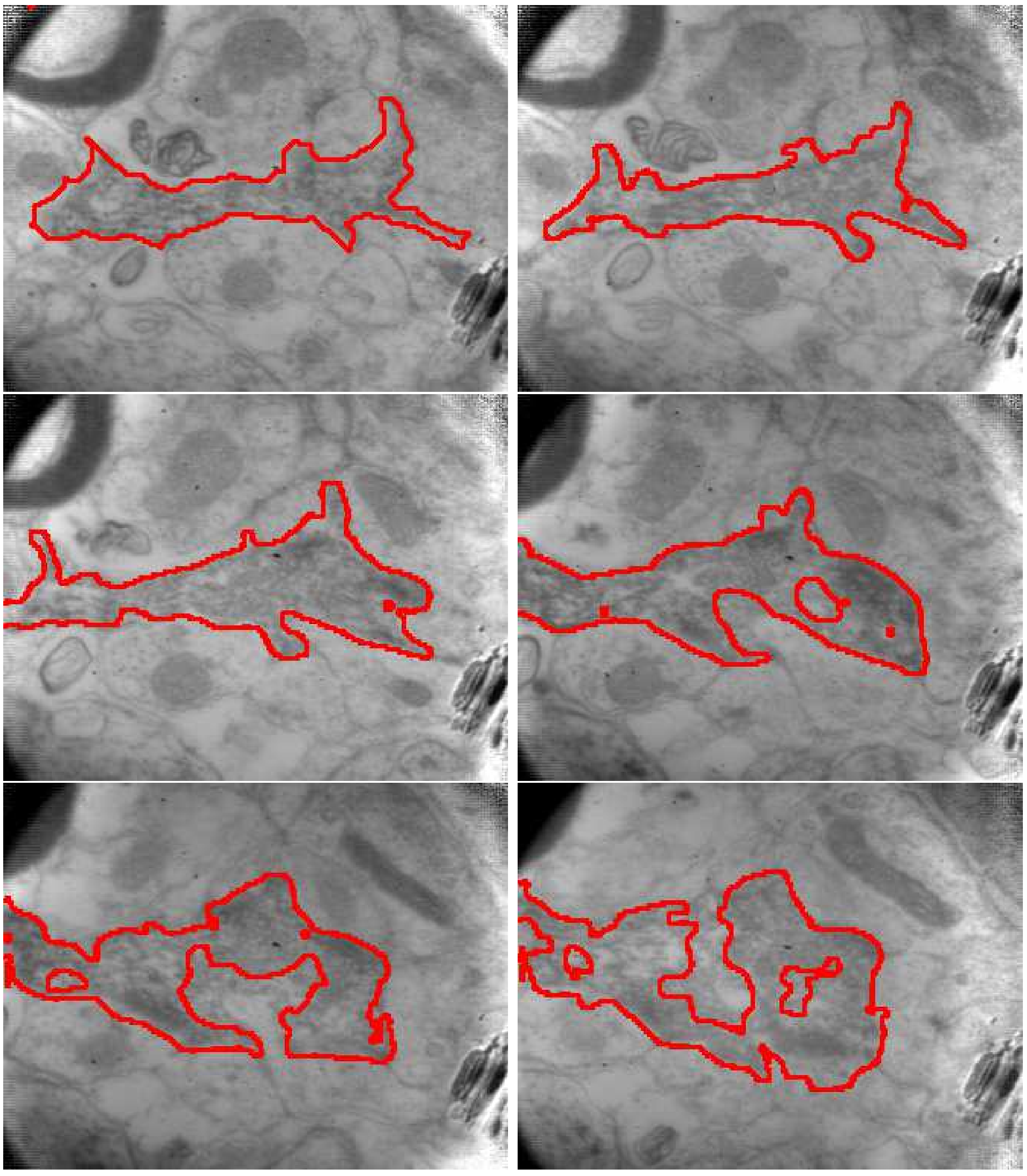

Figure 19: Automatic segmentation on a sequence of fifteen consecutive images: this figure shows images1 to 6. The boundaries found by the MAC algorithm are shown superimposed on the original images.

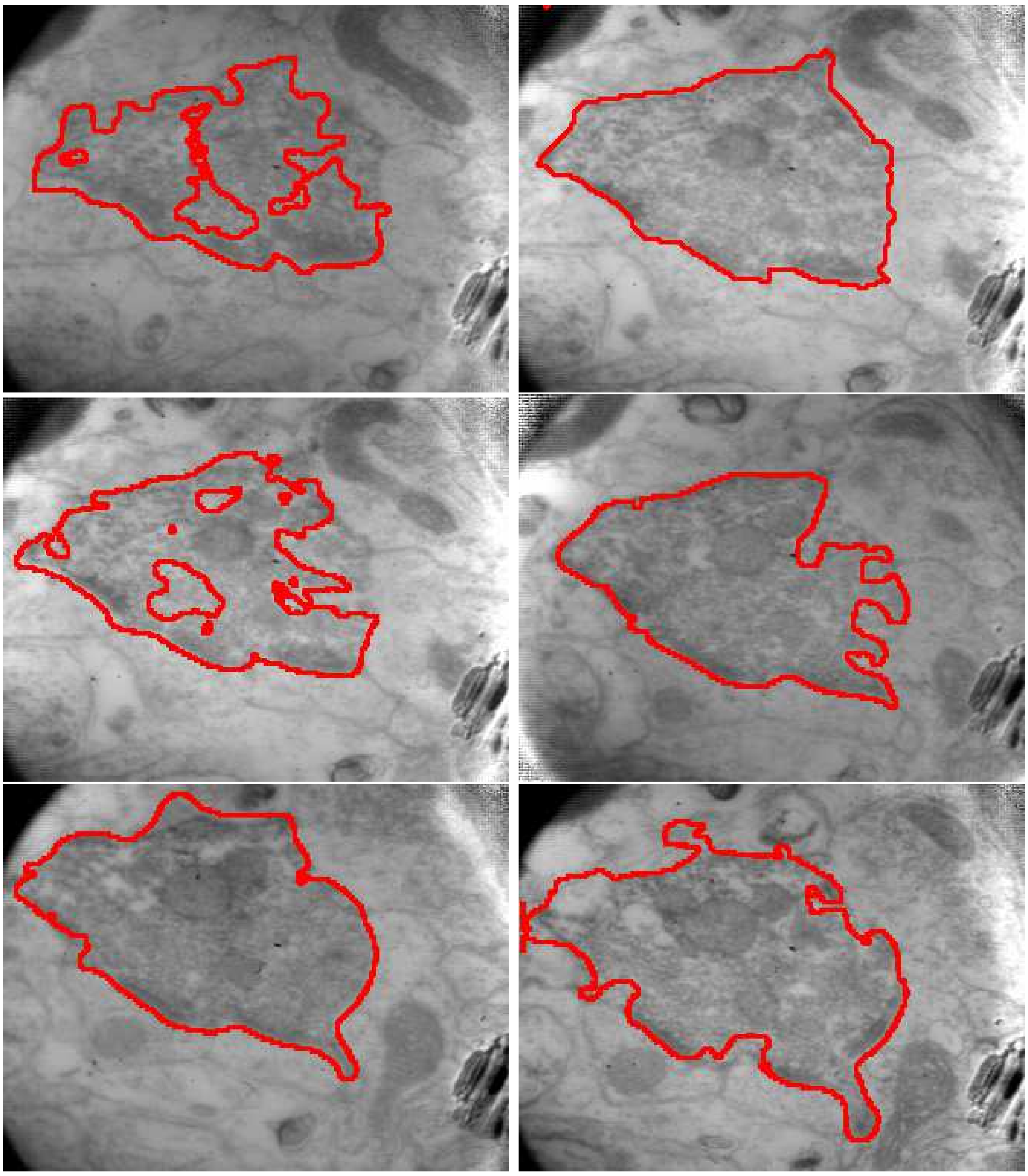

Figure 20: Automatic segmentation on a sequence of fifteen consecutive images: this figure shows images 7 to 11; the evolution was reinitialized on image 8, the third image of this figure. The boundaries found by the MAC algorithm are shown superimposed on the original images.

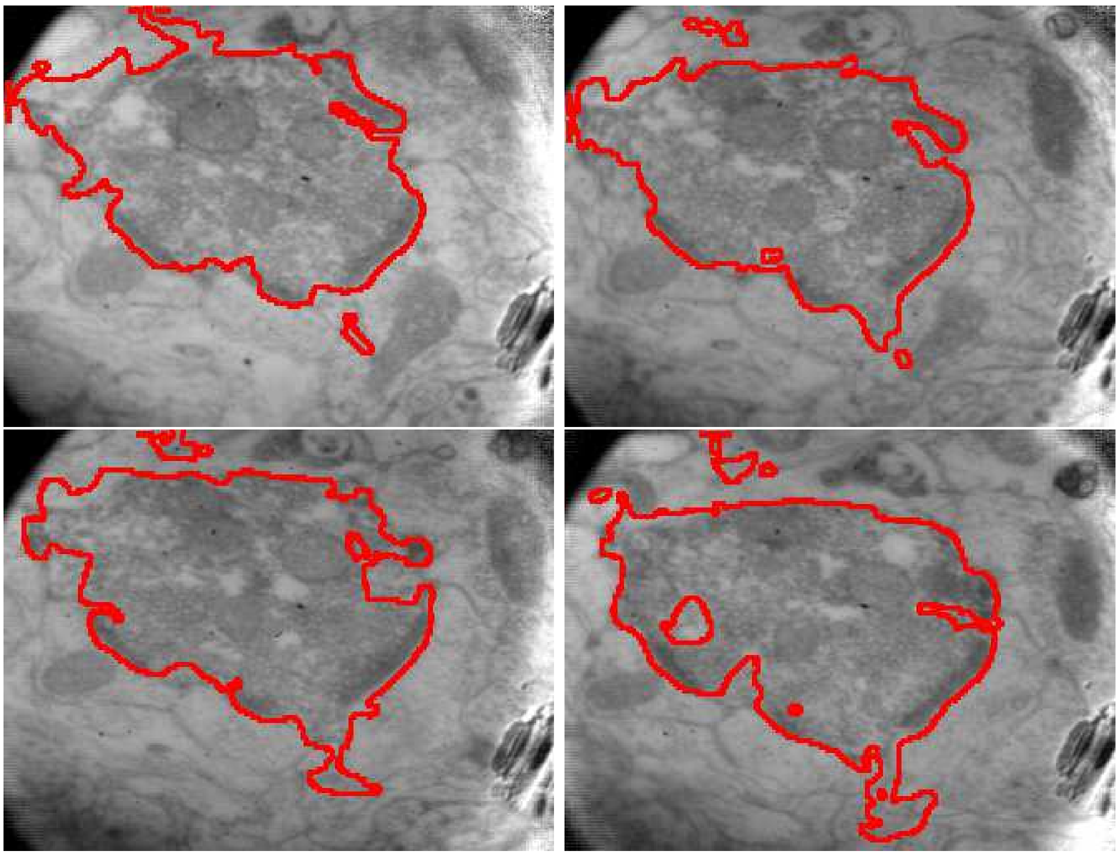

Figure 21: Automatic segmentation on a sequence of fifteen consecutive images: this figure shows images12 to 15. The boundaries found by the MAC algorithm are shown superimposed on the original images.

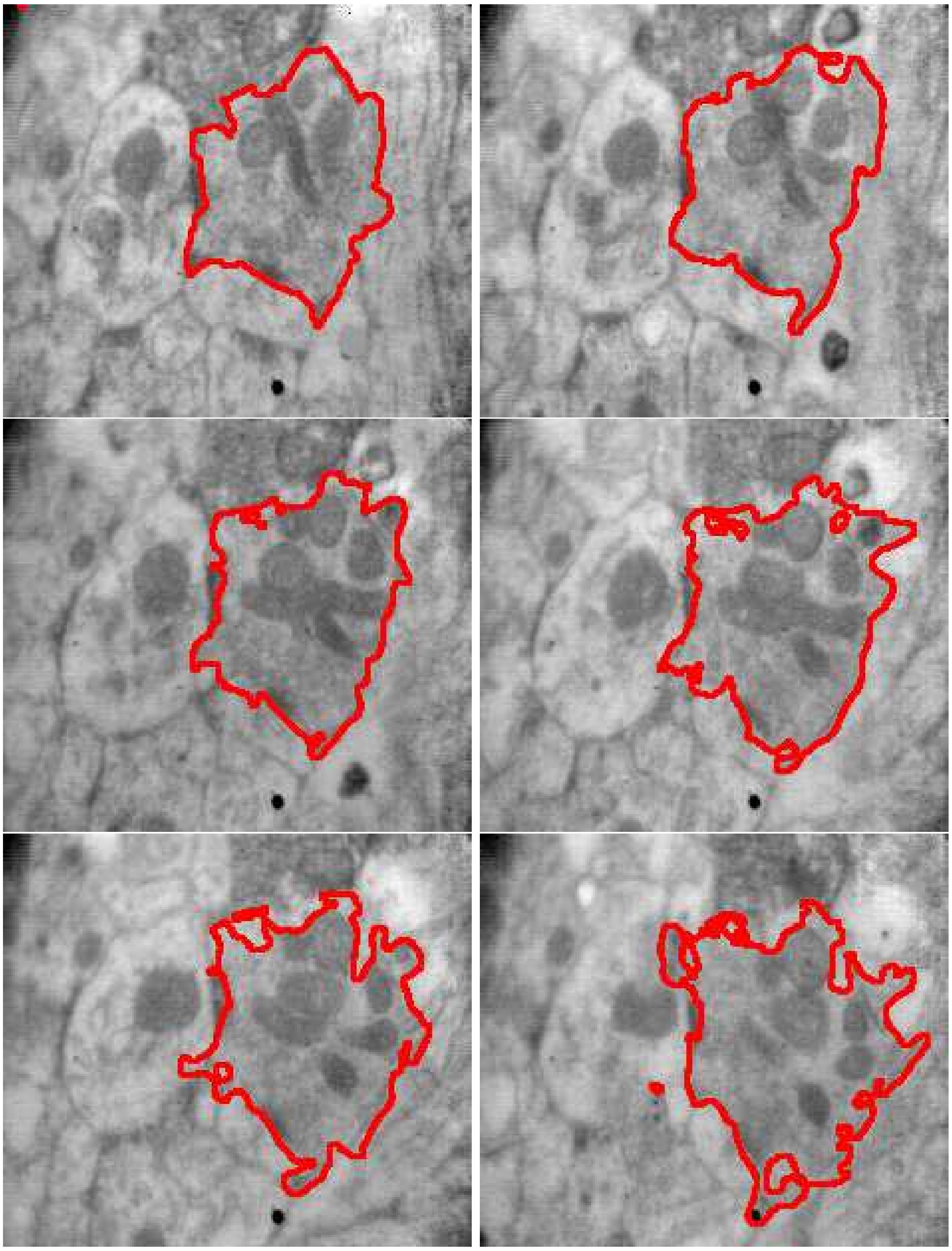

Figure 22: Automatic segmentation on a sequence of six consecutive images. The boundaries found by the MAC algorithm are shown superimposed on the original images.

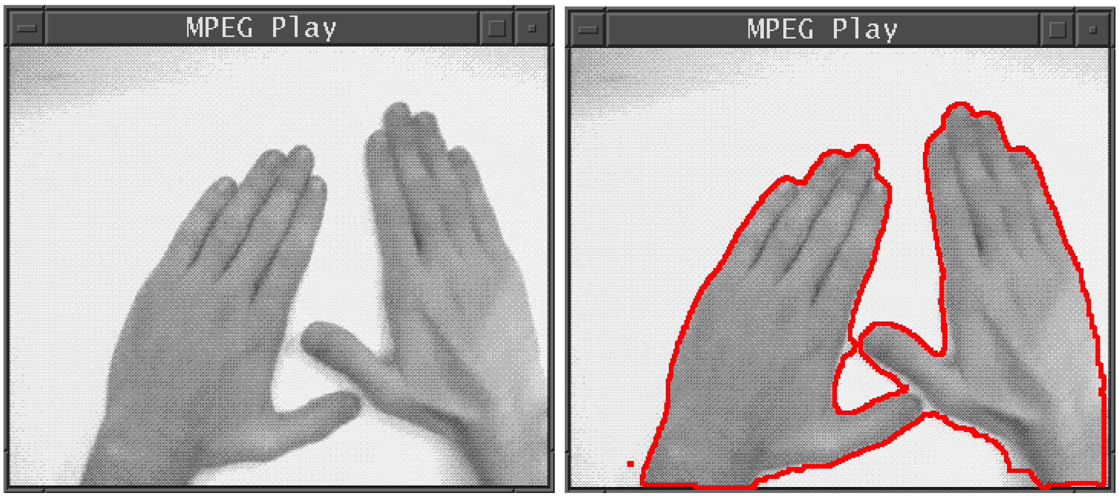

Figure 23: Left: initial image of a sequence. Right: the same image with manual segmentation superimposed.

with (25 frames/second) a number of 30 iterations seems to be adequate. Another approach could be to do a "subsample" of the frames, picking one frame every n, but then the number of iterations should probably increase (this method would improve speed if it required a number of iterations lesser than 30 times n).

- In our neurobiological sequences the boundaries of the object of interest separate a darker interior from a lighter exterior, tipically. That is not the case in general video sequences, where the object of interest may be, near its boundary, darker than the exterior in some places but lighter than the exterior in others. Here again the term $sg(\nabla I_1 \cdot \nabla U)$ allows the curve C_n to "find its way" in the right direction: if this term were missing, the evolving curve would mistake darker exterior for interior, or viceversa, causing the evolution to lose track.

4.3.1 Examples of tracking with MAC.

Figure 23 shows, on the left, the initial image of a sequence, and on the right the same image with its correspondent manual segmentation superimposed. This curve is used as the initial condition for the tracking process, which is performed for the following twenty four images of the sequence.

Results are shown by superimposing the boundaries found by the MAC algorithm on the original greyscale images, but the iterations were performed on smoothed images, with equalized histograms: each image has been smoothed with 20 steps of anisotropic diffusion, with time step of 0.2. The number of iterations was chosen to be 50. The manual segmentation on the initial image was obtained with a very simple drawing program. See the results in figures 24 and 25.

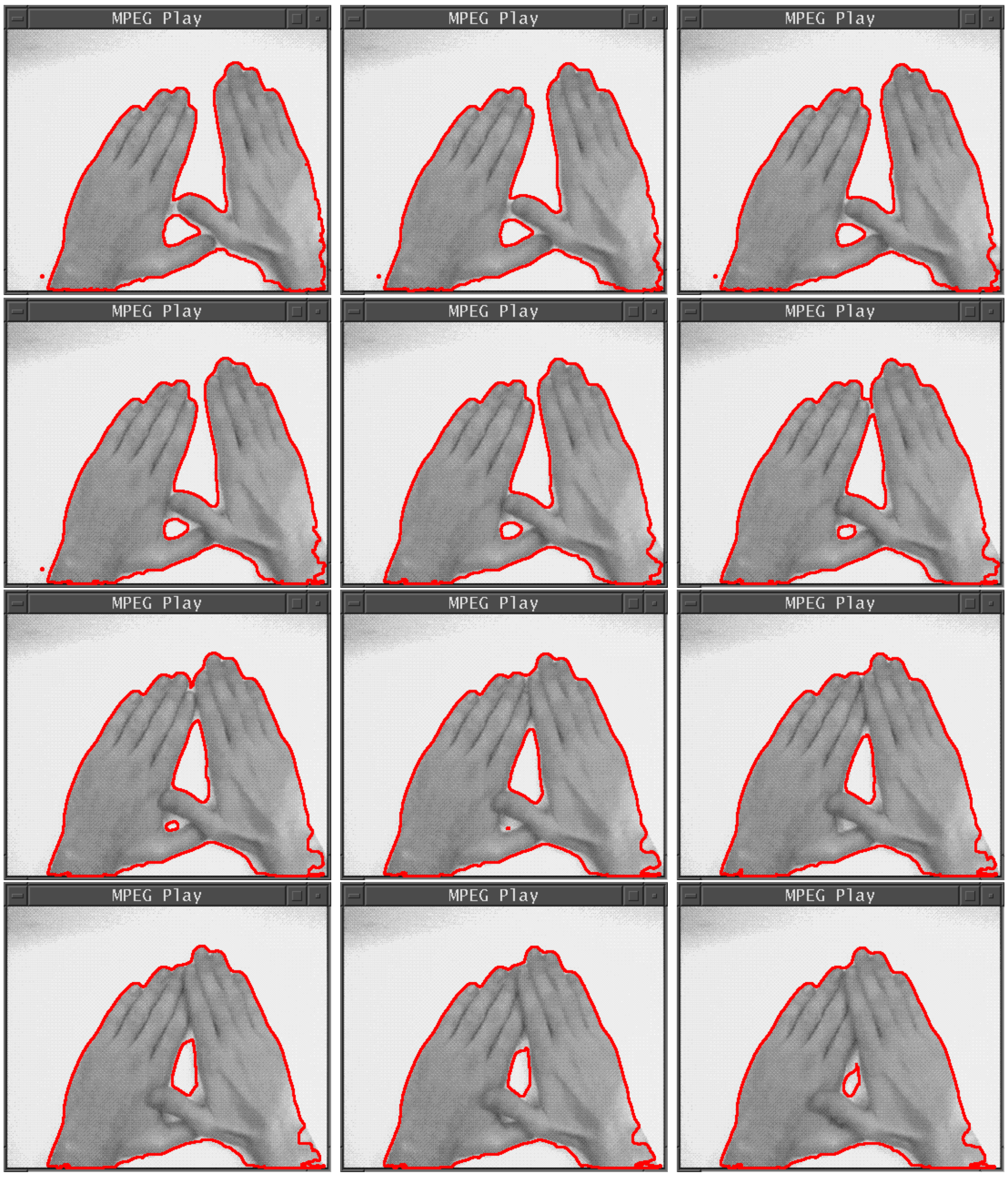

Figure 24: Automatic tracking of the objects marked in figure 23 along consecutive frames of a movie. This figure shows frames 1 to 12. Notice the automatic handling of topology changes.

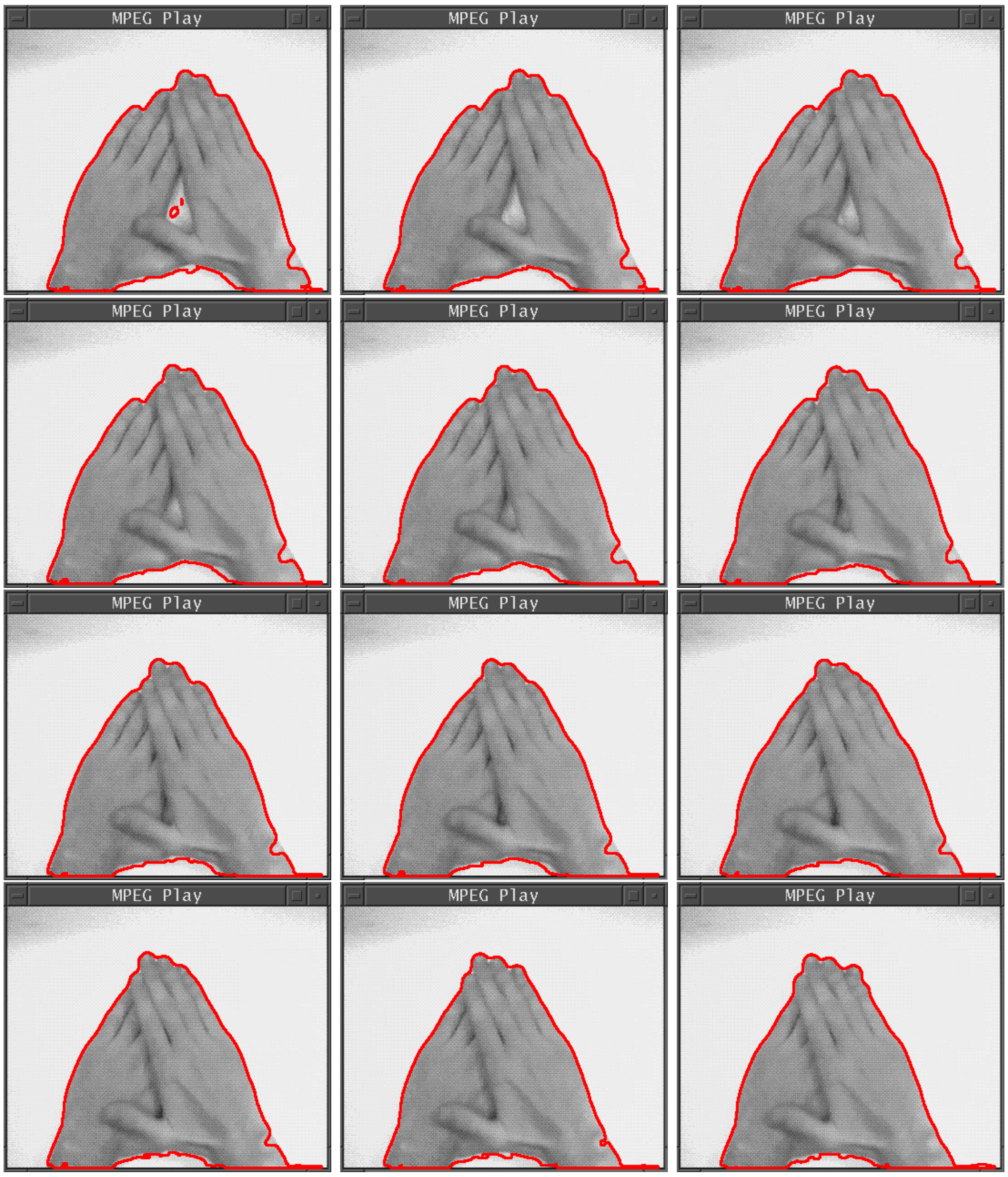

Figure 25: Automatic tracking of the objects marked in figure 23 along consecutive frames of a movie. This figure shows frames 1 to 12. Notice the automatic handling of topology changes.

5 Interpolation and Morphing.

5.1 Introduction.

The MAC framework is also a suitable tool for interpolation and morphing applications: it gives us a continuous deformation from one set of curves/surfaces to another, handling topological changes automatically. This deformation can be "sampled" at will, therefore obtaining as many interpolated curves/surfaces as needed, regardless of the pixel/voxel resolution.

Both problems (interpolation and morphing) have the same formulation in the MAC framework, so we will present them both in this section. The approach here will be different to that of the segmentation and tracking section: we now have both C_1 and C_2 and what we are interested in is the transformation that evolves C_1 into C_2.

In the case of interpolation we will continue with our neurobiological applications, but the forecoming approach is suitable for any kind of interpolation.

5.2 Interpolation of neurobiological boundaries.

5.2.1 The problem.

We recall from section 4.2.1 that high-resolution observation of interneuron connections with the use of a TEM is a very helpful tool for the study of the rules commanding the operations in the nervous system. However, the high resolution offered by the elctron microscope is based on the observation of ultra-thin sections. This represents a serious handicap since the range of thickness of an electron-lucid section is only 1000Å. Consequently, approximately 30-40 serial sections are needed to reconstruct a single interneuronal contact . The biologists have methods to obtain ordered ultra-thin sections (800Åto 1000Å) and observe this slices with a TEM [53]. From this images it is possible to reconstruct the celular structures [46, 47].

But the images obtained with the TEM are of much higher resolution than the slicing of the neurological tissue. That is, the distance between two consecutive slices is much greater (in the order of 10) than the distance between two "points" in a slice that correspond to two adjoint pixels. In other words, resolution in the Ox and Oy axes (the plane axes) is much higher than reolsution in the Oz axis (the slicing or "depth" axis) of the volume. Therefore, the reconstructed 3D volume will have an appearance that greatly depends on the kind of interpolation that the rendering program uses. If we just "repeat" every slice, say, 10 times to form the volume (order zero interpolation), then when it is rendered its surface will have a staircase appearance as the reconstructed sinapsis shown in figure 26.

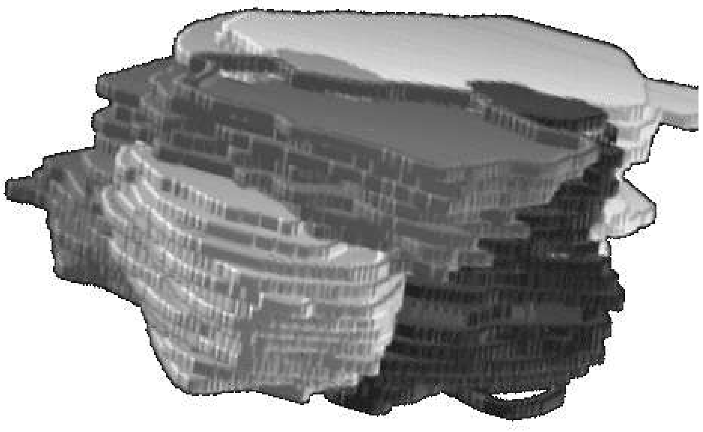

Figure 26: Synaptical terminal reconstructed from TEM images with Neuro3D and rendered with Vprender. Notice the staircase appearance of the surface, due to the lack of resolution in the Oz or "depth" axis.

This is why some kind of smooth interpolation should be used. But higher order interpolation by means of finding correspondent points in both sets of curves has the great difficulty of handling topological changes, while straightforward linear interpolation between the images does not show good results for binary images such as these (where we have objects of constant grey level on a black background: visualization is performed with already segmented images) and may even ignore real structure or create apparent structure [48].

5.2.2 Interpolation with MAC.

Interpolation with the MAC framework only takes into account the information in each pair of consecutive images to interpolate the set of curves in-between them. Interpolation in the whole volume (sequence of images) is then reduced to independent interpolations between pairs of images, so we will just present the simple case where we have two binary images I_1 and I_2. The evolution equation to perform the interpolation would be:

$$I_{1_t}(x,y,t) = (I_2(x,y) - I_1(x,y,t)) \cdot max\left\{|\nabla I_1|, G_m\right\} \tag{39}$$

where G_m is a positive constant.

Some considerations about it. First of all, why do not we have a system of coupled PDE's now?

Remember that the set of curves we want to interpolate are known: C_1 on the first image and C_2 on the second. That is why our images are binary: they have the same constant value (say, zero) in the inside of the curves and another constant value (say, 255) on the outside. But we *construct* these images: what we have is the sets C_1 and C_2, and we create the image I_1 so that C_1 is a level set of it, and I_2 so that C_2 is a level set of it, of the same level as C_1 is to I_1. So actually the images do not have to be binary. We saw in section 3.4.1 that if C_1 and C_2 are both level sets (of the same level) of their respective images, then the generic MAC algorithm as formulated in equation 14 would deform the surface U so that in $t = 0$ its zero level set is C_1, and in the limit $t \to \infty$ its zero level set is C_2. But, recalling the formulation of Osher and Sethian's algorithm in section 2.4.1, the choice of zero as the level set is arbitrary, it can be any α-level set as long as we stick to that value all along the evolution. So if we construct U as a continuous Lipschitz function such that its α-level set in $t = 0$ is C_1, then the evolution 14 would deform U so that in the limit $t \to \infty$ its α-level set is C_2. In our case, since we construct I_1 and I_2, we can make them be continuous Lipschitz functions with α-level sets C_1 and C_2, respectively, so then the surface U can be I_1: thus, the second equation in 14 is exactly the same as the first one, so we can drop it. The evolving set of curves $C_1(t)$ is obtained for any t by finding the α-level set of U, that is, of I_1.

The above reasoning would make us choose only the first of the equations of the system in 14:

$$I_{1_t}(x, y, t) = (I_2(x, y) - I_1(x, y, t)) \cdot |\nabla I_1(x, y, t)| \tag{40}$$

but instead we made an alternative formulation in equation 39. Why so?

The reason for this was already mentioned in section 3.7.4. Recalling the problem of non-convergence from section 3.7.2, equation 40 does not guarantee that I_1 will evolve into I_2: the steady state of the evolution may be achieved not only when $(I_2 - I_1) = 0$, but also when $|\nabla I_1| = 0$. This problem was shown to happen, for this kind of images, when there is a subset of C_2 completely disjoint with C_1. To solve this problem, we can impose a minimum gradient $|\nabla I_1|$, $G_m > 0$, so that whenever $|\nabla I_1| < G_m$ we set $I_t = (I_2 - I_1)G_m$. This guarantees the evolution of I_1 towards I_2, since the steady state is only reached when $(I_2 - I_1) \equiv 0$. If there is no subset of C_2 disjoint with C_1, then G_m can be zero (in which case equation 39 reduces to 40). Else, G_m should be a small positive value, since it is only needed to "start" the evolution in the regions where the gradient of I_1 is flat: once $|\nabla I_1| > G_m$, equation 39 behaves as 40. This can also be interpreted as "bending" the regions of flat gradient of I_1 so that the evolution does not "get stuck". A good value for G_m, for the aforementioned implementation with "binary" images in the range 0-255 and $\alpha = 128$, could be $G_m = \frac{10}{\Delta x}$. Of course, the greater G_m is the faster I_1 will converge to I_2, but too great a value may invalidate the numerical stability results derived in section 3.6.

Recapping, since equation 39 ensures a continuous evolution from I_1 to I_2, then it also implies a continuous deformation of its α-level set $C_1(t)$, from $C_1$to C_2. In practice, the algorithm follows these steps:

1. From two sets of curves C_1 and C_2 we construct the functions I_1 and I_2 so that they have the aforementioned sets as their α-level sets, for an arbitrary value α.

2. A value for G_m is chosen. If we construct the images to have the value of zero inside the curves and 255 outside them, then a good choice for G_m would be $G_m = \frac{10}{\Delta x}$.

3. The evolution is run until $max_{x,y}\{|I_2(x,y) - I_1(x,y,t)|\} < D$ for a certain positive value D, that is, until both images are similar enough (recall that $lim_{t\to\infty} I_1 = I_2$ and that the maximum difference between I_1 and I_2 decreases with time, as seen in section 3.6.2). Another stopping criterion would be one of similarity between the sets C_1 and C_2, more accurate but also computationally more expensive (since we should have to contour I_1 and compute the distance between the two sets everytime we wanted to evaluate the stopping of the evolution, instead of the simpler storing of the maximum $|I_2 - I_1|$). When it stops, the evolution has run for a certain number of iterations, say N.

4. If we must iterate a number of M samples in between C_1 and C_2, then the evolution is run again but this time we contour I_1 (at the level α) every $\frac{N}{M}$ iterations, hence obtaining uniformally spaced (in time) samples of the continuous evolution of C_1 towards C_2. If $N < M$, we should run the previous step with lower speed, that is, a smaller value for the time step in the numerical implementation. At top speed, for curves not too far apart we get $N \simeq 30$.

Finally, one could think of using equation 39 as the first equation in the system of coupled PDE's used for segmentation and tracking, to ensure convergence of I_1 towards I_2 as we have done here. But this is not possible. The reason lies in the fact that that system has the tracking function U, with the purpose that its zero-level set moves with the same speed as it would if it were a level set of I_1. This implies that the value multiplying $|\nabla U|$ has got to be the same that multiplies $|\nabla I_1|$(in absolute value), to ensure that the normal components of the respective level sets be the same (see section 3.4.1). So for the points where $|\nabla I_1| < G_m$ we get $I_t = (I_2 - I_1)G_m$ or, equivalently, $I_t = (I_2 - I_1)\frac{G_m}{|\nabla I_1|}|\nabla I_1|$. Hence the velocity for U at those points should be $(I_2 - I_1)\frac{G_m}{|\nabla I_1|}$ (or its opposite). Since $|\nabla I_1|$ can be arbitrarily close to zero, the numerical stability condition found in section 3.6 would imply a time-step of zero to ensure stability. In other words, we could not have a stable implementation of the evolution. But this is not an issue because in the case of segmentation and tracking, since C_1 is over local edges (where $\nabla I_1 \neq 0$) the evolution will not "get stuck" for some region around C_1: thus the non convergence problem will not influence the tracking operation, as seen in section 3.7.3.

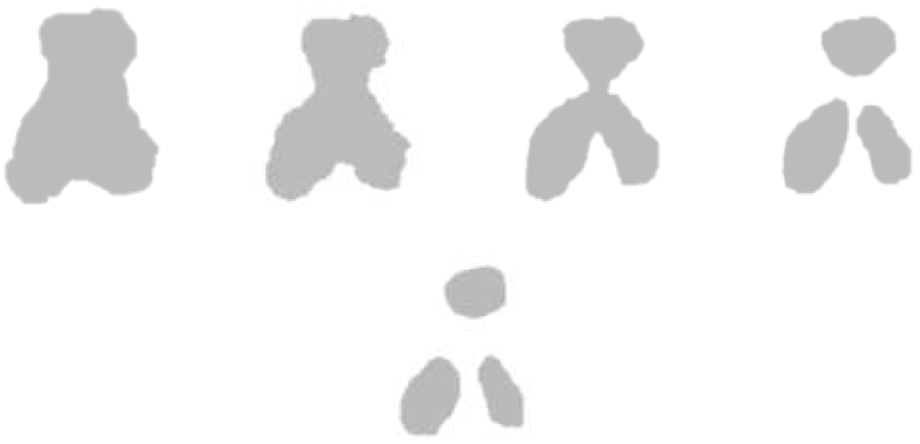

Figure 27: Five consecutive images that we want to interpolate.

Future work should try to incorporate into the interpolation process between two images, information from other (adjacent) images, by means of optical flow or Kalman filtering.

5.2.3 Examples.

Figure 27 shows five consecutive images already segmented. These are synthetic images, but they could be slices of neural tissue just as well. Let us assume that resolution in the plane is six times greater than that in the $\overrightarrow{Oz}$ axis: in that case, we will need to interpolate five images in between each pair of consecutive slices.

Zero-order interpolation gives the expected staircase appearance shown in figure 28.

Interpolation with the MAC algorithm is shown in figures 29 and 30. The top row shows the original images, while each column shows consecutive interpolations between the first image of the column and the first image of the following column. In between each pair of consecutive (original) images the interpolated images are computed by sampling the morphing of one image into the other every three iterations, for a total of fifteen iterations.The 3D reconstruction is shown in figure 31.

5.3 Morphing.

5.3.1 Introduction.

The general problem of metamorphosis between two objects, commonly called *morphing*, has gained considerable popularity in recent years, especially in the entertainment industry.

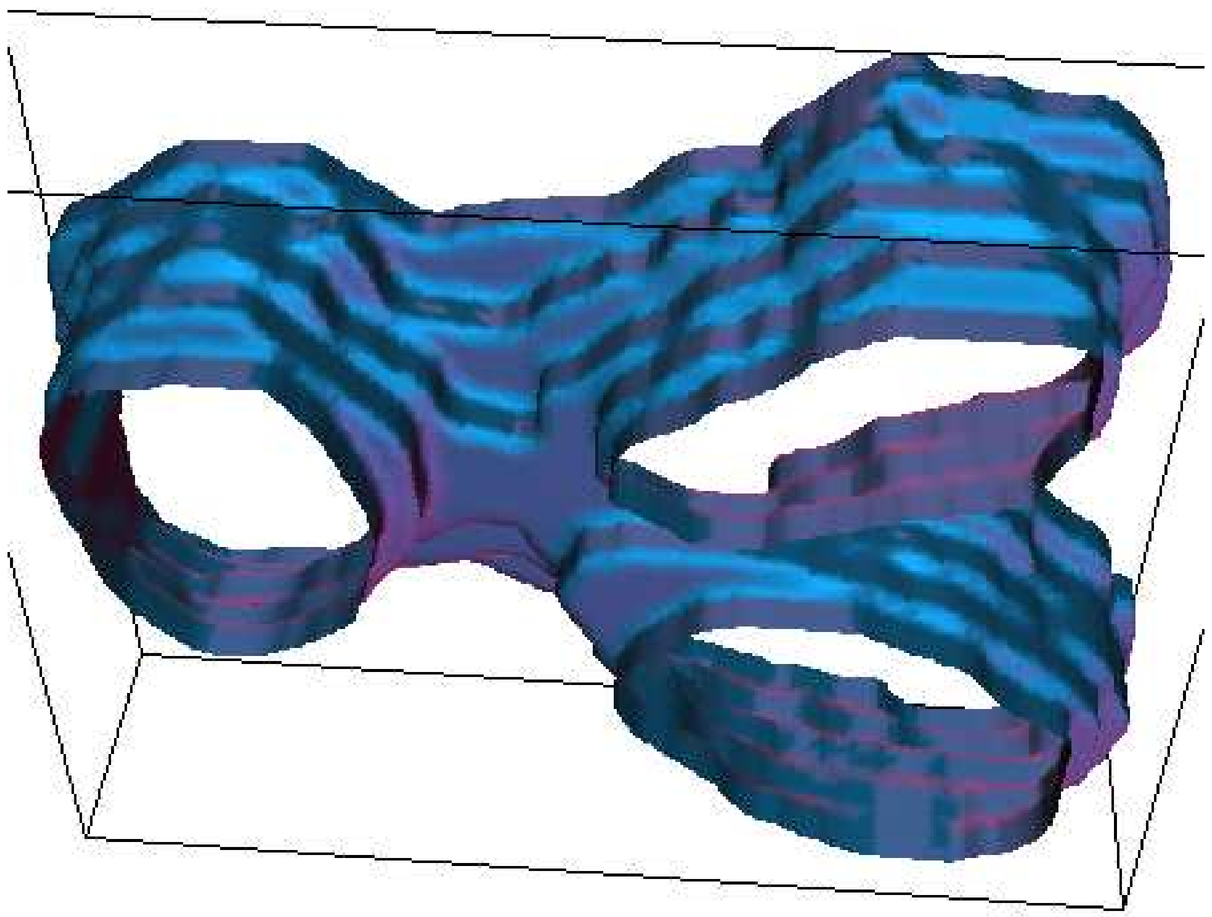

Figure 28: Three dimensional recontruction of a zero-order interpolation of the images shown in figure 27.

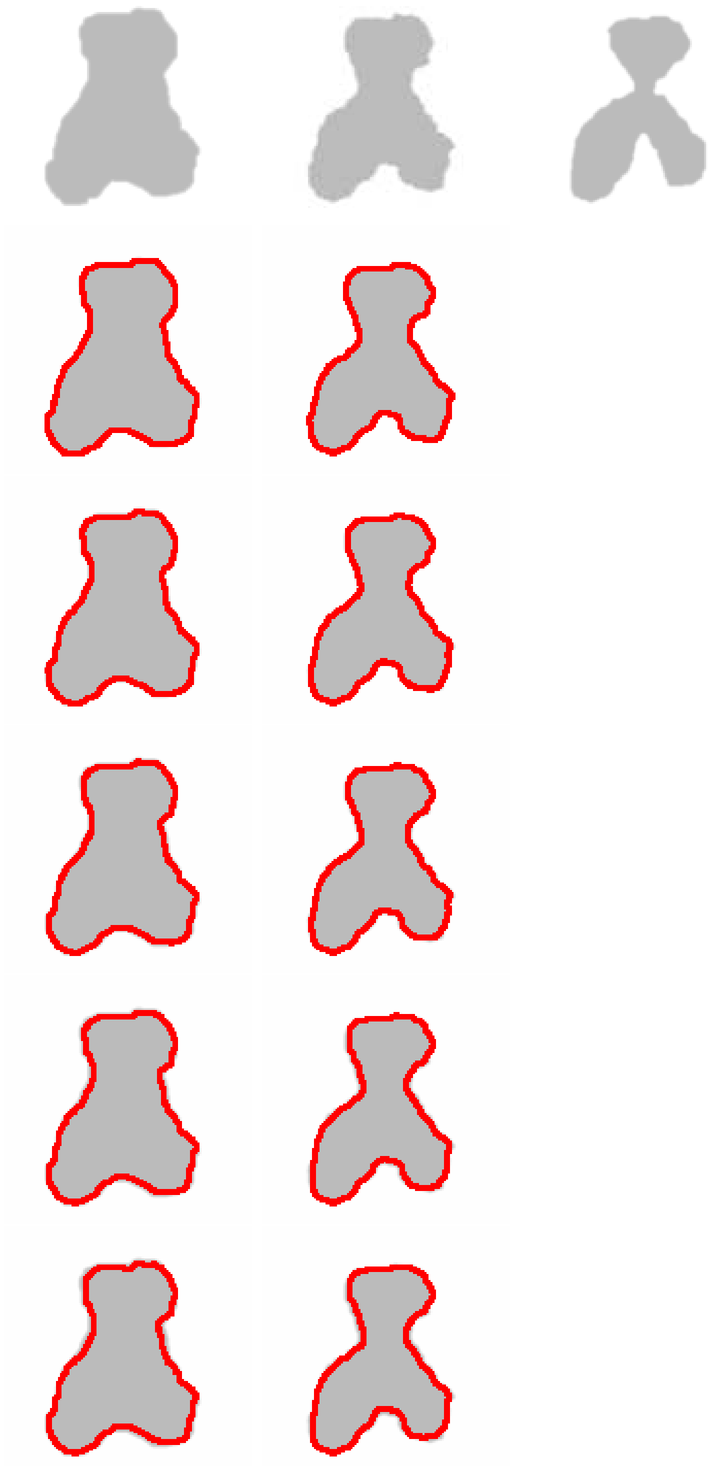

Figure 29: Top row: original images from figure 27. Each column shows consecutive interpolated boundaries, interpolated between the first image of the column and the first image of the following column.

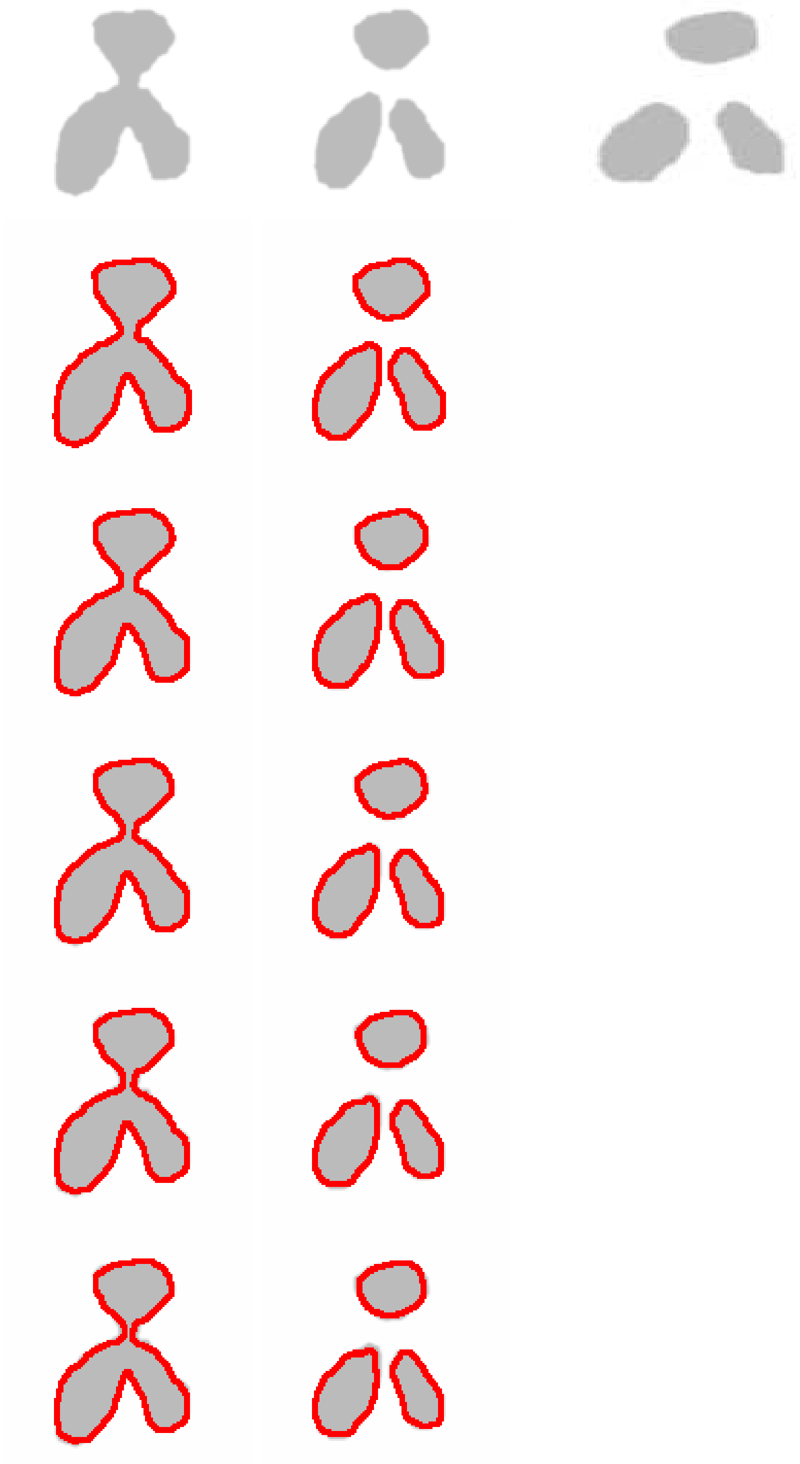

Figure 30: Top row: original images from figure 27. Each column shows consecutive interpolated boundaries, interpolated between the first image of the column and the first image of the following column.

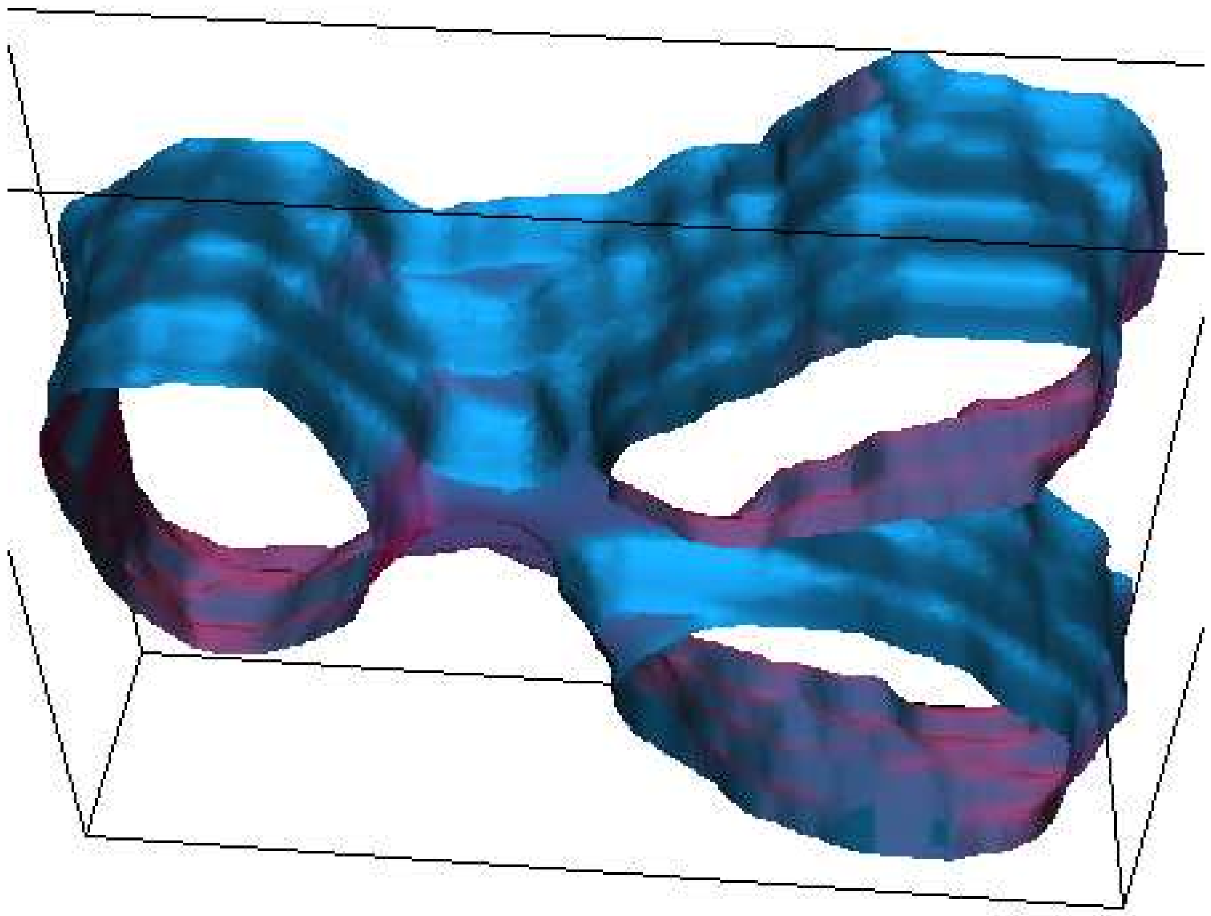

Figure 31: Three dimensional reconstruction of the interpolated boundaries shown in figures 29 and 30.

In its basic form, morphing techniques perform controlled warping of a 2D gray-level image to transfrom it into another, as seen in [48]. The user supplies a set of control points at corresponding locations that are distinctive in both images. Then a program forms a tesselation of the first image into triangles, and each triangle is unformly strectched to fit the location of the corner points in the second image. Using spline or cubic equations to perform the stretch gives a better appearance, but this is not a necessary indication that dimensions are preserved or that measurements can be made on such images. Unfortunately, any intermediate forms produced by image morphing methods exist only in image form: surface models are often required in animation or keyframing, or to allow shadows or lighting effects to be computed [15]. For the metamorphosis of 3D surface models, most research has focused on morphing between a restricted, topologically similar class of shapes; often the user has little or no say in how the morph takes place (see [15] and references therein).

5.3.2 Morphing with MAC.

Using the MAC framework, morphing between two objects of arbitrary dimension and arbitrary topological characteristic can be achieved. The straightforward extension of equation 39 to any dimension allows us to ensure a continuous deformation of one set of curves/surfaces C_1 into another set C_2. The procedure is the same as that presented in the previous subsection for interpolation, now the images I_1 and I_2 being generic functions from R^n into R. The evolution equation gives us the continuity of the deformation, while Osher and Sethian's formulation allows the scheme to handle topological changes automatically. Futhermore, this approach does not require to compute the surface mesh in order to perform the deformation: the surfaces are implicit, α-level sets of the functions I_1 and I_2. The speed of the morphing is regulated by the choice of time step. At the maximum speed that the numerical stability criterion allows (equation 32) and for objects not too far apart, the morphing is achieved in around 30 iterations.

Future work should allow the user to specify correspondant pairs of points on the two objects, and contemplate the extension of the scheme to gray-level volumes (the 2D case being the most interesting, of course).

5.3.3 Examples.

Figure 32 shows two volumes, one of them dipheomorphic to the sphere and the other composed of eight cubes. The morphing of one into the other is shown in figures 33 to 35.

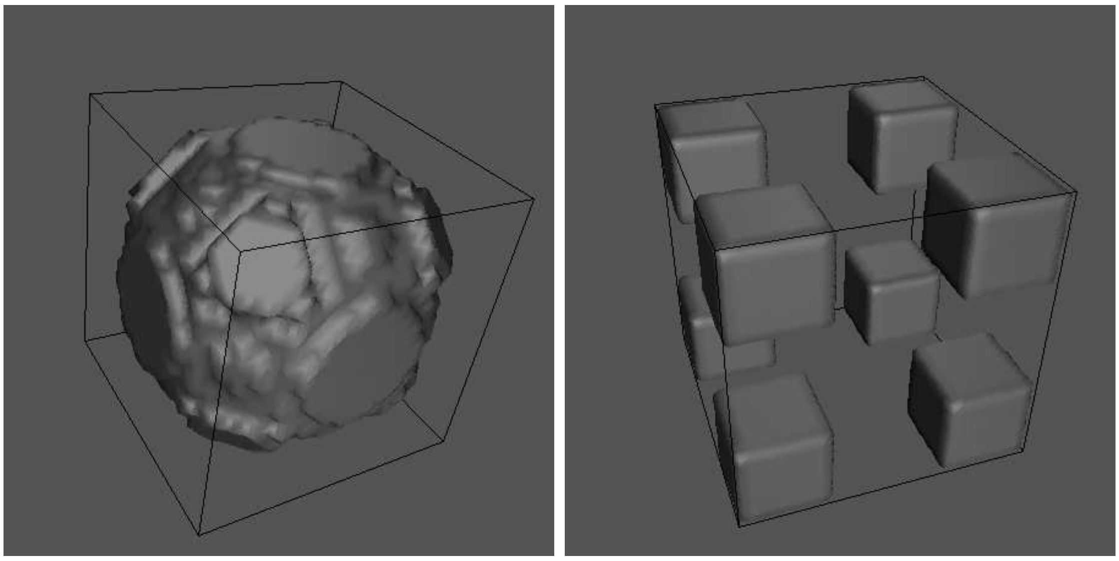

Figure 32: Two 3D surfaces: one is dipheomorphic to the sphere, the other is composed of eight cubes.

6 Flattening+Tracking.

6.1 Introduction.

In this last section we will present yet another application within the MAC formulation. Firstly, the subject of measuring neural activity in the cortex via functional magnetic resonance imaging (fMRI) is introduced, as well as the need of counting with a flattened representation of the cortical surface. Secondly, a novel algorithm that performs flattening of the surface while tracking a set of curves on it (Flattening+Tracking, F+T) is presented. Thirdly, we show how MAC can be adapted to improve the performance of the F+T algorithm. Finally, some examples are given.

6.2 The need of flattening.

This subsection includes excerpts from [56]. Let us begin by saying that fMRI has provided a non-invasive method of visualizing a correlate of neural activity in the brain. This is because the relative amounts of oxygen around active and inactive areas of the cortex are different, and the magnetic properties of oxygenated and deoxygenated blood differ (fMRI measures the relative amount of oxygen in the surrounding blood-flow). The ability to measure cortical activity non-invasively is an important breakthrough, providing us with a new oportunity to study the activity of single human brains at relatively high spatial resolution.

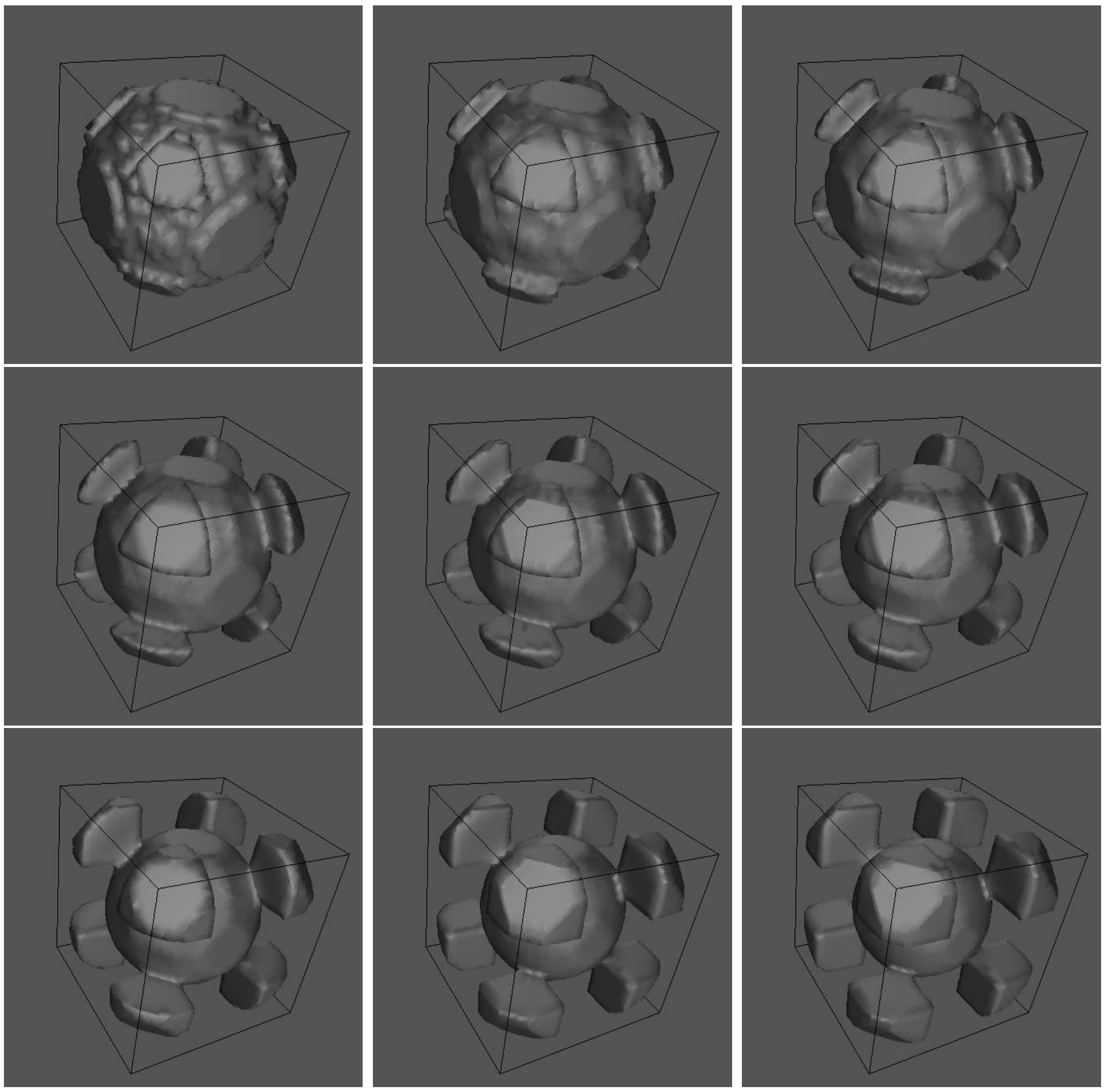

Figure 33: First 9 steps of the morphing between the two surfaces depicted in figure 32.

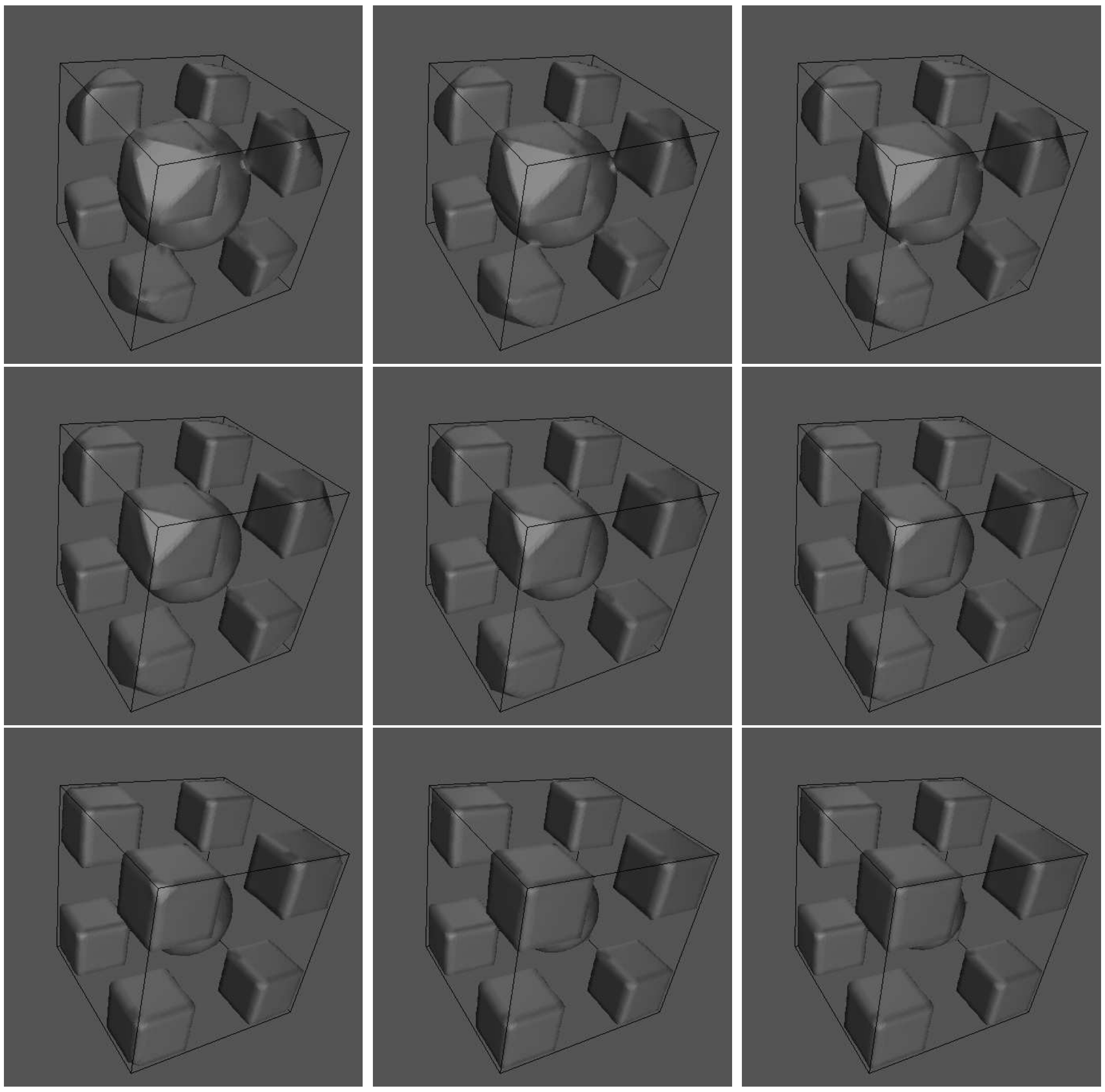

Figure 34: Steps 9 to 18 of the morphing between the two surfaces depicted in figure 32.

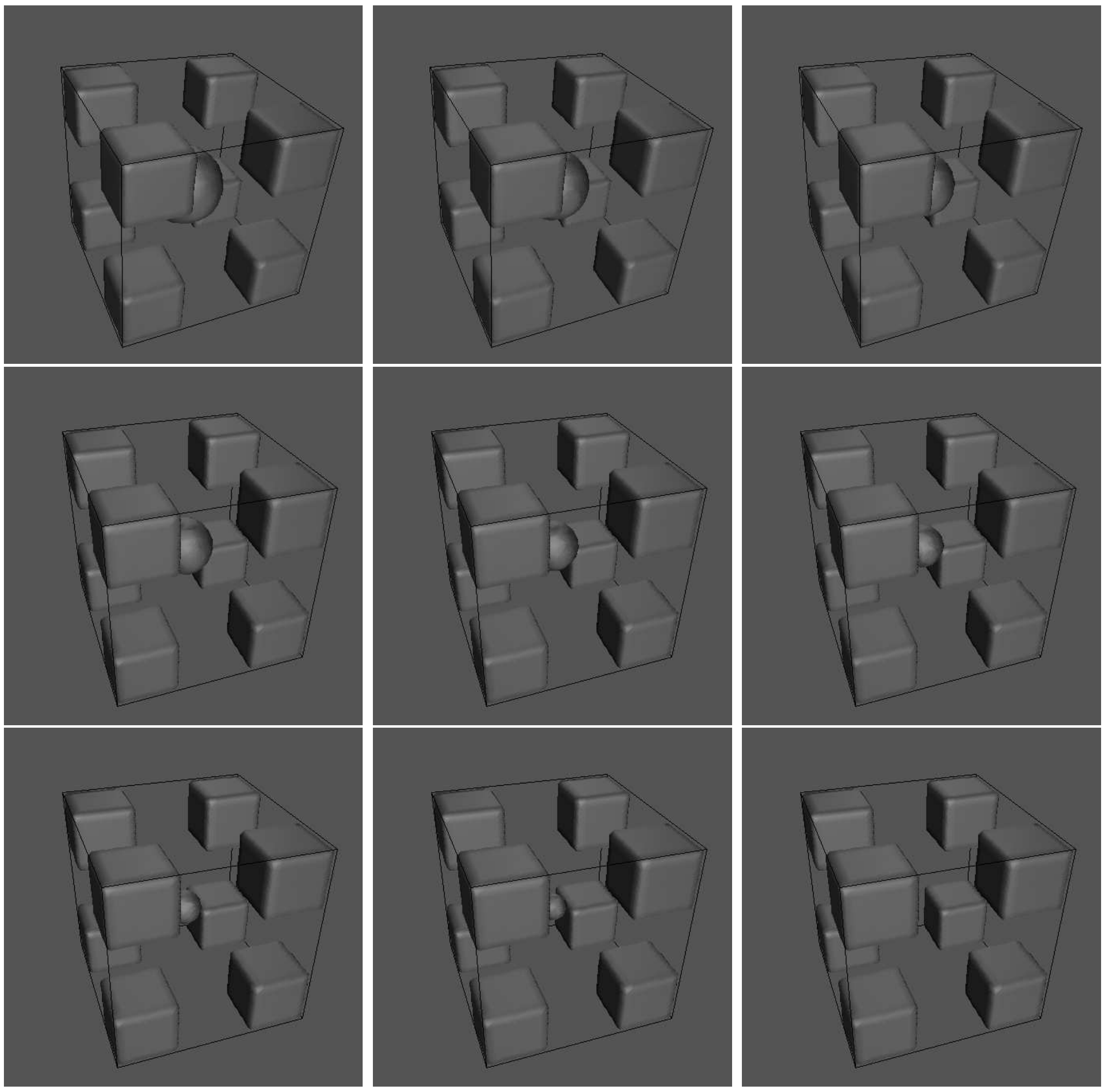

Figure 35: Last 9 steps of the morphing between the two surfaces depicted in figure 32.

Among the various parts of the brain, the cortex is the most prominent, and one of the most intensely studied. The cortex is divided into two hemispheres connected by a massive set of nerve fibers. The cortex is composed largely of two types of tissue: gray matter and white matter. Gray matter forms the outer layer of the cortex, with an average thickness of 3mm in humans, and contains a high density of neurons. It encases the inner white matter almost completely: white matter is made of nerve fibers that connect different parts of the cortex, as well as the cortex with other parts of the brain. fMRI measures the neural activity in the gray matter.

Despite its complex outward appearance, the structure of each hemisphere is quite straightforward and consistent across human brains. The cortical gray matter surface is the same as that of a crumpled sheet; i.e., it does not have holes or self-intersections. After a segmentation process, the brain is rendered as a 3D surface and the observer sees mainly those portions outside the folds. But because it is important to see neural activity buried deep within the 3D folds of the brain, visualizing fMRI data requires novel visualization tehniques. An increasingly popular way of visualizing such mappings is to superimpose fMRI measurements on flattened representations of the cortical surface [14, 18, 20, 60]. For instance, data from monkey and human studies show that neurons within a certain area of the occipital lobe are *retinotopically organized*: neurons that are responsive to nearby regions of the visual field are located close to one another within the gray matter layer. Because the retinotopic organization of visual areas, it is possible to create simple visual stimuli that generate continuous traveling waves of neural activity in the visual cortex. The spatial structure of these traveling waves, represented on the flattened cortical surface, can be used to determine the locations and boundaries of different areas of specialized processing.

6.3 The F+T algorithm.

6.3.1 Introduction.

We will now present a novel algorithm that performs the flattening of a 3D surface (although it can be extended straightforward to any arbitrary dimension) while tracking any given set of curves over it. The motivation for this is clear: since we have the fMRI measurements for the original surface, we want to track the deformation of some user-defined areas (whose boundaries conform the aforementioned set of curves) along the flattening process, so at the end we have the fMRI measurements superimposed on the flattened representation of the cortical surface.

This purpose is achieved via a system of two coupled PDE's: one of them performs the flattening, while the other keeps track of the deforming set of curves. While the concept is very similar to that expressed earlier in this work for segmentation and tracking, note that here we deal with evolving 3D surfaces and tracking of evolving curves *on them*: that is, we have a balloon, we know how it is inflated, and we want to know how a drawing on it would deform along the inflation process. To the best of our knowledge, there is no related work in the literature.

6.3.2 Flattening.

Here we shall include some concepts on surface smoothing from [41]. Flattening of 3D surfaces can be achieved using geometric heat-type flows, a subject that has received much attention in the past few years. The smoothing of planar objects was originally performed by filtering their boundary with a Gaussian filter, but this process is extrinsic, unrelated to the geometry of the given image. This and other problems can be solved by replacing the classical heat flow with geometric heat flows, as seen in sections 2.2 and 2.5 (see [49]).

In the 3D case, in contrast to the planar case, certain geometrical constraints must be imposed to the initial surface in order that the evolving surface remains smooth. We consider the surface analogue of the curve flow 2 in which β is a geometric function of the surface. The most popular choice for β is so that it depends on the principal curvatures of the surface. The most important special case is the mean curvature flow, when β is the mean curvature or average of the principal curvatures. Another important choice is when β is a function of the Gaussian curvature (which is the product of the principal curvatures), although this case is much less well understood.

For inward mean curvature flow, Huisken [30] proved that a convex surface evolves into a round point, meaning that it becomes asymptotically spherical before collapsing to a point in finite time. Chow [11] proved the same result when the (inward) velocity is given by the square root of the Gaussian curvature. Urbas investigated the expanding evolution of convex surfaces in [57, 58], proving they become asymptotically spherical under certain conditions. The situation for non-convex surfaces is much more complicated and still the subject of much research; see for example [55, 26, 2]. In general, a non-convex surface evolving according to the mean curvature will not remain smooth, or even connected, as illustrated by the famous dumbbell example. In this example, a dumbbell (two spheres joint by a cylinder) which is a non-convex but connected surface becomes disconnected during the smoothing process: the joining cylinder splits (being its curvature higher than that of the spheres) anf the two spheres become disconnected.

As stated above, the very wrinkled cortical surface is topologically equivalent to a sphere, not having holes nor self-intersections, but it is not convex. Therefore, the smoothing process may transform it into a non-connected surface. This is what happens in the example in figure 36 if we use the mean curvature flow. Here we see how the bottom-left protuberance of the surface disconnects from the main body.

In [8], Caselles et al. propose this flow:

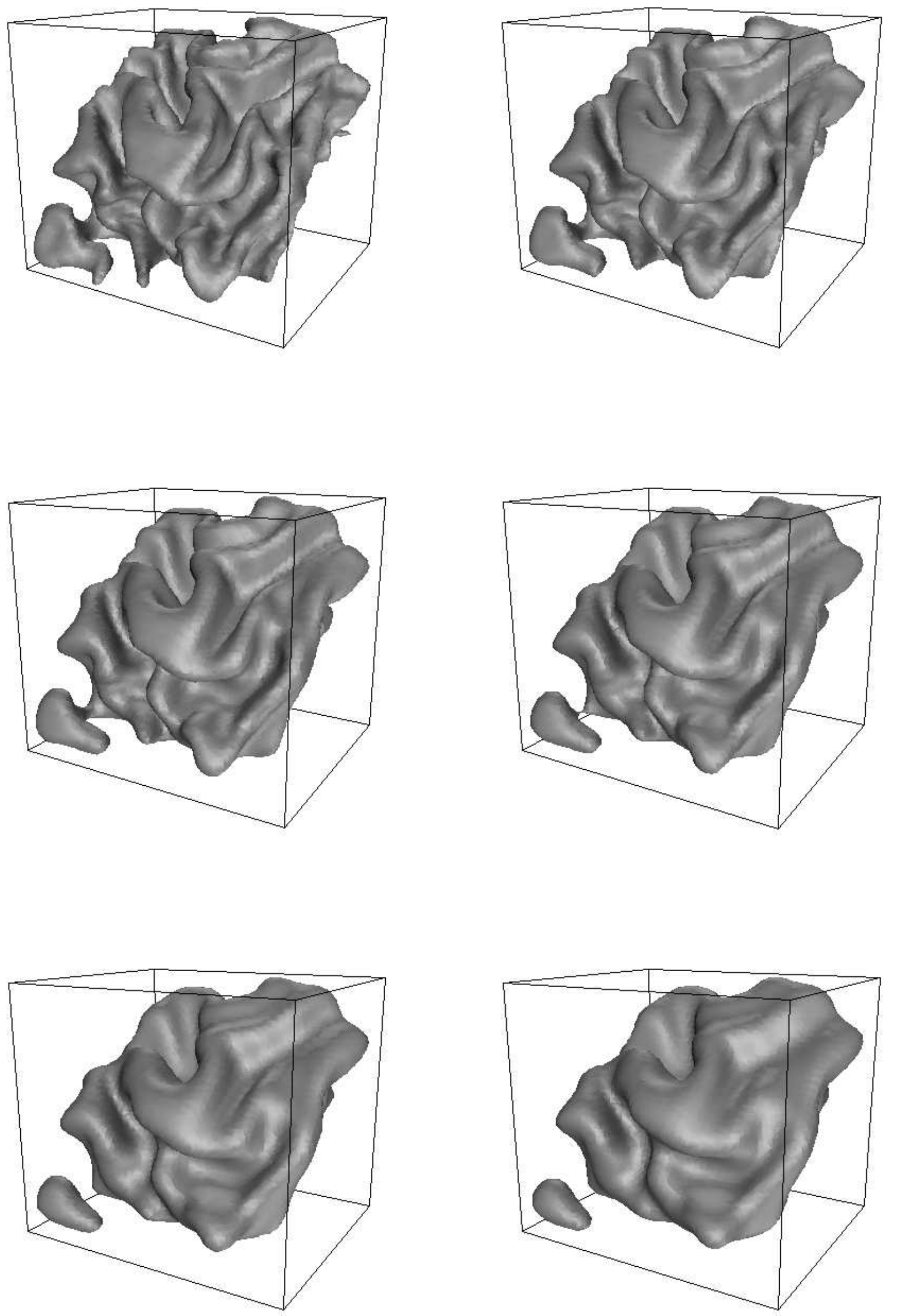

Figure 36: Flattening of a cortical surface with mean curvature flow. Notice how the bottom-left protuberance disconnects from the main body due to the smoothing process.

$$V_t = min(\kappa_1 \kappa_2)\,|\nabla V| \tag{41}$$

where V is a function $V : R^3 \to R$ and its zero-level set is the 3D surface V_0 that is smoothed.

With this flow, an evolving dumbbell (bent or straight) does not divide into several sets [8]. For the same surface of the last example, we can see in figure 37 how its flattening keeps the surface connected (evolving with equation 41). Unfortunantely, this is not so for every surface dipheomorphic to the sphere (see an example in [8]), so we can not assure that any cortical surface will remain connected if smoothed with this flow. Furthermore, we see in this example how the surface remains connected but a hole appears on it, therefore the flow does not keep the surface dipheomorphic to the sphere.

For our applications we propose the following flow:

$$V_t = \begin{cases} sg(\kappa_1 + \kappa_2) \cdot max(|\kappa_1|, |\kappa_2|)\,|\nabla V|, & \kappa_1 \kappa_2 > 0 \\ 0 & else \end{cases} \tag{42}$$

This flow acts as follows:

- at the points where both main curvatures are positive, “hills” on the surface, the volume shrinks, smoothing them;
- at the points where both main curvatures are negative, “valleys” on the surface, the volume expands, smoothing them;
- at the rest of the points, “saddle” points where the main curvatures have different sign, the volume does not move.

Thus, this flow will shrink protuberances on the surface and therefore it will not “melt” them with any other part of the surface, like the flow given in equation 41 did. And, since the “saddle” points do not move, the protuberances will not get disconnected from the main body of the surface if their joint to it is cylindrical, not bent. But it is easy to show that if we apply this flow to the bent dumbbell, the “handle” disappears: in fact, the “handle” of the bent dumbbell is a bent cylinder, where the outward points of its surface have both main curvatures positive, and the inward points are “saddle” points, and so the “handle” will shrink from one side until it vanishes (in finite time). So this flow does not guarantee, either, that the surface will remain dipheomorphic to the sphere. But it gives better results that the aformentioned smoothing flows for the volumes we have worked with, so it is the flow we will be using for flattening. See some results of it on figure 38, for the same cortical surface shown previously.

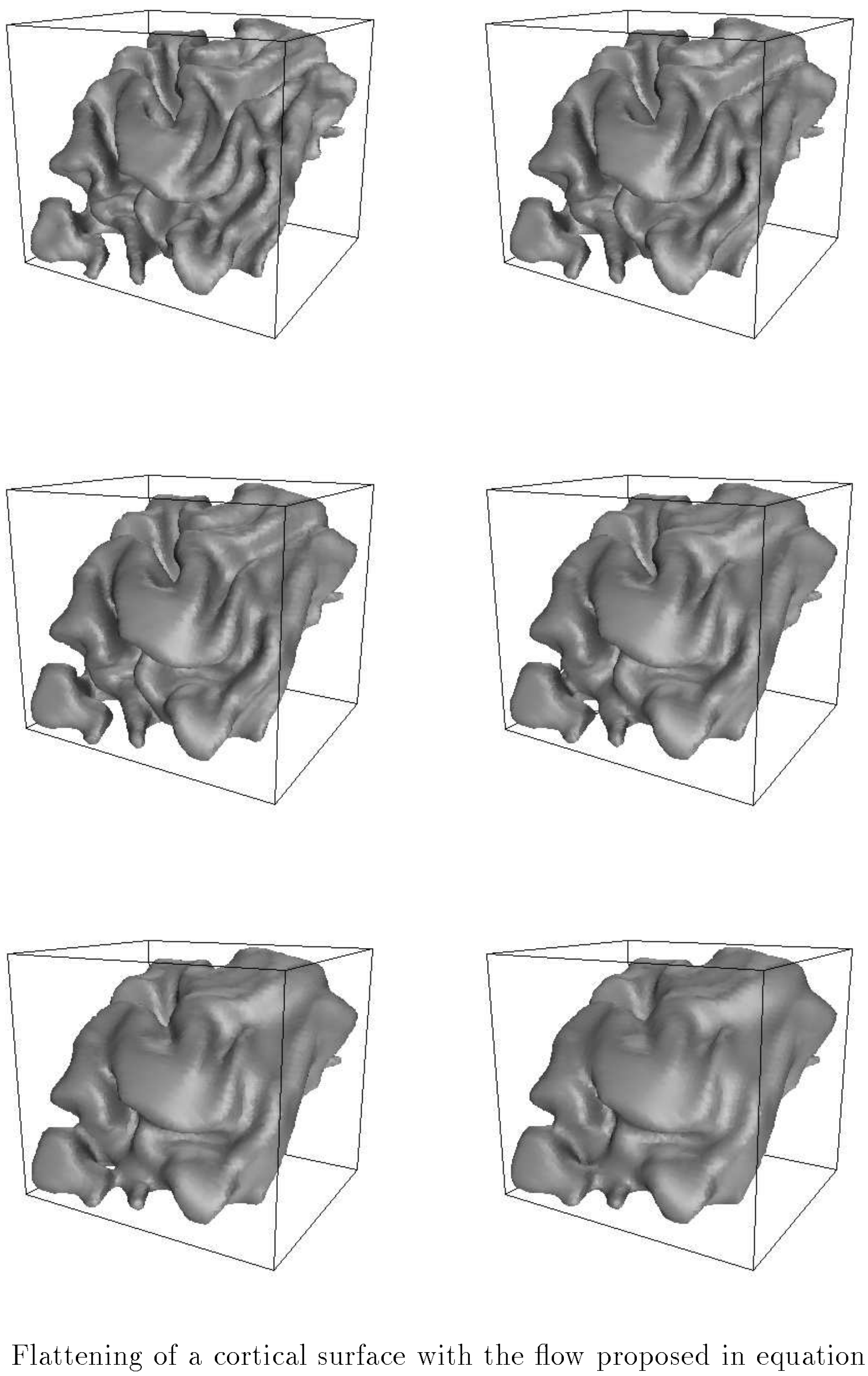

Figure 37: Flattening of a cortical surface with the flow proposed in equation 41. Notice how the bottom-left protuberance does not get disconnected with this evolution, but a hole appears when this protuberance “melts” with the main body (compare with figure 36).

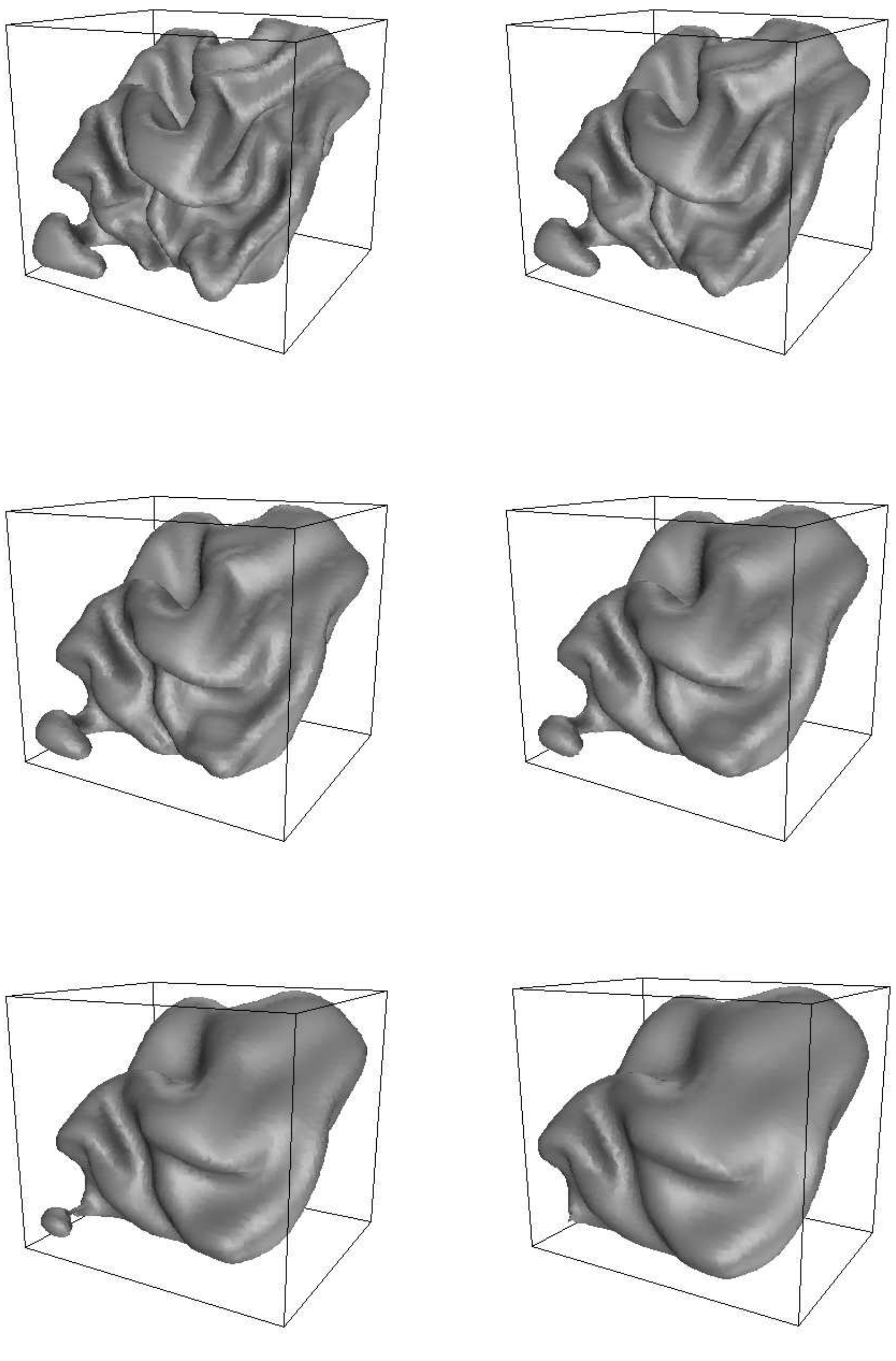
Figure 38: Flattening of a cortical surface with the flow proposed in equation 42. Notice how the bottom-left protuberance does not get disconnected with this evolution, nor does this protuberance "melt" with the main body (compare with figures 36 and 37).

6.3.3 Tracking.

The tracking problem, as stated in the previous subsection, can be formulated for our case in this way:

1. The 3D surface V_0 evolves in time according to certain (known) evolution equation.
2. There is a set of curves C_0 on $V_0(0)$. That is, each of the curves of the set C_0 is a curve $C : [0,1) \to R^3$such that $C(p) \in V_1(0), \forall p \in [0,1)$.
3. We want to know $C_0(t)$, the evolution of the set of curves on V_0.

A possible approach would consist in placing some nodes on C_0 and follow their trajectories for the evolution equation 42. But this "Lagrangian" approach has all the problems already mentioned in section 2.4.2: the marker particles on the evolving curve may come very close during the evolution. This could be solved by a redistribution of the marker particles, altering the equations of motion in a nonobvious way. Besides, a simple, Lagrangian, difference approximation, requires an impractically small time step in order to achieve stability, while in some cases stability implies a time step equal to zero.

We chose a completely different approach. Trying to use Osher and Sethian's formulation, we came up with the following scheme:

1. Let V be a function $V : R^3 \to R$ such that its zero-level set is the 3D surface V_0 that is going to be smoothed. Let C_0 be the set of curves on V_0 that we want to track. We also know $V(t)$ (and therefore $V_0(t)$).
2. We construct a 3D surface U_0 such that $C_0 = \left\{ \overrightarrow{X} \in R^3 : \overrightarrow{X} \in U_0 \bigcap V_0 \right\}$. Then we construct another function $U : R^3 \to R$ such that its zero-level set is U_0. In other words, $C_0(0) = U_0(0) \bigcap V_0(0)$.
3. The purpose of the tracking is to find $C_0(t)$. If we find the right evolution equation for $U(t)$, we could determine $C_0(t)$ as the intersection of $U_0(t)$ and $V_0(t)$.

6.3.4 F+T evolution equation.

The following system of equations is the F+T evolution equation that we propose:

$$\left\{ \begin{array}{c} V_t = \beta \left| \nabla V \right| \\ U_t = \beta (\overrightarrow{N}_V \cdot \overrightarrow{N}_U) \left| \nabla U \right| \end{array} \right. \qquad (43)$$

where

$$\beta = sg(\kappa_1 + \kappa_2) max(|\kappa_1|, |\kappa_2|), \quad \overrightarrow{N}_V = -\frac{\nabla V}{|\nabla V|}, \quad \overrightarrow{N}_U = -\frac{\nabla U}{|\nabla U|}$$

(44)

It is a system of coupled PDE's. The first equation in 43 smoothes the surface V_0. The second equation evolves the volume U with a normal speed that depends on the previous equation. This speed is chosen to be the projection of the normal speed of V onto the normal of U. Why is this so?

Firstly, recall from section 2.2 that it can be proven (see [21]) that if the normal component of the velocity is a geometric function of the surface (i.e., does not depend on the parameterization) then $Img[\bullet]$ (which represents the "geometry" of the curve) is only affected by its normal component. The tangential component affects only the parameterization and not $Img[\bullet]$, which is independent of the parameterization by definition. Secondly, consider the following evolution of the surface U_0:

$$U_{0_t}(\overrightarrow{X}) = \begin{cases} \beta\overrightarrow{N_V} = \beta\gamma\overrightarrow{N_U} + \beta\delta\overrightarrow{T_U} & , \overrightarrow{X} \in U_0 \bigcap V_0 \\ whatever(geometric) & , else \end{cases} \tag{45}$$

where $\gamma = \overrightarrow{N}_V \cdot \overrightarrow{N_U}$, and $\delta = \overrightarrow{N}_V \cdot \overrightarrow{T}_U$. That is, if $\overrightarrow{P} \in U_0 \bigcap V_0$, then

$$U_{0_t}(\overrightarrow{P}) = V_{0_t}(\overrightarrow{P}) \tag{46}$$

for $t = 0$.

Since $\beta\gamma$ does not depend on the parameterization of U_0, and for the points $\overrightarrow{X} \notin U_0 \bigcap V_0$ (for whose there would not be a normal $\overrightarrow{N_V}$ defined) the evolution is geometric, then the evolution in equation 45 is (according to the aforementioned property) equivalent to:

$$U_{0_t}(\overrightarrow{X}) = \begin{cases} \beta\gamma\overrightarrow{N_U} & , \overrightarrow{X} \in U_0 \bigcap V_0 \\ whatever(geometric) & , else \end{cases} \tag{47}$$

We are projecting the speed of every $\overrightarrow{P} \in V_0 \bigcap U_0$ onto the normal $\overrightarrow{N_U}$ because the tangential component affects only the parameterization of the surface and not its geometric appearance.Then, the evolution

$$\left\{\begin{array}{l} V_{0_t} = \beta\overrightarrow{N_V} \\ U_{0_t} = \left\{\begin{array}{ll} \beta\gamma\overrightarrow{N_U} & ,\overrightarrow{X} \in U_0 \bigcap V_0 \\ whatever(geometric) & ,else \end{array}\right. \end{array}\right. \tag{48}$$

satisfies that, for every $\overrightarrow{P} \in U_0 \bigcap V_0$, then $U_{0_t}(\overrightarrow{P}) = V_{0_t}(\overrightarrow{P})$ for $t = 0$. In other words, a point $\overrightarrow{P}$ on the intersection of V_0 and U_0 would move on U_0 so as to be coincident with the arrival point of $\overrightarrow{P}$ as moving with V_0: if $U_0(\overrightarrow{P}(0), 0) = V_0(\overrightarrow{P}(0), 0)$ and $U_{0_t}(\overrightarrow{P}(0), 0) = V_{0_t}(\overrightarrow{P}(0), 0)$, then $lim_{\Delta t \to 0} U_0(\overrightarrow{P}(\Delta t), \Delta t) = lim_{\Delta t \to 0} V_0(\overrightarrow{P}(\Delta t), \Delta t)$. This allows us to generalize equation 46 for every t, and therefore $U_0(\overrightarrow{P}(t), t) = V_0(\overrightarrow{P}(t), t)$. So, recalling that $C_0(0) = U_0(0) \bigcap V_0(0)$, we have the result $C_0(t) = U_0(t) \bigcap V_0(t)$, wich is what we were looking for.

Equation 43 is just the formulation of Osher and Sethian for equation 48, where we chose the arbitrary term (for points not in the intersection of the surfaces) as the simplest one. So if we run equation 43, the evolving set of curves $C_0(t)$ can be determined as the intersection of the zero-level sets of the functions $V(t)$ and $U(t)$.

6.3.5 Numerical implementation.

For the numerical implementation of the scheme the same observations made in sections 2.4.2 and 3.6 are valid.

In the first of the equations in 43, the velocity depends on the main curvatures of the surface, wich imposes a time step in the order of

$$\Delta t < \frac{(\Delta x)^2}{9} \tag{49}$$

where Δx is the smallest dimension of the voxels (see [4]).

In the second equation of the system, the velocity depends of an external field, external to U, so the time step imposed by this equation (alone) would be

$$\Delta t < \frac{\Delta x}{\sqrt{3}C} \tag{50}$$

where $C = max_{i,j,k}\left\{\left\|\beta\frac{\nabla V}{|\nabla V|}\frac{\nabla U}{|\nabla U|}\right\|\right\}$.

Since the equations in 43 are coupled, each iteration of the evolution must use the same time step for both equations. Therefore, for each iteration the maximum time step that ensures stability is

$$\Delta t < min(\frac{(\Delta x)^2}{9}, \frac{\Delta x}{\sqrt{3}C}) \tag{51}$$

6.3.6 The problem of speed.

This formulation provides a novel and simple way to perform flattening (or any arbitrary geometric deformation) of surfaces while keeping track of certain curves on it. But in the past subsection, the main drawback of this approach could be glimpsed, and that is that the speed of the algorithm may be too slow.

For our fMRI applications, where the main curvatures of the folds may be very high (in absolute values), the time step is generally imposed by the tracking equation, and it is often very small. Therefore, many iterations would be needed to arrive to a suffciently smooth surface while simultaneously keeping track of the curves. There are certain ways that can be tried to increase the speed:

1. Define the surface U_0 so that it is as orthogonal as possible to V_0 at their intersection points, and define the embedding function U trying to keep this orthogonality with V for all their level sets. Thus, $\beta \frac{\nabla V}{|\nabla V|} \frac{\nabla U}{|\nabla U|} \cong 0$ and $C \cong 0$ (at least at the beginning iterations) and the time step of the evolution is that of equation 49. Reinitialize U along the evolution whenever C increases much.
2. Compute the time step within a band of $U_0 \bigcap V_0$ of a few voxels width and reinitialize the band whenever it is reached by C_0. Thus, we ensure stability on a certain band surrounding the region of interest and if there are greater values of the curvature elsewhere they do not affect the time step.

While the second measure may not increase the speed substantially, the first one is rather hard to implement. Another approach is needed, and that is what our reformulation of MAC will provide, as we will see in the following.

6.4 F+T with MAC.

6.4.1 Introduction.

The MAC framework may be used to increase the speed of the F+T algorithm presented in the past subsection. Actually, we will now present a different algorithm (not only a numerical optimization) for F+T, that is somehow a synthesis of the aforementioned F+T scheme and the morphing technique introduced in section 5.3. This new algorithm is both faster and more practical for our fMRI applications, as we shall see.

6.4.2 Fundamentals.

It was mentioned earlier that in the coupled system of equation 43 and for our fMRI applications, the maximum time step to ensure stability of the numerical implementation was generally imposed by the tracking equation (the second equation of the system). This is due to the facts that:

1. the velocity in the tracking equation is external and
2. it is proportional to the (main) curvatures in V: its maximum (which determines the time step) is not only large but also does not decrease monotonically as it happens with mean curvature evolution of planar shapes (consider for instance that in the 3D case connected non-convex shapes may get disconnected during the smoothing).

We recall from section 3.6.2 that the MAC segmentation algorithm has an ever decreasing velocity for its equations. Besides, the maximum velocity is achieved at $t = 0$ as the maximum difference between the functions I_1 and I_2. If we could write the flattening equation (the first equation in the system F+T) within the MAC framework, as an evolution of volume V towards a (known) volume V_2, then we would have an ever decreasing speed with arbitrary initial value (given by our choice of the "binary" volumes V and V_2). In other words, the tracking equation would still have an external velocity, but it would be as small as we wanted.

Let us see how we can turn our flattening equation into an evolution from one known volume into another.

6.4.3 The algorithm.

The following system of coupled PDE's is the MAC formulation for the F+T problem:

$$\begin{cases} V_{1_t} = (V_2 - V_1)\,|\nabla V_1| \\ U_t = (V_2 - V_1)\frac{\nabla V_1}{|\nabla V_1|} \cdot \frac{\nabla U}{|\nabla U|}\,|\nabla U| \end{cases} \tag{52}$$

Note that this equation is just equation 43 where we have replaced β by $(V_2 - V_1)$, and named V as V_1. So firstly we can state that this equation will perform a certain deformation on our initial (non-smooth) volume V_1 and that, if we constructed U as in the previous subsection, then U will track a given set of curves on the zero-level set of V_1 along the evolution. Therefore, equation 52 performs a certain deformation *and* tracking. Now we want to ensure that this deformation is actually a flattening of the zero-level set of V_1. Can this be done?

We only need to choose the right V_2. And that is precisely our smoothed V_1. The first of the equations of the system is just a *morphing* equation, as seen in section 5, and therefore it makes volume V_1 evolve into volume V_2. Since V_2 is the flattened version of V_1, then the deformation in equation 52 is a flattening of V_1. Thus, equation 52 performs F+T.

6.4.4 Advantages of this formulation.

Now that we have proved that equation 52 performs the desired F+T operation, let us see its advantages as respect to the formulation given in equation 43.

Firstly, one could argue that it is not necessarily faster since we need to compute V_2 prior to this evolution, and V_2 requires to perform a certain number of iterations of a smoothing evolution of V_1. But for the fMRI applications we are working with, where the surface (zero-level set of V_1) is very wrinkled, we have that:

- the independent (that is, not coupled) smoothing of $V_1$is very fast, either using mean curvature or the flow proposed in equation 42; for our purposes, a flattened surface can be obtained within $\approx$130 iterations.
- since our volumes V_1 and V_2 can be "binary", in the sense introduced in section 5.3 that we only require that they have as level set a certain surface, then the deformation of V_1 into V_2 is also very fast, and we can stop it after $\approx$30 iterations; in practice, our volumes will be binary actually, since the computer "interprets" any function as being Lipschitz.

Therefore, convergence in our applications is obtained after $\approx$160 iterations. If we wanted to implement F+T with equation 43, the number of iterations required would be an order of magnitude greater, of $\approx$2000 (because the time step would be fixed by the slower tracking equation). The speed increase is therefore evident, at least for the volumes we have worked with.

But there is still another advantage of this formulation, and that is that, in practice, the user (the neurobiologist) may want to mark a certain area on the cortical surface, see its mapping on the flattened representation, then mark another area and see its mapping, and so on. If we wanted to do this with the original F+T formulation, we would need N iterations for each area. In the MAC F+T formulation we would need $M + 30$ iterations for the first area and 30 for any of the following, where the flattening would require M iterations but it would only be needed for the first area (since for the rest we would already have the volume V_2). So, unless $N \ll 30$, the MAC F+T scheme is more convenient.

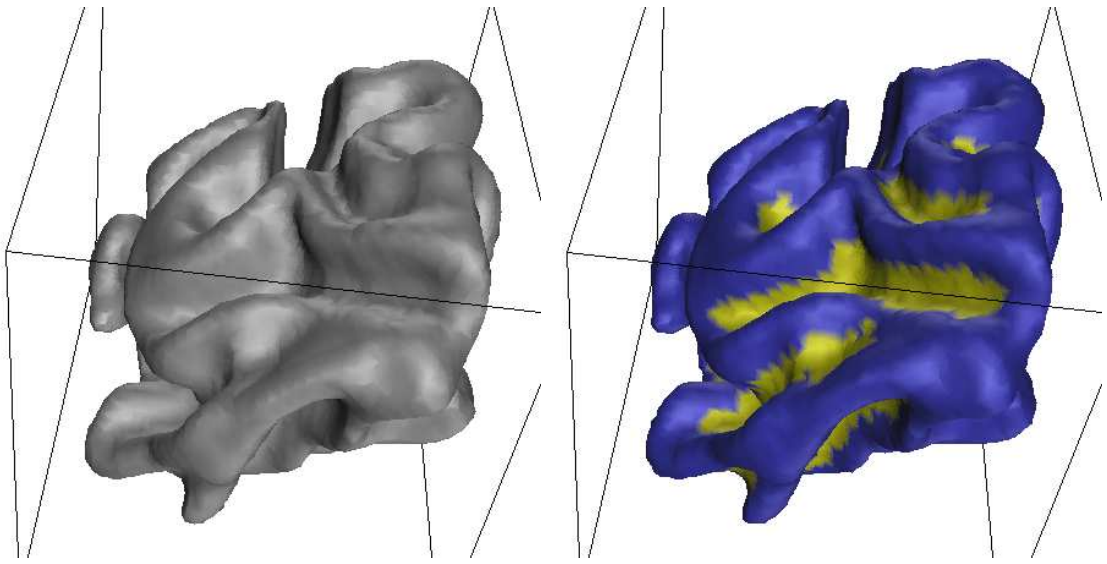
Figure 39: Three dimensional view of a cortical surface. The left image shows the volume, the right one shows the volume with superimposed color information (that could correspond to neural activity, for instance).

6.4.5 Examples.

As an example of the possibilities of this formulation we now procceed to flatten the volume shown in figure 39, while tracking the evolution of the yellow (or light grey in greyscale) areas on the blue (or dark grey) surface.

These areas can be specified at will by the user, as we already mentioned, but here we chose to track the "valleys", the areas where both main curvatures are negative. The specifying of an area on the volume's surface is a task that would require a somehow sophisticated GUI, and in this particular case the boundaries would be very hard to determine manually. That is why we directly constructed the volume U in such a way that its zero-level set intersects the zero-level set of the original volume V on the valleys' boundaries. This can be accomplished if we define U as

$$U(x,y,z) = \begin{cases} sg(\kappa_1 + \kappa_2) \cdot max(|\kappa_1|, |\kappa_2|), & \kappa_1\kappa_2 > 0 \\ 0 & else \end{cases} \tag{53}$$

where the curvatures are computed on $V(x,y,z)$.

So, given the volume shown on the left in figure 39, we construct U in the aforementioned way and we can mark the valleys as shown on the right image of the same figure.

Then we procceed to flatten V, obtaining Vf. And finally we employ the F+T MAC algorithm to morph V into Vf while tracking U. We obtain Uf, and then we can superimpose the tracked valleys on Vf. Note that Uf is not a flattened version of U, and that the areas superimposed on Vf are not valleys of Vf, but the result of the evolution of the valleys on V.

Figures 40 to 41 show the results for several Vf, each of them a more smoothed version of V: the neurobiologist would only require to perform the operation on the last one, but here we computed the intermediate volumes just to exemplify. Each image shows a volume that has been flattened for 10 iterations more than its predecessor. The last Vf took 130 iterations to compute, while each morphing required (less than) 30 steps.

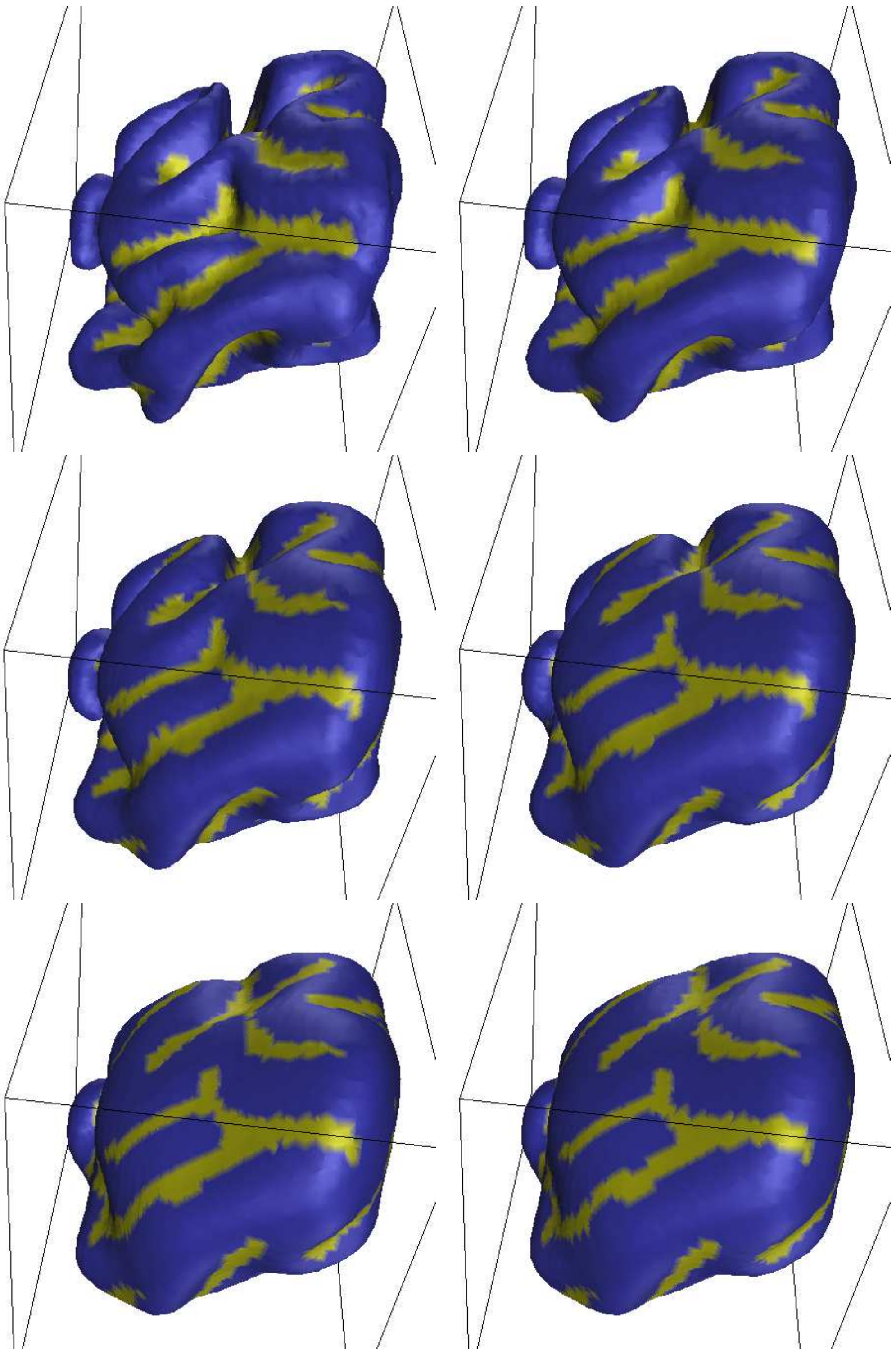

Figure 40: Flattening and Tracking of the volume shown in figure 39. Each image shows a volume that has been flattened for 10 iterations more than its predecessor (see text).

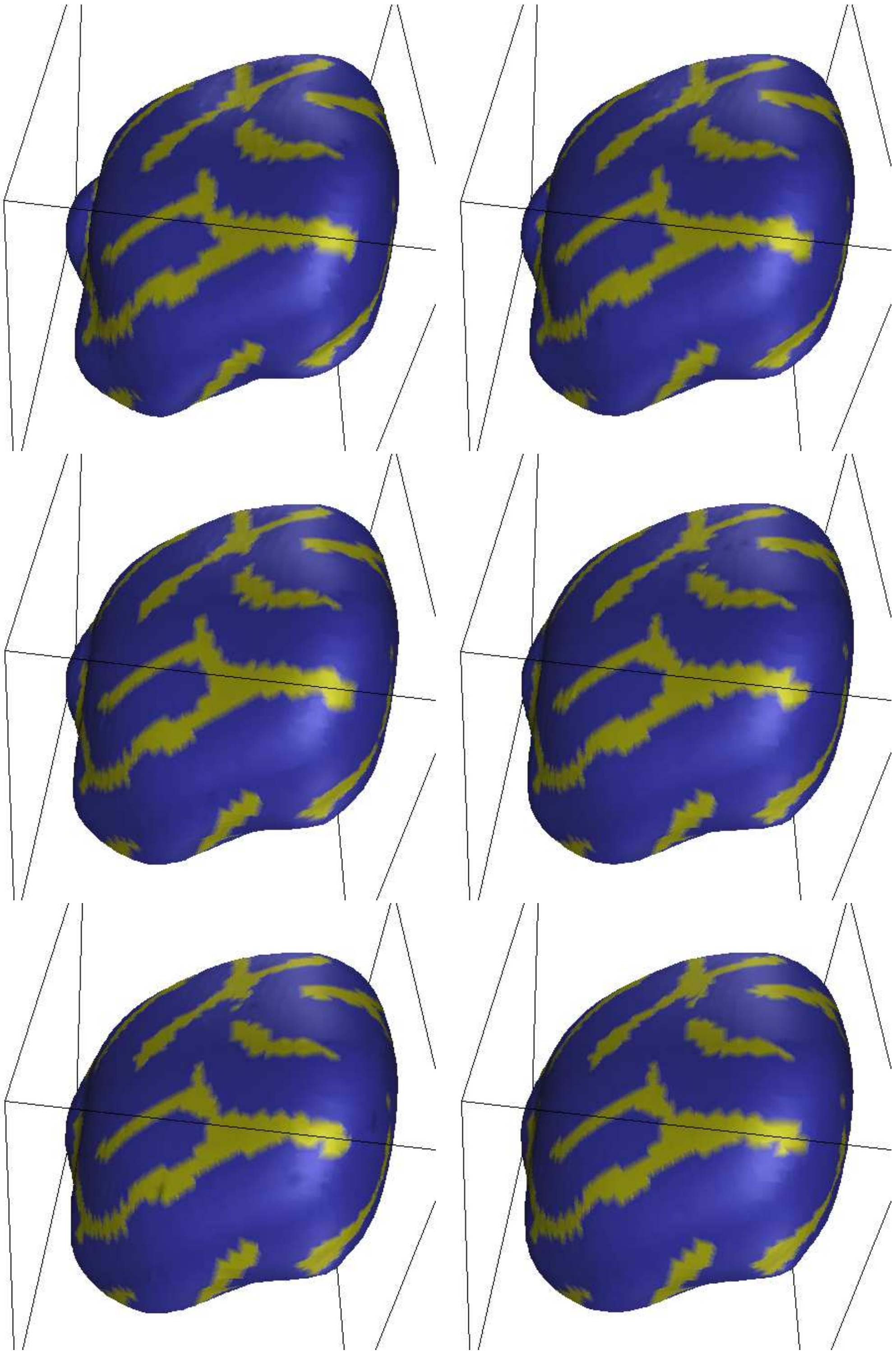

Figure 41: Flattening and Tracking of the volume shown in figure 39. Each image shows a volume that has been flattened for 10 iterations more than its predecessor (see text).

7 Conclusions and future work.

In this work a novel system of coupled PDE's has been presented. The first PDE deforms an image (of arbitrary dimension) towards another one, while the additional PDE tracks the deformation of the curves of interest of the first image until they converge to the desired position in the second one.

This framework can be adapted for applications in automatic segmentation of sequences, tracking, interpolation and morphing, with independence of the topology. It can also improve the performance of an algorithm, introduced in this work as well, that performs flattening of cortical surfaces while keeping track of certain areas on them.

There are a number of directions in which we can continue this work. Formal mathematical results concerning existence and uniqueness of the flow must be derived, with the use of the Theory of Viscosity Solutions. The problem of non-convergence from the first image to the second must be addressed, finding a description of the kind of images for which the convergence is not achieved.

The use of singular value decomposition and principal components analysis became very popular in computer vision and image processing in the past years. The basic idea is to represent a given event as a linear combination of principal components from learned events. We can see the technique here described as a first step toward the deformation of principal components. That is, we can look at the curve obtained from the current image as a principal component. We are currently investigating the extension of this technique to the deformation of a number of principal components, thereby representing a given event as a combination of *deformed* learned principal components. The deformations will be obtained as a system of coupled PDE's.

The equations introduced in this paper are basically "short in memory," that is, only the present frame is used to segment the next one, and to interpolate in-between two consecutive images only those images are used. We can incorporate past information to these equations, in the form of optical flow or Kalman filtering, in order to improve the detection and interpolation results. This will be the subject of further study as well.

Finally, in morphing applications future work should allow the user to specify correspondant pairs of points on the two objects, and contemplate the extension of the scheme to gray-level images (the 2D case being the most interesting, of course).

References

[1] W. F. Ames, “Nonlinear PDE's in Engineering”, volumes 1-2, Academic Press, New York, San Francisco, London, 1972.

[2] S. Angenent, “Some recent results on mean curvature flows”, Recent advances in PDE's, M. A. Herrero and E. Zuazua eds., John Wiley and Sons, 1994.

[3] G. Barles, “Remarks on a flame propagation model”, Technical Report #464, INRIA Rapports de Recherche, 1985.

[4] Bart M. ter Haar Romeny, ed., “Geometry Driven Diffusion in Computer Vision”, Kluwer Academic Plubishers, 1994.

[5] J. Bigün, “Speed, frequency and orientation tuned 3D Gabor filter banks and their design”, Proceedings IAPR-94, Jerusalem, Oct. 1994.

[6] , I. Carlbom, D. Terzopoulos and K. M. Harris, "Computer-Assisted Registration, Segmentation and 3D Reconstruction from Images of Neuronal Tissue Sections", IEEE Transactions on Medical Imaging, vol 13, pp 351-362, 1994.

[7] V. Caselles, V Catte, F. Coll, T. and Dibbos F., “A geometric model for active contours”, Numersiche Mathematik, 66:1-31, 1993.

[8] V. Caselles and C. Sbert, “What is the best casual space for 3D images”, SIAM J. App. Maths.,vol. 36, no.4, pp. 1196-1246, august 1996.

[9] V. Caselles, R. Kimmel and G. Sapiro, “Geodesic Active Contours”, International Journal on Computer Vision 22(1), 61-79, 1997.

[10] Y. G. Chen, Y. Giga and S. Goto, “Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations”, J. Diff. Geometry, 33:749-786, 1991.

[11] B. Chow, “Deforming convex hypersurfaces by the nth root of the Gaussian curvature”, J. of Diff. Geom. 22, pp. 117-138, 1985.

[12] L. Cohen and R. Kimmel, “Global minimum for active contour models: a minimal path approach”, International Journal of Computer Vision, submitted.

[13] M.G. Crandall, H. Ishii and P.L. Lions, “User's guide to viscosity solutions of second order partial linear differential equations”, Bulletin of the American Math. Society, 27:1-67.

[14] A. Dale and M. Sereno, “Improved localization of cortical activity by combining EEG and MEG with MRI cortical surface reconstruction: a linear approach.”, Journal of Cognitive Neuroscience, 5(2):162-176,1993.

[15] D. DeCarlo and J. Gallier, “Topological evolution of surfaces”, Graphics Interface 96, pp. 194-203, ??, 1996.

[16] R. Deriche and O. Faugeras, “Les EDP en traitement des images et vision par ordinateur”, Rapport de Recherche #2697, INRIA, 1995.

[17] M. P. Do Carmo, “Differential Geometry of curves and surfaces”, Prentice-Hall: Englewood Cliffs, 1976.

[18] H. Drury, D. Van Essen, C. Anderson, C. Lee, T. Coogan and J. Lewis, “Computerized mappings of the cerebral cortex: a multiresolution flattening method and a surface-based coordinate system”,Journal of Cognitive Neuroscience, 8(1):1-28, 1996.

[19] D. Dunn, E. Higgins and J. Wakeley, “Texture segmentation using 2D Gabor elementary functions”, IEEE PAMI, vol. 16, number 2, february 1994.

[20] S. Engel, G. Glover and B. Wandell, “Retinotopic organization in human visual cortex and the spatial precision of fMRI”, to appear in Cerebral Cortex, 1997.

[21] C. L. Epstein and M. Gage, “The curve shortening flow” in “Wave Motion: Theory, Modelling and Computation”, A. Chorin and A. Majda, eds., Springer-Verlag: NewYork, 1987.

[22] M. Gage, “An isoperimetric inequality with applications to curve shortening”, Duke Math. J. 50: 1225-1229, 1983.

[23] M. Gage, “Curve shortening makes convex curves circular”, Invent. Math. 76: 357-364, 1984.

[24] M. Gage and R. S. Hamilton, “The heat equation shrinking convex plane curves”, J. Diff. Geom. 26: 285-314, 1987.

[25] M. Grayson, “Shortening embedded curves”, Ann. Math. 129: 71-111, 1989.

[26] M. Grayson, “A short note on the evolution of a surface by its mean curvature”, Duke Math. Journal, pp. 555-558, 1989.

[27] M. Grayson, “The heat equation shrinks embedded plane curves to round points”, J. Diff. Geom. 26: 285-314, 1987.

[28] H.W. Guggenheimer, “Differential Geometry”, McGraw-Hill, New York, 1963.

[29] D. Heeger and J. Bergen, “Pyramid based texture analysis/synthesis”, ??.

[30] G. Huisken, “Flow by mean curvature of convex surfaces into spheres”. J. Diff. Geom. 20, pp. 237-266, 1984.

[31] M. Kass, A. Witkin and D. Terzopoulos, “Snakes: Active contour models”, International Journal of Computer Vision, 1:321-331, 1988.

[32] B. Kimia, A. Tannenbaum and S. Zucker, “Shapes, Shocks and Deformations I: The Components of Two-Dimensional Shape and the Reaction-Diffusion Space”, International Journal of Computer Vision, 15, 189-224 (1995).

[33] P. D. Lax, “Hyperbolic systems of conservation laws and the mathematical theory of shock waves”, SIAM Regional Conference series in Applied Mathematics, Philadelphia 1973.

[34] T. Lee, D. Mumford and A. Yuille, “Texture segmentation by minimizing vector-valued energy functionals: the coupled membrane model”, Lecture Notes in Computer Science, Vol. 588, Springer Verlag, Berlin, 1992.

[35] L. Alvarez, P.˜L. Lions, and J.˜M. Morel, “Image selective smoothing and edge detection by nonlinear diffusion,” SIAM J. Numer. Anal, pp. 845-866, 1992.

[36] W. Lorensen and H. Cline, “Marching cubes: a high resolution 3D surface reconstruction algorithm”, Computer Graphics, Volume 21, Number 4, July 1987.

[37] R. Malladi, J.A. Sethian, B.C. Vemuri, “Evolutionary fornts for topology independent shape modeling and recovery”, Proc. of the 3rd ECCV, Stockholm, Sweden, pp. 3-13, 1994.

[38] R. Malladi, J. Sethian and B. Vemuri, “Shape Modeling with front propagation: a level set approach”, IEEE Trans. on PAMI, 17:158-175, 1995.

[39] Kevin Montgomery and Muriel Ross, "A Method for Semiautomated Serial Section Reconstruction and Visualization of Neural Tissue from TEM Images", SPIE-93, San Jose, California, 1993.

[40] , Kevin Montgomery and Muriel Ross, "Improvements in Semiautomated Serial Section Reconstruction and Visualization of Neural Tissue from TEM Images", SPIE-94, San Jose, California, 1994.

[41] P. Olver, G. Sapiro and A. Tannenbaum, “Invariant Geometric evolutions of surfaces and volumetric smoothing”, SIAM J. of App. Math, ??, 1994.

[42] P. Olver, G. Sapiro and A. Tannenbaum, “Affine Invariant Edge Maps and Active Contours”, IMA Preprint Series #1360, 1995.

[43] S. Osher and J. Sethian, “Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations”, Journal of Computational Physics, 79:12-49, 1988.

[44] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion”, IEEE PAMI 12, pp 629-639, 1990.

[45] M. Proesmans, E. Pauwels and L. van Gool, “Coupled geometry-driven diffusion equations for low-level vision”, in [4].

[46] , G. Randall, A. Fernández, O. Trujillo, P. Morelli, F. Malmierca, G. Apelbaum, M. Bertalmío, L. Vázquez. "Image enhancement for a low cost TEM acquisition system". Proceedings of SPIE, Vol. 3621, California, 1998.

[47] , G. Randall, A. Fernández, O. Trujillo, P. Morelli, F. Malmierca, G. Apelbaum, M. Bertalmío, L. Vázquez. "Neuro3D: an interactive 3D reconstruction system of serial sections using automatic registration". Proceedings of SPIE, Vol. 3621, California, 1998.

[48] J. C. Russ, “The image processing handbook”, 2nd edition, CRC Press, 1994.

[49] G. Sapiro and A. Tannenbaum, “Affine invariant scale-space”, International Journal of Computer Vision, 11(1):25-44, 1993.

[50] G. Sapiro, V. Caselles, “Histogram modification via PDE’s”, HP Laboratories Technical Report, June 1995.

[51] J. A. Sethian, “Numerical ALgorithms for propagating interfaces: Hamilton-Jacobi equations and conservation laws”, Int. Journal on Diff. Geom. (1990) 131-161.

[52] E. Simoncelli, W. Freeman, E. Adelson and D. Heger, “Shiftable multi-scale transforms”, IEEE Trans. on Infornation Theory, vol. 38 (2), pp. 587-607, March 1992.

[53] F. Sjostrand,"Electron Microscopy of Cells and Tissues", Academic Press, New York, 1967.

[54] J. Smoller, “Shock waves and reaction-diffusion equations”, Springer-Verlag, New York, 1993.

[55] H. M. Soner and P.E. Souganidis, “Singularities and uniqueness of cylindrically simetric surfaces moving by mean curvature”, Comm. in PDE’s 18, pp. 859-894, 1993.

[56] P. Teo, G. Sapiro and B. Wandell, “Anatomically consistent segmentation of the human cortex for fMRI visualization”, HP Laboratories Technical Report, HPL-97-03, january 1997.

[57] J. I. E. Urbas, “An expansion of convex hypersurfaces”, J. Diff. Geom. 33, pp. 91-125, 1991.

[58] J. I. E. Urbas, “Correction to ’An expansion of convex hypersurfaces’ ”, J. Diff. Geom. 35, pp. 763-765, 1992.

[59] L. Vázquez, G. Sapiro and G. Randall, “Segmenting neurons in electronic microscopy via geometric tracking”, IEEE-ICIP98, Chicago, 1998.

[60] B. Wandell, S. Engel and H. Hel-Or, “Creating images of the flattened cortical sheet”, Invest. Opth. and Vis. Sci, 36(S612), 1996.