Deformable Model with non-Euclidean Metrics

Benjamin Taton and Jacques-Olivier Lachaud

Laboratoire Bordelais de Recherche en Informatique (LaBRI)
351, cours de la Libération, 33405 TALENCE, FRANCE

Abstract. Deformable models like snakes are a classical tool for image segmentation. Highly deformable models extend them with the ability to handle dynamic topological changes, and therefore to extract arbitrary complex shapes. However, the resolution of these models largely depends on the resolution of the image. As a consequence, their time and memory complexity increases at least as fast as the size of input data. In this paper we extend an existing highly deformable model, so that it is able to locally adapt its resolution with respect to its position. With this property, a significant precision is achieved in the interesting parts of the image, while a coarse resolution is maintained elsewhere. The general idea is to replace the Euclidean metric of the image space by a deformed non-Euclidean metric, which geometrically expands areas of interest. With this approach, we obtain a new model that follows the robust framework of classical deformable models, while offering a significant independence from both the size of input data and the geometric complexity of image components.

Keywords: image segmentation, deformable model, non-Euclidean geometry, topology adaptation, optimization

1 Introduction

Deformable models were first introduced in the field of image segmentation by Kass, Witkin and Terzopoulos [5]. Their behavior is governed by the minimization of a functional (*energy*), which depends both on the matching of the curve to the contour and on the smoothness of the curve. This segmentation framework remains valid even with poor quality images with weak contours. Furthermore, the energy formulation is rather intuitive. This makes it easier to include user interaction as well as other kinds of constraints in the segmentation process. The deformable model framework has been extended with various tools that improve both convergence speed and segmentation quality (e.g. [13] and [2]).

However, deformable models in their original formulation cannot dynamically change their topology. This means that they can only extract image components with the same topology as the initial model. As a consequence, the model initialization requires hypotheses about its final expected shape. In domains like biomedical image analysis, this kind of hypotheses is sometimes difficult to justify. Indeed, the objects in the images often have complex shapes with several connected components and holes. Furthermore they are likely to be abnormal and hence may have an unexpected topology.

Several methods have been proposed to overcome this limitation. These *highly deformable models* are able to achieve automated or supervised topology adaptations, based on their geometric evolution. Nevertheless, although they largely extend the application field of deformable models, these new methods come in general with an important increase of computational costs. The following paragraphs describe some of these techniques and emphasize their time complexities. From now on we will assume that the size of a bi-dimensional image is n^2 and that the number of vertices necessary to describe the shape of the image components is $O(n)$.

The first approach keeps the energy formulation as well as the explicit shape representation. The idea is to check the model at each step of its evolution. Then, if its topology is no longer consistent, mesh reconfigurations are performed to solve the problems. Delingette and Montagnat [4] analyse the intersections of their model with a regular grid. Detecting special configurations allows them to detect and handle topological changes. McInerney and Terzopoulos [9] propose a similar method that use a simplicial regular subdivision of the space to detect and solve topological problems. This algorithm is less efficient than Delingette's ($O(n^2)$ per iteration instead of $O(n)$), but has the advantage of being easy to extend to the three-dimensional case. The authors of [6] propose a different approach based on distance constraints on the edges of the deformable model. Although they formulate it for the three-dimensional case, the principle is valid for the bi-dimensional case too. With an appropriate data structure the time complexity of this algorithm is reduced to $O(n \log n)$ per iteration.

Another approach [10] consists in formulating the problem in term of front propagation instead of minimizing an energy. In this context the model is no longer represented explicitly: it is viewed as a particular level set of a scalar function f defined on the image space and which evolves with the time. This level set propagates in the image space with respect to two main constraints: (i) the propagation slows down in the neighborhood of high image gradients, (ii) the level set propagates faster in places where its curvature is important (this is to preserve the contour smoothness, see [1] and [8] for details). These constraints are expressed as differential equations involving f. Iteratively solving these equations makes the level set approach image components. With this formalism, the topological changes are automatically embedded in the evolution of f. In addition it is very easy to extend this kind of model to higher dimensional spaces. Nevertheless these advantages come with heavy computational costs: theoretically, the new values of f have to be re-computed over the whole image at each step of the algorithm (each step would thus cost $O(n^2)$ operations). However optimizations based on quad-trees or on the narrow band algorithm may reduce this complexity to $O(n \log n)$ [11, 12]. Note also that, user interactions or other constraints are more difficult to embed in this kind of models compared to explicit ones.

Finally, in a third approach, the energy minimization problem is formulated in a purely discrete context [7]. Image components are viewed as sets of pixels (or voxels). There is therefore no need to write specific algorithms to handle

topology changes. The energy of the components is computed according to the local configuration of each element of its boundary (called a *bel*). The model evolution is then performed by adding or removing pixels (or voxels) so as to minimize the global energy. Complexity per iteration linearly depends on the size of the model boundary. Namely, for a bi-dimensional image the time cost is $O(n)$ bel energy computations for each iteration. However, the time needed to compute one bel energy is constant but rather important.

This argumentation shows that the computational costs of highly deformable models largely depend on the size of input data. The images produced by acquisition devices have higher and higher resolutions. As a consequence, computational cost of these segmentation algorithms becomes prohibitive. In this paper we propose a highly deformable model which dynamically adapts its resolution depending on its position in the image, and automatically changes its topology according to the changes of its geometry. A fine resolution is thus achieved in the interesting parts of the image while a coarse one is kept elsewhere. With this technique, the complexity of the model is made significantly more independent of the image resolution.

More precisely, the model we propose is an extension of the one presented in [6]. In this model, the topological consistency is maintained using distances estimations. However, to work properly, the model needs to have a regular density. To achieve adaptative resolution, our idea is to change the Euclidean metric with a locally deformed metric that geometrically expands the interesting parts of the image. The original model was designed to work with three-dimensional images. In this paper, while we propose to extend a bi-dimensional version of this model, our real goal is to later extend it to segment three-dimensional images. A particular attention is thus paid to using only methods that remain available in three-dimensional spaces.

In the first section of this paper, we describe more in details the model we propose to extend. In the second section we define a more general notion of distance that is used in the third section to define our new model. The last section deals with the choice of the metric with respect to the image.

2 Initial Model

In this section we first make some recalls about the classical snake model, as it was introduced by Kass, Witkin and Terzopoulos [5]. Then we describe the highly deformable model we propose to extend. The last part of the section explains how changing the metric of the image space allows us to locally change the resolution of this model.

2.1 Classical Snake Model Formulation

Snakes are defined as curves $\boldsymbol{C} : [0,1] \to \mathbb{R}^2$ that evolve in the image space so as to minimize an energy functional expressed as

$$\mathcal{E}(\boldsymbol{C}) = \int_0^1 \mathcal{E}_{image}(\boldsymbol{C}(u)) + \mathcal{E}_{internal}(\boldsymbol{C}, u)\, du\,. \tag{1}$$

The energy functional is composed of two terms: the first term $\mathcal{E}_{image}$ ensures that the curve matches the contours of the image, the second term $\mathcal{E}_{internal}$ forces the curve to remain rather smooth. The contour matching is ensured by choosing $\mathcal{E}_{image}$ so as to make it low in the neighborhood of image gradients. The smoothness of the curve is obtained by penalizing both its length and curvature. These remarks lead to the following choice for these energies

$$\begin{cases} \mathcal{E}_{image}(x,y) = -\|\nabla G * I(x,y)\| \\ \mathcal{E}_{internal}(\boldsymbol{C},u) = \frac{\alpha}{2}\left\|\frac{\boldsymbol{dC}}{du}\right\|^2 + \frac{\beta}{2}\left\|\frac{\boldsymbol{d^2C}}{du^2}\right\|^2 \end{cases}$$

where I and G respectively denote the image function and a Gaussian smoothing filter.

To find the curve that minimizes (1), the most common method consists in discretizing the curve. Once this has been done, each discretized point is considered as a particle having its own energy, and therefore evolving under forces derivating from this energy. Namely these forces are

- an image-based force which attracts the particle towards image gradients,
- an elastic force which attracts each particle towards its neighbors,
- a curvature minimizing force which attracts each particle toward the straight line joining its neighbors,
- a friction force that makes the system more stable.

This point of view has a significant advantage over the first formulation: it makes it possible to introduce new forces that speed up the convergence of the model or improve the segmentation (for examples see [13] and [2]). Moreover, such a mechanical system is easy to extend to three dimensions [3, 6].

To determine the displacement of each particle at a given step, many authors directly apply Newton's laws of motion and then solve the resulting differential equations (*e.g.* [13, 2, 4]). However, we propose to use the Lagrangian approach to express the dynamics of each particle, since it can be formulated in non-Euclidean spaces.

Now, consider a particle of mass m, its position being described by $\boldsymbol{x} = (x_1, \ldots, x_n)$, moving under the action of the force $\boldsymbol{F}$. Let T denote its kinetic energy ($T = \frac{1}{2} m\ \dot{\boldsymbol{x}} \cdot \dot{\boldsymbol{x}}$). Let $\boldsymbol{\delta x}$ be a small variation of the particle trajectory ($\boldsymbol{\delta x}(t_1) = \boldsymbol{\delta x}(t_2) = \boldsymbol{0}$). Then, the dynamics of the particle is described by the least action principle which postulates (for all possible $\boldsymbol{\delta x}$):

$$\int_{t_1}^{t_2} (\delta T + \boldsymbol{F} \cdot \boldsymbol{\delta x})\, dt = 0\,. \tag{2}$$

In particular, it has to be true for $\boldsymbol{\delta x} = (0, \ldots, 0, \delta x_i, 0, \ldots, 0)$. Equation (2) may then be written as

$$\forall i,\ \int_{t_1}^{t_2} \left(\frac{\partial T}{\partial x_i}\delta x_i - \frac{d}{dt}\frac{\partial T}{\partial \dot{x}_i}\delta x_i + \boldsymbol{F} \cdot \boldsymbol{\delta x} \right) dt = 0\,. \tag{3}$$

In an Euclidean space with the usual dot product, $\boldsymbol{F} \cdot \boldsymbol{\delta x} = F_i \delta x_i$. Hence equation (3) leads to

$$\forall i,\ \frac{\partial T}{\partial x_i} - \frac{d}{dt}\frac{\partial T}{\partial \dot{x}_i} + F_i = 0\ . \tag{4}$$

Replacing T with $\frac{1}{2}m\sum_{i=1}^{n}\dot{x}_i^2$ (which is true in Euclidean spaces only) and writing the equations for each possible value of i result in the well known Newton's laws of motion: $m\ddot{\boldsymbol{x}} = \boldsymbol{F}$. The Lagrangian formalism may thus be considered as a variational principle equivalent to the Newton's laws, but with the advantage of being independent of the Euclidean structure of the space.

2.2 Highly Deformable Model

As said in the introduction, classical snake models are unable to handle topological changes. Highly deformable models are extensions to the basic snake model that overcome this limitation. This section describes the bi-dimensional version of the highly deformable model we propose to extend [6], and how it is able to dynamically adapt its topology to keep it consistent with its geometry.

General Description. Our deformable model is an oriented (not necessarily connected) closed polygonal curve that evolves in the image space. To preserve a consistent orientation of the model, it is necessary to avoid self-intersections. The main idea is to detect them before they occur. When a problem is expected to arise, it is solved using appropriate local topological operators. By this way, we ensure that the orientation of the model remains consistent at each step of its evolution. The following subsections detail each step of this algorithm.

Collision Detection. Detection is made possible by imposing distance constraints between the vertices of the model. Let δ and ζ be real numbers satisfying $0 < \delta$ and $2 \leq \zeta$. Then, suppose that for each edge (a, b) of the model, the Euclidean distance $d_E(a, b)$ between a and b satisfies

$$\delta \leq d_E(a, b) \leq \zeta\delta\ . \tag{5}$$

That means that all vertices are rather evenly spaced on the model mesh (provided that ζ is close to 2). Now let us show how this is helpful to detect self intersections of the model.

Let (u, v) be an edge of the deformable model. Suppose that a vertex p crosses over (u, v). Then it is easy to verify that there is a time interval during which the distance from p to the edge (u, v) is less than a particular threshold. It follows that, during this time interval, the distance from p to either u or v is less than $\lambda_E\zeta\delta$, where λ_E is an appropriate real number (see Fig. 1a-b).

Theoretically, without additional hypothesis, λ_E should be greater than 1 to ensure that no self-intersection may occur without being detected. However, if the initial model configuration is consistent, and if the motion of the particles

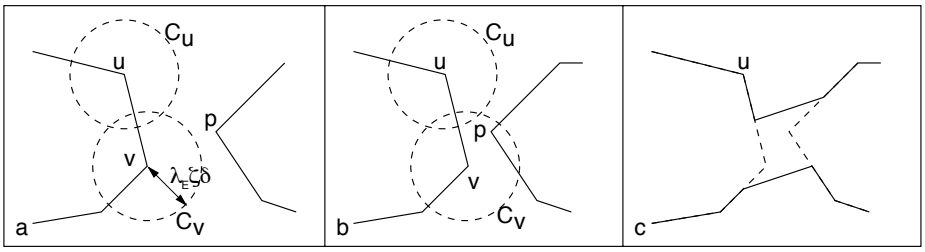


Fig. 1. Collision detection and handling: a point p cannot crosses over (u, v) without entering circle C_u or C_v (a) and (b). When a collision is detected (b), a local reconfiguration is performed (c): new vertices are created around v and p. These vertices are connected in the appropriate way. Then the vertices v and p are deleted.

is small enough, smaller values of λ_E may be chosen. In particular, if the vertex speed is assumed to be less than v_{max}, $\lambda_E = 1/2 + v_{max}/\zeta\delta$ is a suitable choice.

With this property, if the inequality (5) is satisfied at each step of the evolution of the model, then self-intersections are detected by checking the inequality

$$\lambda_E \zeta\delta \leq d_E(u, v) \tag{6}$$

for each pair of non-neighbor vertices of the model. Each time this inequality is not verified, a local reconfiguration of the model is performed in order to restore the constraint. Let us describe these reconfigurations.

Local Reconfigurations. As said in the previous paragraph, two kinds of constraints have to be maintained:

- constraints between neighbor vertices, given by (5),
- constraints between non-neighbor vertices, given by (6).

Consider an edge of the deformable model that does not respect the constraint (5). If it is shorter than the δ threshold, then the constraint is recovered by merging the two vertices at the extremities of the edge. Symmetrically, if the edge is longer than the $\zeta\delta$ threshold, then it is split into two equal edges by inserting a new vertex in the middle of the original edge. Note that this kind of transformations affect the local resolution of the model, but not its topology.

If we now consider a pair of vertices that does not respect (6), then the transformations described on Fig. 1c have to be performed. The topology of the model is thus changed. Detecting all these pairs of vertices with a naive method would have a time complexity of $O(n^2)$ for a model with $O(n)$ vertices. Nevertheless a hierarchical subdivision of space with quad-trees reduces the costs to an average time complexity of $O(n \log n)$.

It is important to note that this approach to topology adaptation also works for three-dimensional image segmentation. In this context, the deformable model is a triangulated surface, and with an adapted value of λ_E, all the statements made before remain true.

2.3 Influence of Distance Estimations

As said before, the evolution of the model, and especially its resolution, largely depend on distance estimations. When the length of an edge is to small, vertices are merged, and when the length of an edge is too high, new vertices are inserted in the mesh. Therefore, the parameter δ determines the resolution of the model: the smaller it is, the greater the resolution becomes.

Since (5) has to be true for each edge of the model, increasing the resolution of the deformable model only in a restricted interesting part of the image cannot be achieved without increasing the resolution everywhere, that is, even where it is not needed. This implies a significant increase of the time and memory costs of the segmentation algorithm.

Suppose now that we change the way distances are measured instead of changing the value of δ. Namely, suppose that distances are overestimated in the interesting parts of the image, while they are underestimated elsewhere. In all areas where distances are overestimated, the lengths of the edges seem to be greater so that they tend to exceed the $\zeta\delta$ threshold. Therefore, new vertices are inserted to split the edges. As a consequence, the model resolution increases. Symmetrically, in places where distances are underestimated, the lengths of the edges seem to be smaller. To prevent the edge length from falling under δ, neighbor vertices are merged, so that the resolution decreases.

The extension we propose is based on these remarks. Our idea is to replace the Euclidean metric with a deformed one, in order to geometrically expand the areas of interest. With this new definition of distances, it becomes possible to perceptibly increase the resolution of the model in restricted parts of the image without significantly increasing the cost of the segmentation algorithm.

Nevertheless, changing the distance notion rises many problems:

- first we have to find a more general definition of distances and to design algorithms to measure these distances between arbitrary points,
- then the motion equations of the deformable model particles have to be rewritten in this context,
- lastly, it is important to discuss how distances are defined with respect to the image.

The following paragraphs successively deal with each of these problems.

3 Riemannian geometry

The new definition of distances is based on Riemannian geometry. Giving all the background on this mathematical theory is beyond the scope of this article, therefore we only describe the notions which are used to define our new deformable model. Moreover, to clarify the presentation, everything will be described in $\mathbb{R}^2$. The given statements however remain true in higher dimensional spaces.

3.1 Metrics

In an Euclidean space, the length of an elementary displacement $\boldsymbol{ds} = (dx, dy)$ is given by the relation

$$||\boldsymbol{ds}|| = dx^2 + dy^2 \ .$$

It only depends on the values of dx and dy. In particular, the origin of the displacement has no influence on its length.

In a non-Euclidean space this is no longer true. Suppose for example that the rectangle $[-\pi, \pi[\times]-\frac{\pi}{2}, \frac{\pi}{2}[$ is the map of a sphere, obtained with the usual parameterization using the longitude θ and latitude ϕ. Then the length of a elementary displacement $\boldsymbol{ds} = (d\theta, d\phi)$ changes whether the displacement occurs near one pole or near the equator of the sphere (actually it may be written $||\boldsymbol{ds}|| = R^2 \cos^2\phi \, d\theta^2 + R^2 d\phi^2$, where R denotes the sphere radius).

To measure the length of arcs in a non-Euclidean space, a mapping which measures the lengths of elementary displacements anywhere in this space must be given. This mapping is derived from the notion of *metric*.

We call a *metric*, or a *Riemannian metric* a mapping g that associates a dot product $g_{(x,y)}$ with each point (x, y) of the space, provided that g is of class C^1. At a given point of the space, the dot product $g_{(x,y)}$ induces a norm which may be used to measure the lengths of elementary displacements starting from (x, y): $||\boldsymbol{ds}|| = g_{(x,y)}(\boldsymbol{ds}, \boldsymbol{ds})^{\frac{1}{2}}$. Then, let $\boldsymbol{c} \ : \ [0,1] \to \mathbb{R}^2$ be an arc. Its lengths $L(\boldsymbol{c})$ may be expressed as

$$L(\boldsymbol{c}) = \int_0^1 \left(g_{\boldsymbol{c}(u)}(\boldsymbol{c'}(u), \boldsymbol{c'}(u))\right)^{\frac{1}{2}} du \ . \tag{7}$$

Metrics are conveniently defined with a matrix form. Indeed, $g_{(x,y)}$ is a dot product, and hence it is a symmetric positive definite bilinear form. In the bi-dimensional case, it may therefore be expressed as

$$(\boldsymbol{u}, \boldsymbol{v}) \to \boldsymbol{u}^T \times \begin{pmatrix} g_{11}(x,y) \ g_{12}(x,y) \\ g_{12}(x,y) \ g_{22}(x,y) \end{pmatrix} \times \boldsymbol{v} \ , \tag{8}$$

where the g_{ij} are functions of class $\mathcal{C}^1$. From now on, in order to make the equations simpler and since no confusion may arise, g_{ij} will denote $g_{ij}(x, y)$. Note that if we choose the metric given by the identity matrix, an elementary displacement $\boldsymbol{ds} = (dx, dy)$ has a length given by $||\boldsymbol{ds}|| = dx^2 + dy^2$. Namely, the Euclidean metric may be interpreted as a particular Riemannian metric.

3.2 New Definition of Distances

Given this metric definition, it becomes easy to build a new definition of the distances between two points: the *Riemannian distance* between two points $\boldsymbol{A}$ and $\boldsymbol{B}$ is denoted $d_R(\boldsymbol{A}, \boldsymbol{B})$ and its value is given by

$$d_R(\boldsymbol{A}, \boldsymbol{B}) = \inf \left\{ L(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathcal{C}^1([0,1], \mathbb{R}^2), \ \boldsymbol{c}(0) = \boldsymbol{A}, \ \boldsymbol{c}(1) = \boldsymbol{B} \right\}$$

where $\mathcal{C}^1([0,1],\mathbb{R}^2)$ denotes the set containing all the arcs of class $\mathcal{C}^1$ in the $\mathbb{R}^2$ space. Informally speaking, the Riemannian distance between two points is the length of one shortest path between these points. When such a path exists, which is always true in a compact space, it is called a *geodesic*.

3.3 Effective Distance Estimations

Although this definition complies with intuition, its major drawback is that computing exact distances is difficult. Indeed, contrary to what occurs in the Euclidean space, shortest paths are no longer straight lines. Finding a minimal path between two given points may be efficiently achieved by using a relaxation method to minimize the discretized version of (7). The length of the discretized path is then easy to compute.

However, one may verify that in an area of the space where g remains constant, geodesics are straight lines. Since g is of class $\mathcal{C}^1$, g may be considered as constant on small neighborhoods of space. Therefore, the geodesic connecting two given points may be viewed as a straight line, provided these two points are close enough from each other (which is the case for neighbor vertices of our model).

4 New Formulation of the Deformable Model

Changing the space metric has two consequences, each of which is described in the following subsections:

- the motion equations are modified, which affects the model dynamics,
- the collision detection method has to take the metric change in account.

4.1 New Laws of Motion

To define the new dynamics of our deformable model, we have to write the differential equations which describe the motion of each particle. Therefore, we rewrite the least action principle (2) with the new dot product defined by the metric. With this dot product, we have:

$$\begin{cases} T = \frac{1}{2} m \dot{\boldsymbol{x}} \cdot \dot{\boldsymbol{x}} = \frac{1}{2} m \sum_{i,j=1}^{n} g_{ij} \dot{x}_i \dot{x}_j \\ \boldsymbol{F} \cdot \boldsymbol{\delta x} = \sum_{i,j=1}^{n} g_{ij} F_i \delta x_j \end{cases} .$$

Choosing $\delta x = (0, \ldots, 0, \delta x_i, 0, \ldots, 0)$ leads to

$$\forall i, \int_{t_1}^{t_2} \left(\frac{\partial T}{\partial x_i} \delta x_i - \frac{d}{dt} \frac{\partial T}{\partial \dot{x}_i} \delta x_i + \sum_{k}^{n} g_{ki} F_k \delta x_i \right) dt = 0 . \tag{9}$$

Once δx_i has been factored, one may deduce

$$\forall i,\ \frac{\partial T}{\partial x_i} - \frac{d}{dt}\frac{\partial T}{\partial \dot{x}_i} + \sum_{k=1}^{n} g_{ki} F_i = 0\ . \tag{10}$$

Using now the new expression of T leads to the motion equations

$$\forall k,\ m\ddot{x}_k + m \sum_{i,j=1}^{n} \Gamma_{ij}^{k} \dot{x}_i \dot{x}_j = F_k\ , \tag{11}$$

where the Γ_{ij}^{k} coefficients are known as the *Christoffel's symbols*, and are given by

$$\Gamma_{ij}^{k} = \frac{1}{2} \sum_{l=1}^{n} g^{kl} \left(\frac{\partial g_{il}}{\partial x_j} + \frac{\partial g_{lj}}{\partial x_i} - \frac{\partial g_{ij}}{\partial x_l} \right)\ , \tag{12}$$

(g^{kl} denotes the coefficient at position (k,l) in the inverse matrix of $(g_{ij})_{1 \leq i,j \leq n}$). Note that the Newton's laws are obtained from (11) by choosing the identity matrix for g_{ij} (*i.e.* by making the space Euclidean).

From these equations our new model can be defined as a set of moving particles each of which follows (11). Its behavior is thus totally determined by the choice of F. As usual F defines the sum of the image force, the elastic force, the curvature force, and the friction force. These forces do not need to be adapted to the Riemannian framework, since the metric is already embedded into the motion equations. The only necessary change is to replace the Euclidean distance measurement with the Riemannian one in the computations of the elastic forces. Of course, new forces such as those proposed in [13] or [2] are likely to be introduced to enhance the behavior of the model.

4.2 Topology Constraints

In this section we show that the constraint (6) remains a good way of detecting where and when topology changes have to be performed. We will thus assume that the regularity constraint (5) written for a Riemannian metric holds:

$$\delta \leq d_R(u,v) \leq \zeta\delta\ , \tag{13}$$

for each edge of the deformable model. Now, let us prove that self-intersections may be detected in the same way as before, namely, by comparing the distances between non-neighbor vertices with a particular threshold.

To do so, we use a relation that compares the Riemannian metrics to the Euclidean metric. We need to consider the lower and upper bounds of the eigenvalues of the metric matrix used in (8). They will respectively be denoted μ_{min} and μ_{max}. With these notations, the relation between the Euclidean and the Riemannian distances is expressed as

$$\mu_{min} d_E(\boldsymbol{M},\boldsymbol{N})^2 \leq d_R(\boldsymbol{M},\boldsymbol{N})^2 \leq \mu_{max} d_E(\boldsymbol{M},\boldsymbol{N})^2\ . \tag{14}$$

This inequality is very useful since we can deduce from it that the constraint (6) can still be used to detect topology changes even in non-Euclidean spaces. Furthermore, it allows us to check efficiently whether this constraint holds.

Let (u, v) be an edge of our deformable model, and p be a vertex that moves towards (u, v). Let m denote the point of (u, v) which minimizes $d_R(m, p)$. Without loss of generality, we may assume that $d_R(m, u) \leq d_R(m, v)$. The triangular inequality is expressed as

$$d_R(p, u) \leq d_R(p, m) + d_R(m, u) \; .$$

From (13) and (14), and from the fact that $d_E(u, m) \leq \frac{1}{2} d_E(u, v)$ we obtain

$$d_R(p, u) \leq d_R(p, m) + \frac{\mu_{max}}{2\mu_{min}} d_R(u, v) \leq d_R(p, m) + \frac{\mu_{max}}{2\mu_{min}} \zeta\delta \; .$$

If p crosses over (u, v), there is a time interval during which $d_R(p, m) \leq \epsilon$. Therefore, comparing $d_R(p, u)$ with $\epsilon + \frac{\mu_{max}}{2\mu_{min}} \zeta\delta$ detects self intersections. If v_{max} is the upper bound of the particles displacement between two steps of the model evolution, a good choice for ϵ is v_{max}. The constraint between non-neighbor vertices is hence expressed as

$$\frac{\mu_{max}}{2\mu_{min}} \zeta\delta + v_{max} \leq d_R(u, v) \; . \tag{15}$$

With $\lambda_R = \frac{\mu_{max}}{2\mu_{min}} + \frac{v_{max}}{\zeta\delta}$ this is written as

$$\lambda_R \zeta\delta \leq d_R(u, v) \; . \tag{16}$$

Note that with the Euclidean metric, $\mu_{min} = \mu_{max}$, so that we get back the value λ_E that was used for the original model.

5 Metric Choice

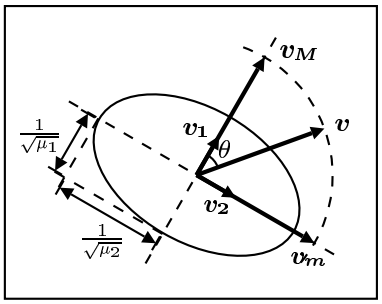
In this section, we explain how the knowledge of the local metric eigenvalues and eigenvectors is useful to properly choose the metric according to the image.

5.1 Metrics Properties

In this paragraph we adopt a local point of view, and it is assumed that metrics are smooth enough to be considered as locally constant.

At a given point of the image space, the metric is given as a symmetric positive definite matrix G with two eigenvectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ and their associated eigenvalues μ_1 and μ_2. If the vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are choosen with a unitary Euclidean length, then the Riemannian length of $\boldsymbol{x} = x_1 \boldsymbol{v_1} + x_2 \boldsymbol{v_2}$ is given as

$$||\boldsymbol{x}||_R = \sqrt{\mu_1 x_1^2 + \mu_2 x_2^2} \; . \tag{17}$$



The unit ball induced by the norm G is an ellipse with its semi axes oriented according to $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. The lengths of the semi axes are given by $1/\sqrt{\mu_1}$ and $1/\sqrt{\mu_2}$. The length of a vector depends on its direction:
$||\boldsymbol{v_M}||_R = ||k\boldsymbol{v_1}||_R = k\sqrt{\mu_1}$
$||\boldsymbol{v_m}||_R = ||k\boldsymbol{v_2}||_R = k\sqrt{\mu_2}$
$||\boldsymbol{v_\theta}||_R = k\sqrt{\mu_1 \cos^2\theta + \mu_2 \sin^2\theta}$

Fig. 2. Local behavior of metrics

Then, as shown in Fig. 2, the local unit ball is an ellipse with an orientation and a size that depend on the local spectral decomposition of G. One may easily verify that the unit balls of a metric with two equal eigenvalues are Euclidean unit balls up to a scale factor. These metrics are thus called *isotropic metrics*, while metrics with two different eigenvalues are called *anisotropic metrics*.

Reciprocally, it is easy to get the symmetric positive definite matrix corresponding to given eigenvectors and eigenvalues. The eigenvectors are used to define locally the main directions of the metric, and the eigenvalues determine to what extent the space is expanded along to these directions. In the next section we explain how we use these statements to choose the image space metrics.

5.2 Metric Definition over the Image

Isotropic Metrics. Isotropic metrics are the easiest to use and define (they may indeed be written $g(x, y) \times I$, where I denote the identity matrix). Therefore, in a first approach, we will only use metrics of this kind. Two methods to define metrics are then considered.

The first method is to let the user himself choose the interesting parts of the image. The metric g is set to high values in these areas, and remains low elsewhere (namely equal to 1, to get back the classical Euclidean behavior of the model). An example of segmentation using this method is shown in Fig. 3.

The second method is to automatically determine the areas of interest. In the deformable contours framework, these places are typically characterized by high values of the gradient norm. Therefore, a good choice for the function g is $g(x, y) = 1 + ||\nabla I|| * G_\sigma$, where G_σ denotes the usual Gaussian smoothing filter. This ensures that the model resolution increases nearby strong image contours, while remaining coarse elsewhere.

Anisotropic Metrics. The metrics previously described are simple and intuitive. However, they do not take advantage of all the properties of Riemannian metrics. With anisotropic metrics, space deformations depend on the direction.

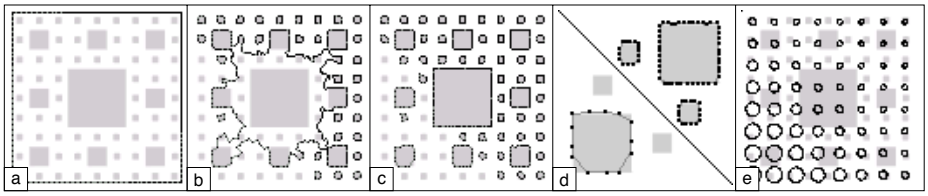


Fig. 3. Results with a user-defined metric. The images (a), (b) and (c) show different evolution steps of the segmentation algorithm. The image (d) shows details of the upper left and lower right parts of the image (c), and emphasizes the resolution difference between the two parts of the image. The metric is isotropic and has high values in the upper right corner of the image. In the lower left corner, it is equivalent to the Euclidean metric. This is shown in the picture (e), where the circles represent the local unit ball of the metric. Note that this experience also validates the automated topological changes of the model in a space with a deformed metric.

The following paragraph shows how that kind of metric improves the segmentation.

Consider the metric g defined as the Euclidean metric in the image places where the gradient norm is low or nul. Suppose that elsewhere, g is the metric with the spectral decomposition $(\boldsymbol{v_1}, \mu_1)$, $(\boldsymbol{v_2}, \mu_2)$, with $\boldsymbol{v_1} = \boldsymbol{\nabla I}$, $\mu_1 = ||\boldsymbol{\nabla I}||$, $\boldsymbol{v_2} = \boldsymbol{v_1}^{\perp}$ and $\mu_2 = 1$.

In places where there are no strong contours, the deformable model behaves as if the metric were Euclidean. Therefore, no new vertices are created, and the model resolution remains coarse. In the neighborhood of a contour, the deformable model mesh may either cross over or follow the contour. Suppose that it crosses over the contour. In a small neighborhood the mesh is roughly parallel to $\boldsymbol{v_1}$. With our choice for μ_1 edge lengths are overestimated. Therefore the model locally increases its resolution, which provides it with more degrees of freedom. In contrast, suppose that the model mesh follows the contour. Then the mesh is roughly parallel to $\boldsymbol{v_2}$ (*i.e.* orthogonal to $\boldsymbol{\nabla I}$), and with our choice for μ_2, distances are estimated as in the Euclidean case. Consequently, the model resolution remains coarse, and the number of vertices used to describe the image component remains low.

The different methods considered before are experimented and compared on computer generated images in Fig. 4 and 5. Additional results with a MR brain image are shown on Fig. 6.

6 Conclusion

We have proposed a bi-dimensional highly deformable model that dynamically and locally adapt its resolution. Hence, a fine resolution is achieved in the interesting parts of the image while a coarse resolution is kept elsewhere. By this way, the segmentation algorithm complexity is made significantly more independent from the size ot input data. In a bi-dimensional image, dividing twice the

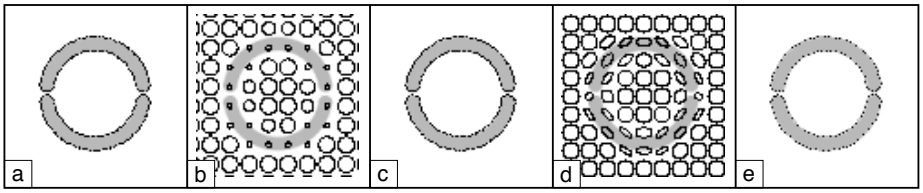
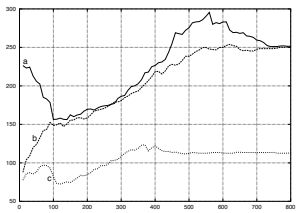


Fig. 4. Segmentation with different automatically computed metrics: the model is initialized around the figure. The image (a) shows the result obtained with the Euclidean metric with the lowest model resolution that allows to distinguish the two connected components of the image. The image (c) shows the result obtained with the isotropic Riemannian metric described in image (b). The picture (e) shows the segmentation obtained with the anisotropic Riemannian metric described in Fig. (d).



Metric	Time (s)	Complexity
(a) Euclidean	3.88	186,567
(b) Isotropic	49.72	152,860
(c) Anisotropic	8.33	55,258

Fig. 5. Statistics: the graph represents the evolution of the number of vertices of the model with respect to the iteration number, and for different metrics. The segmented image is the same as in Fig. 4. This emphasizes that our new model is able to extract image components with significantly less vertices than the initial model, especially if anisotropic metrics are used (curve c on the graph). It also shows that the use of deformed metrics speeds up the model convergence. Computation times remain higher than with an Euclidean metric, however, many optimizations are likely to be performed to reduce the computational cost per vertex.

model resolution divides the number of vertices by two. For three-dimensional image, the same loss of resolution divides four times the number of vertices. Consequently, highly deformable models with adaptative resolution would be even more useful for three-dimensional image segmentation. The model we have extended was initially designed for this purpose. Moreover, all the tools that are used to define our new deformable model remain available in three-dimensional spaces. We are thus currently working on a three-dimensional version of this model.

References

1. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *ICCV95*, pages 694–699, 1995.
2. L. D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, March 1991.

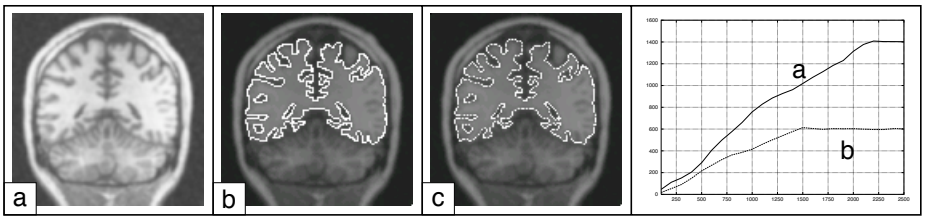


Fig. 6. Results with a biomedical image. These images show the segmentation of image (a) using an Euclidean metric (b) and an automatically computed anisotropic Riemannian metric (c). The segmentation quality is rather equivalent with the two metrics, whereas the number of vertices used to describe the shape is far smaller with the anisotropic metric (only 596 vertices for image (c), instead of 1410 for image (b)). Figure (c) shows the evolution of the number of vertices with the iteration number. This graph illustrates that using deformed metrics speeds up the model convergence.

3. H. Delingette. General object reconstruction based on simplex meshes. Research Report 3111, INRIA, Sophia Antipolis, France, February 1997.
4. H. Delingette and J. Montagnat. New algorithms for controlling active contours shape and topology. In D. Vernon, editor, *European Conference on Computer Vision (ECCV'2000)*, number 1843 in LNCS, pages 381–395, Dublin, Ireland, June 2000. Springer.
5. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
6. J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
7. J.-O. Lachaud and A. Vialard. Discrete deformable boundaries for the segmentation of multidimensional images. In C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, editors, *Proc. 4th Int. Workshop on Visual Form (IWVF4), Capri, Italy*, volume 2059 of *Lecture Notes in Computer Science*, pages 542–551. Springer-Verlag, Berlin, 2001.
8. R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape Modelling with Front Propagation: A Level Set Approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):158–174, February 1995.
9. T. McInerney and D. Terzopoulos. Medical Image Segmentation Using Topologically Adaptable Snakes. In *Proc. of Computer Vision, Virtual Reality and Robotics in Medicine*, pages 92–101, Nice, France, 1995. Springer-Verlag.
10. S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulation. *Journal of Computational Physics*, 79:12–49, 1988.
11. J.A. Sethian. *Level Set Methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 1996.
12. J. Strain. Tree methods for moving interfaces. *Journal of Computational Physics*, 15(2):616–648, May 1999.
13. C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 7(3):359–369, March 1998.