Image Segmentation with Cage Active Contours

Lluís Garrido, Marité Guerrieri, Laura Igual

***Abstract*—In this paper, we present a framework for image segmentation based on parametrized active contours. The evolving contour is parametrized according to a reduced set of control points that form a closed polygon and which have a clear visual interpretation. The parametrization, called mean value coordinates, stems from the techniques used in computer graphics to animate virtual models. Our framework allows to easily formulate region-based energies to segment an image. In particular, we present three different local region-based energy terms: the mean model, the Gaussian model and the histogram model. We show the behavior of our method on synthetic and real images and compare the performance with state of the art level set methods.**

***Index Terms*—Parametrized active contours, Level sets, Mean value coordinates**

I. Introduction

Since their invention in [1], active contours have proven to be a powerful tool for segmentation in image processing. In active contours an evolving interface is propagated in order to recover the shape of the object of interest. The evolution of the interface is driven by minimizing an energy defined by a variational formulation which mathematically expresses the properties of the object to be segmented. Classically, the terms associated to image features are either edge-based, such as the image gradient on the contour as in [2], or region-based terms, as introduced by Chan and Vese in [3]. Region-based terms are known to be more robust to noise than edge-based contours and they do not need the initialization to be near the solution. The work of Chan and Vese is based on evolving the interface according to the variance of the gray-level values of both interior and exterior regions. This approach has been extended, since then, to other features such as the Bayesian model [4] and histogram model [5]. The latter approaches do define the inner and outer regions as the whole inner and outer space, respectively, of the evolving contour. Thus, they may fail if these features are not spatially invariant. This problem is tackled in [6] that proposes to compute the features in a band around the evolving contour. In [7] the inner and outer regions are defined at each point of the evolving region as the intersection between a ball and the inside and outside space. In [8] authors use a kernel function at each point to define a region-scalable fitting term. Finally, two fast algorithms are presented in [9]) and [10], where a B-Spline parametrization and a discrete approximation-based representation are presented, respectively.

L. Garrido and L. Igual are within the Department of Applied Mathematics and Analysis of the Univerity of Barcelona and the Computer Vision Center, Spain. E-mail: {lluis.garrido, ligual}@ub.edu.

M. Guerrieri is within the Geometry and Graphics Group of the University of Girona, Spain. E-mail: mariteg@ima.udg.edu

According to the representation of the evolving interface, active contours may be classified into parametric and geometric approaches. Geometric approaches represent the evolving interface as the zero level set of a higher dimensional function, which is usually called level set function. Level set approaches are able to cope with the curve topology and thus can segment multiple unconnected regions. This property has made level sets to constitute a very popular approach. In parametric approaches the curve may be described by means of a set of discrete points [1] or using basis functions such as B-splines [11], [12]. Using basis functions requires less parameters than direct discretization and have inherent regularity.

Parametric contours are able to deal easily with edge-based energies. However, dealing with region-based energies is more difficult, see [11] for instance. In addition, in parametric contour approaches one needs to define the number of parameters or control points that will be used to evolve the contour. Level set approaches are usually computationally more complex and difficult to deal with since, in a two dimensional problem, they evolve a surface rather than a curve. This makes level sets a difficult approach in 3D applications and thus some authors are currently using a parametric formulation of the curve for these type of problems [13].

In this paper, we contribute with a framework for segmenting connected objects using a new class of parametric active contours. In particular, the evolving interface is driven by a parametrization based on a class of deformable models well known in computer graphics. In this area, an important problem is the animation of models for video games or movies. Such models are usually made up of millions of triangles, whereas the motion of the character is controlled by a reduced number of control points. When these control points are moved the associated mesh deforms accordingly. A similar idea is applied in our approach: the evolving interface is represented by a set of points. These points are then parametrized by a set of reduced control points that evolve according to an energy to be minimized. When these control points move the evolving interface evolves correspondingly to the object to be segmented.

Our work stems from the ideas of free-form deformation technique [14], [15]. Free-form deformation has been actively used for medical image registration [16]. However, to the best of our knowledge, free-form deformation has not been used for parametric active contours. In our work, we use the *mean value coordinates* as the parametrization to deform the evolving contour [17]. Mean value coordinates have several advantages over free-form deformation, namely that control points only need to form a closed polygon that may have any shape. Any point of the space be inside or outside of this polygon may be parametrized with respect to the control points. For free-form deformation the control points need to form a regular shape

and only interior points may be parametrized with them.

In our approach, we can easily deal with region-based approaches. This is an advantage with respect to most previous parametrized approaches, which are only able to deal with edge-based energies. According to our knowledge, there is almost no work in the field of parametric-based approaches, except for [13] in 3D, able to deal in a unified manner with the mean, Bayesian and histogram model, which is the case in this paper. As with other similar parametric-based approaches, in our approach the evolving contour is locally deformed as the polygon vertices are moved. The deformation applied to the evolving contour is implicitly regularized, as will be seen, according to the distance of the polygon to the evolving interface. The versatility of our approach opens the door to new applications such as medical image segmentation, where most of the times the structure to be segmented has only one regular connected component.

A preliminary version of this work has already been presented in [18]. The latter paper focuses only on the mean model for the energy term, see section III, for both 2D and 3D problems, and uses a non-uniform sampling, see section IV-A. We focus here only on 2D models, extend the latter work to Gaussian and histogram based models, see section III, and improve the method by using a uniform sampling and a more robust gradient descent method, see section IV-B.

The rest of the paper is organized as follows: section II reviews the related state-of-the-art work, section III introduces the proposed segmentation method, section IV details the implementation, section V shows and explains the experimental results, and section VI concludes the paper.

II. Related Work

A. Level Sets

Let $\mathcal{C}(p) : [0,1] \to \mathbb{R}^2$ be an Euclidean parametrization of the curve and let $I : \mathbb{R}^2 \to \mathbb{R}$ be a gray level image. Level set methods are based on embedding the curve $\mathcal{C}$ in a higher dimensional function ϕ which is defined over all the image. Instead of evolving the curve $\mathcal{C}$, the function ϕ is evolved.

The level set method was introduced in [19]. Pioneering works in this field are the *geometric active contour* introduced in [20] and [21], and the *geodesic active contour* model proposed in [2].

Chan and Vese presented a method to evolve a curve by minimizing the variance in the interior, Ω_1, and exterior region, Ω_2, defined by $\mathcal{C}$ [3]. The energy the authors minimize is

$$E(\mathcal{C}) = \frac{1}{2}\iint_{\Omega_1} (I-\mu_1)^2\,dx\,dy + \frac{1}{2}\iint_{\Omega_2} (I-\mu_2)^2\,dx\,dy, \quad (1)$$

where the terms based on the contour length and area enclosed by Ω_1 have been dropped for simplicity. In (1), the image I corresponds to the observed data. The μ_1 and μ_2 refer to mean intensity values in the interior and exterior region, respectively.

In [4], an approach that assumes a Gaussian model for Ω_1 and Ω_2 was presented. For that issue the pixel intensities inside and outside the contour $\mathcal{C}$ are assumed to follow a Gaussian probability distribution. The energy to be minimized (considering only the region based terms) is given by

$$E(\mathcal{C}) = \iint_{\Omega_1} e_1\,dx\,dy + \iint_{\Omega_2} e_2\,dx\,dy. \quad (2)$$

Here e_1 and e_2 correspond to the log-likelyhood function

$$e_h = \log\sigma_h + \frac{(I(p)-\mu_h)^2}{2\sigma_h^2} \quad (3)$$

where $h = \{1,2\}$, μ_h is the internal ($h = 1$) or external ($h = 2$) mean and σ_h is the internal or external variance.

In [5], a level set method to segment images using histograms is presented. In this case, the objective is to segment an image by maximizing the distance between the two probability distributions. The authors propose to use the *Bhattacharyya* distance to maximize the distance between probability distributions p_i. Particularly, the distance is $-\log B$ with B defined as

$$B = E(\mathcal{C}) = \int_R \sqrt{p_1(z)p_2(z)}dz, \quad (4)$$

and $z \in R$. The functions $p_1(z)$ and $p_2(z)$ correspond to the gray-level histogram of regions Ω_1 and Ω_2 and are computed by means of Parzen windowing. By minimizing the energy function B, we maximize the distance between the two probability distributions.

B. Mean Value Coordinates

An important problem in computer graphics is to define an appropriate function to linearly interpolate using data given at a set of vertices of a closed contour. Such interpolants can be used in applications such as parametrization, shading or deformation. The latter application is our main interest in this work and thus we will discuss it next.

Goraud [22] presented a method to linearly interpolate the color intensity at the interior of a triangle given the color at the triangle vertices. Indeed, if the triangle has vertices $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ and corresponding color values $\{f_1, f_2, f_3\}$, the color value of an interior point $\mathbf{p} = (x, y)$ of this triangle can be computed as $\hat{f}[\mathbf{p}] = \sum_j w_j f_j / \sum_j w_j$, where w_j is the area of the triangle given by vertices $\{\mathbf{p}, \mathbf{v}_{j-1}, \mathbf{v}_{j+1}\}$.

Many researchers have used these types of interpolants for mesh parametrization methods [23], [24], [17] as well as free-form deformation methods [25], [15] among others. Both applications require that a point $\mathbf{p}$ be represented as an *affine combination* of the vertices of an enclosing closed contour given by vertices $\mathbf{v}_i$. That is,

$$\mathbf{p} = \sum_{i=1}^{N} \varphi_i(\mathbf{p})\mathbf{v}_i \quad (5)$$

where $\varphi_i(\mathbf{p})$ is the corresponding *affine coordinate* of the point $\mathbf{p}$ with respect to the vertex $\mathbf{v}_i$ and N is the number of vertices.

A wealth of approaches have been presented for the computation $\varphi_i(\mathbf{p})$. We may mention those that have interest for deformation applications, namely mean value coordinates [17], harmonic coordinates [26] or Green coordinates [27]. Among them, we have used the mean value coordinates, since they are easy to compute and allow to parametrize any point of the space, be inside or outside the polygon.

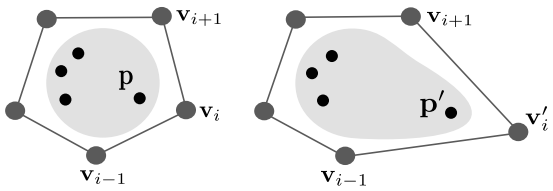

Fig. 1. Example of the deformation of a region by means of a polygon. From right to left: the polygon vertex $\mathbf{v}_i$ is moved producing the consequent deformation of the region after applying the interpolation function.

Mean value coordinates were initially proposed for mesh parametrization problems [17]. The author demonstrated that the interpolant generated smooth coordinates for star-shaped polygons for any point $\mathbf{p}$ inside the polygon. Later on, in [28], it was demostrated that mean value coordinates extended to any planar polygon and to any point $\mathbf{p}$ on the plane.

The mean value coordinates of a point $\mathbf{p}$, $\varphi_i(\mathbf{p})$, given a set of vertices $\mathbf{v}_i$ of a polygon of N points, $j = 1 \dots N$, are computed as

$$\varphi_i(\mathbf{p}) = \frac{w_i}{\sum_{j=1}^{N} w_j} \quad i = 1 \dots N, \tag{6}$$

and w_i is computed as

$$w_i = \frac{tan(\alpha_{i-1}/2) + tan(\alpha_i/2)}{||\mathbf{v}_i - \mathbf{p}||},$$

where $\|\mathbf{v}_i - \mathbf{p}\|$ is the distance between the vertex $\mathbf{v}_i$ and the considered point $\mathbf{p}$ and α_i is the *signed* angle of $[\mathbf{v}_i, \mathbf{p}, \mathbf{v}_{i+1}]$.

Given the affine coordinates $\varphi_i(\mathbf{p})$ of a point $\mathbf{p}$, the point $\mathbf{p}$ can be recovered with (5). If the vertices $\mathbf{v}_i$ of the polygon move to positions $\mathbf{v}'_j$, the "deformed" point $\mathbf{p}'$ can be recovered as

$$\mathbf{p}' = \sum_{i=1}^{N} \varphi_i(\mathbf{p}) \mathbf{v}'_i, \tag{7}$$

where note that the point $\mathbf{p}'$ is recovered from the affine coordinates $\varphi_i(\mathbf{p})$, see Fig. 1.

Given a set of points $\{\mathbf{p}\}$, the affine coordinates for each point are computed in an independent way using (6). If a point $\mathbf{v}_i$ of the polygon is stretched in a particular direction, all the points $\{\mathbf{p}\}$ follow the same direction with an associated weight given by $\varphi_i(\mathbf{p})$, see Fig. 1. The points $\mathbf{p}$ that are near the moved vertex have higher weight, see denominator of (6), and thus suffer a larger "deformation" than the points which are farther where the weight is smaller and hence, they are barely affected by the deformation.

Fig. 2 depicts the influence of the polygon distance to the curve. The polygon is shown with a solid line whereas the evolution curve with a dashed line. The first and third column shows the initial configuration. The second and fourth column shows how the curve is deformed when a control point of the polygon is moved. As can be seen, the movement of the polygon produces a smooth deformation of the curve. The closer the polygon to the curve the higher the deformation that is applied to the curve.

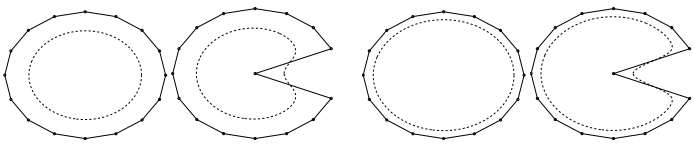

Fig. 2. The curve deformation for two polygon configurations. The polygon is shown as solid line and the evolving curve as dashed line.

III. Cage Active Contours

In this work we use the term *cage* to refer to the polygon that allows to deform the evolving contour. Therefore we call our method Cage Active Contours (CAC). The image segmentation problem is formulated as the minimization of a region-based energy. Edge-based energies (such as the length of the contour) can also be included within the energy formulation, but this is out of the scope of this work.

We propose three different region-based energy models based on the energies reviewed in section II-A. The proposed energy models are based on a discretization of the energy functionals rather than the corresponding evolution equations. Let us denote $\mathbf{v} = \{\mathbf{v}_1, \dots, \mathbf{v}_N\}$ the set of cage vertices (or control points) associated to our parametrization, and let Ω_1 and Ω_2 be the set of pixels of the interior and exterior, respectively, of the evolving interface $\mathcal{C}$. Ω_1 and Ω_2 are automatically computed given an initial mask (see section IV for a discussion on the way Ω_1 and Ω_2 are defined). The cage vertices $\mathbf{v}$ can be manually placed or automatically set from the evolving contour $\mathcal{C}$, as is done in this work.

1) Mean model: The first model assumes that the gray-level of pixels inside Ω_1 and Ω_2 can be modelled with a single value which corresponds to the mean. This method follows the formulation presented in (1).

The measure of the region energy can be defined as follows

$$E_{\text{mean}} = \sum_{h=1}^{2} \sum_{\mathbf{p} \in \Omega_h} \frac{1}{2} (I(\mathbf{p}) - \mu_h)^2$$

where the mean gray-level value of Ω_h is defined as

$$\mu_h = \frac{1}{|\Omega_h|} \sum_{\mathbf{p} \in \Omega_h} I(\mathbf{p}), \tag{8}$$

and $|\cdot|$ denotes the cardinal of the set.

Since the objective is the minimization of the previous energy, we need to compute its derivatives with respect to the cage vertices $\mathbf{v}$. Let $\mathbf{v}_j = (v_{j,x}, v_{j,y})^T$ be the coordinates of a cage vertex. The gradient of E_{mean} with respect to $\mathbf{v}_j$ is

$$\nabla_{\mathbf{v}_j} E_{\text{mean}} = \sum_{h=1}^{2} \sum_{\mathbf{p} \in \Omega_h} (I(\mathbf{p}) - \mu_h)(\varphi_j(\mathbf{p}) \nabla I(\mathbf{p}) - \nabla_{\mathbf{v}_j} \mu_h),$$

where

$$\nabla_{\mathbf{v}_j} \mu_h = \frac{1}{|\Omega_h|} \sum_{\mathbf{p} \in \Omega_h} \nabla I(\mathbf{p}) \varphi_j(\mathbf{p}). \tag{9}$$

The previous formulation assumes a dependency of μ_h with respect to the region Ω_h. In classical level set approaches, such as [3], authors do not take into account this dependency and use a coordinate descent approach to minimize the functional: the mean μ_h and the evolution equation are updated iteratively

to evolve the equation. In this work we also have tested this approach by assuming $\nabla_{\mathbf{v}_j}\mu_h = 0$. Our experiments have shown no appreciable difference between using a gradient that uses $\nabla_{\mathbf{v}_j}\mu_h$ as expressed in (9) or a gradient $\nabla_{\mathbf{v}_j}\mu_h = 0$. We therefore use the latter approach for the Gaussian model since it simplifies the expressions for the gradient.

2) Gaussian model: The second model assumes a Gaussian distribution of pixel gray-level values inside Ω_1 and Ω_2, similar as in [4], and follows the formulation in (2).

We assume that interior and exterior pixels follows a Gaussian distribution: we use the discretized probabilistic formulation presented in [4], [11] and take the logarithm of the Gaussian probabilistic function. The energy function to be minimized is

$$E_{\text{gauss}} = -\sum_{h=1}^{2}\sum_{\mathbf{p}\in\Omega_h} e_h \tag{10}$$

where e_h is given by (3). Here μ_h is the internal or external mean value as defined in (8) and σ_h^2 is the associated variance. The gradient is given by

$$\nabla_{\mathbf{v}_j} E_{\text{gauss}} = \sum_{h=1}^{2} \frac{1}{\sigma_h^2} \sum_{\mathbf{p}\in\Omega_h} (I(\mathbf{p}) - \mu_h)\varphi_j(\mathbf{p})\nabla I(\mathbf{p}).$$

To compute the latter expression we have assumed that $\nabla_{\mathbf{v}_j}\mu_h = 0$ and $\nabla_{\mathbf{v}_j}\sigma_h^2 = 0$, as commented in the previous subsection.

3) Histogram model: The third model follows the histogram formulation presented in section II-A. In particular, it uses the Bhattacharyya distance, defined as $-\log(B)$, where B is defined as in (4). Our objective is to maximize the distance between two probability distributions, so B should be minimized.

We assume that the image has K gray-level values: $I(x, y) \in [0, K-1]$. Therefore, the histogram has K bins.

The discretized energy can be expressed as

$$E_{\text{histogram}} = \sum_{k=0}^{K-1} \sqrt{p_1(k)p_2(k)},$$

where $p_1(k)$ and $p_2(k)$ are, respectively, the discretized gray-level histogram associated to Ω_1 and Ω_2. The histogram bin $p_h(k)$, $h \in \{1, 2\}$, is computed by means of the Parzen window [5]

$$p_h(k) = \frac{1}{|\Omega_h|} \sum_{\mathbf{p}\in\Omega_h} g_{\sigma_h}(k - I(\mathbf{p})),$$

where g_{σ_h} is the Gaussian function of zero mean and variance σ_h. The gradient is

$$\nabla_{\mathbf{v}_j} E_{\text{histogram}} = \sum_{k=0}^{K-1} \left(\frac{1}{2}\sqrt{\frac{p_2(k)}{p_1(k)}}\nabla_{\mathbf{v}_j} p_1(k) + \frac{1}{2}\sqrt{\frac{p_1(k)}{p_2(k)}}\nabla_{\mathbf{v}_j} p_2(k) \right),$$

$$\nabla_{\mathbf{v}_j} p_h(k) = -\frac{1}{|\Omega_h|} \sum_{\mathbf{p}\in\Omega_h} g'_{\sigma_h}(k - I(\mathbf{p}))\nabla I(\mathbf{p})\varphi_j(\mathbf{p}),$$

where $g'_{\sigma_h}(.)$ is the derivative of the Gaussian function. The value of σ_h is automatically computed as proposed in [5].

IV. Implementation

Assume gray-level input image I and a given input mask $M$are available. The input mask M is a binary mask that is used as initialization for the algorithm and its outer boundary is an approximation to the object boundary we want to segment. In the case of medical images, for instance, this input mask can be obtained by means of a registration of the image to be segmented with an image representing an atlas or mean case. In other cases, the input mask can be manually defined by the user or automatically placed using a predefined shape and position. Several choices are available to compute the inner Ω_1 and outer pixels Ω_2. For instance, Ω_1 may be composed of the pixels of M, $\Omega_1 = M$, and Ω_2 may be taken as the set $\Omega_2 = \Omega \setminus M$, where Ω is the whole image support. This is the way in which Ω_1 and Ω_2 are defined for the level sets reviewed in section II-A and also is the approach taken in this paper. We assume here that Ω_1 is a connected component. However, the method presented here is not restricted to this case. That is, Ω_1 may be composed of multiple connected components. Nevertheless, note that our method is not able to deal with topological changes of Ω_1. In addition, Ω_1 and Ω_2 may contain pixels that are too far away from the evolving contour and thus should not be taken into account for computations. We may only use a band of pixels around the evolving contour for minimization of the energy such as is done in [7]. This is however out of the scope of this paper and is left for future work.

Minimization of the energy is performed by means of a gradient descent process that iteratively updates the cage vertex positions. At each step of the gradient descent, Ω_1 and Ω_2 are recomputed. The details associated to this process of minimization are described in the following sections.

A. Pixel Sampling

An important aspect to define is the way the pixels are sampled during the evolution of the cage vertices. Two approaches are proposed in this paper, namely the non-uniform and the uniform sampling, showing different properties.

1) Non-uniform sampling: The first approach, described in [18], is based on computing Ω_1 and Ω_2 using the initial mask, before starting the gradient descent. The affine coordinates $\varphi_i(\mathbf{p})$ of the pixels $\mathbf{p}$ of Ω_1 and Ω_2 are then computed using the cage points $\mathbf{v}$, see (6). These coordinates only are computed once, before the optimization algorithm is initiated. The cage vertices are then evolved using the gradient descent. For a given $\mathbf{v}$, the corresponding "deformed" pixel positions $\mathbf{p}$ are recovered using (7). The energy $E(\mathbf{v})$ and gradient $\nabla E(\mathbf{v})$ is then computed using the recovered pixel positions $\mathbf{p}$ which may be non-integer positions. Hence, we apply bilinear interpolation to estimate $I(p)$ and $\nabla I(p)$ at such points. Our experience has shown that the method is computationally fast but the image is sampled in a non-uniform way leading to a non-robust method.

2) Uniform sampling: The second approach is based on extracting at the initialization stage the contour $\mathcal{C}$ associated to the mask M. The contour is represented as a list of neighboring pixels that form the contour and can be ordered

either in clockwise or counter-clockwise manner. Let $\mathbf{c}_i$ be the points of the discretized contour. The affine coordinates of $\mathbf{c}_i$ are computed before the gradient descent is started. For each iteration of the gradient descent approach, the "deformed" contour $\mathcal{C}'$ is recovered. The sets Ω_1 and Ω_2 can be easily extracted by describing them using integer pixel positions, and its associated affine coordinates are computed. Then $E(\mathbf{v})$ and $\nabla E(\mathbf{v})$ can be evaluated and the cage vertices are updated. This approach implies computing Ω_1, Ω_2 and its associated affine coordinates at each iteration of the gradient descent. Thus, it is a computationally intensive method. However, the computation of the affine coordinates can be easily parallelized using, e.g. OpenCL. The advantage of this approach is that, at each iteration, the image is sampled at integer pixel positions, and thus, in a uniform way.

For the exposed properties, all the experiments in Section V are done with the uniform sampling technique.

B. Gradient descent

For a given energy E a gradient descent is used to minimize it. The gradient method iteratively updates $\mathbf{v}^{k+1}$, $\mathbf{v} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, of iteration $k+1$ from $\mathbf{v}^k$ of iteration k using $\mathbf{v}^{k+1} = \mathbf{v}^k + \alpha^k \mathbf{s}^k$, where $\mathbf{s}^k$ is the so called search direction and α^k the step. Usually the steepest descent direction $\mathbf{s}^k = -\nabla_{\mathbf{v}^k} E$ is taken as the search direction. In this work we take the negative of the normalized gradient vector obtained as follows:

$$\mathbf{s}^k = -\left(\frac{\nabla_{\mathbf{v}_1^k} E}{||\nabla_{\mathbf{v}_j^k} E||_{\max}}, \frac{\nabla_{\mathbf{v}_2^k} E}{||\nabla_{\mathbf{v}_j^k} E||_{\max}}, \ldots, \frac{\nabla_{\mathbf{v}_N^k} E}{||\nabla_{\mathbf{v}_j^k} E||_{\max}} \right)$$

where

$$||\nabla_{\mathbf{v}_j^k} E||_{\max} = \max\left\{ ||\nabla_{\mathbf{v}_j^k} E|| \, : \, j = 1 \ldots N \right\}.$$

We assume that the distance between 4-based neighboring pixels is one. Normalization of the gradient vector implies that for $\alpha = 1$ vertices move at most one pixel from its current position. This allows to intuitively interpret the search direction and to improve the robustness of the method. The computation of α is a critical issue for the good performance of the algorithm. For that issue we use the line search algorithm described in [29]. This algorithm updates $\mathbf{v}^k$ by performing an inexact line search such that $\alpha^k \leq \alpha_{\max}$. The computation of $\alpha_{\max}$ is described in Section IV-C. The gradient descent algorithm stops if the line search algorithm selects an $\alpha^k < \beta$, that is, when the displacements of the vertices $\mathbf{v}_j^k$ is below β pixels. In this work a value of $\beta = 0.05$ has been used.

In this work, the gradient descent is performed by means of a two stage process.

1) First stage: The objective in this stage is to obtain a segmentation as close as possible to the solution by restricting the direction in which the cage vertices may evolve. For that we use balloon forces similar as the ones described in [30]. Let $\mathbf{p}_c$ represent the initial (at $k = 0$) center of the contour. At each iteration of the gradient descent each cage vertex $\mathbf{v}_j^k$ is only allowed to move along the line that passes through $\mathbf{p}_c$ and $\mathbf{v}_j^k$. That is, the gradient vector $\nabla_{\mathbf{v}_j^k} E$ is projected on the line joining $\mathbf{p}_c$ and $\mathbf{v}_j$ before normalizing it. Thus, for each point $\mathbf{v}_j^k$ the problem is reduced to a one-dimensional problem. The descent method is thus iteratively applied until the line search method converges, that is, until a value $\alpha^k < \beta$ is selected.

2) Second stage: The second stage is devoted to adjust the segmentation by evolving the cage vertices without restrictions on the directions. At each iteration k the search direction $\mathbf{s}^k$ is computed. The line search method then updates $\mathbf{s}^k$ to obtain $\mathbf{s}^{k+1}$. The gradient desent converges when a value $\alpha^k < \beta$ is selected.

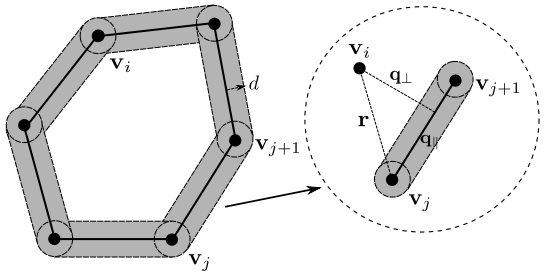

Fig. 3. A band of side d is setup around cage edges, to avoid that they cross with themselves during minimization. The value of $\alpha_{\max}$ is computed so as to ensure that vertices do not enter this band, see section IV-C.

C. Cage edge constraints

In this section we deal with the problem of avoiding that cage edges cross with each other during iterations. We impose this restriction in order to avoid that the corresponding contour points $\mathcal{C}$ cross with each other.

In the context of active contours, the problem of handling the crossings of the active contour $\mathcal{C}$ has already been tackled in the literature [31]. Here we propose a method that avoids that cage vertices do not get too close to the cage edges, thus avoiding edge crossings. Since the number of cage vertices is, in our experiments, very low compared to the number of points in the contour $\mathcal{C}$, the computational effort involved in our proposal is rather low.

The main idea is depicted in Fig. 3. In order to avoid edge crossings we have to avoid that vertex $\mathbf{v}_i$ enters into the gray band linking vertices $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$. The gray band is parametrized by d which can be interpreted as a distance in pixels measured from the edge between $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$. This constraint has been implemented by defining two cost functions. The first cost function allows to detect if a vertex $\mathbf{v}_i$ gets at a distance less than d to a vertex $\mathbf{v}_j$, with $j \neq i$, as follows

$$\Phi_{\text{vertex}}(\mathbf{v}_i) = \sum_{j \neq i}^{N} \Psi_{\text{vertex}}(\mathbf{v}_i, \mathbf{v}_j),$$

where $\Psi_{\text{vertex}}(\mathbf{v}_i, \mathbf{v}_j) = 0$ if $\|\mathbf{v}_i - \mathbf{v}_j\| > d$ and $\Psi_{\text{vertex}}(\mathbf{v}_i, \mathbf{v}_j) = (\|\mathbf{v}_i - \mathbf{v}_j\| - d)^2$ otherwise. Note that Φ_{vertex} is 0 if all vertex are at a distance greater than d between themselves, otherwise there will be some term $\Psi_{\text{vertex}}(\mathbf{v}_i, \mathbf{v}_j) \geq 0$.

The second cost function detects if a vertex $\mathbf{v}_i$ gets at a distance less than d to an edge linking two vertices $\mathbf{v}_j$ and

$\mathbf{v}_{j+1}$, where $j \neq i$ and $j+1 \neq i$. As shown in Fig. 3, $\mathbf{q}_{\parallel} = \mathbf{v}_{j+1} - \mathbf{v}_j$, $\mathbf{q}_{\perp}$ is the orthogonal vector to $\mathbf{q}_{\parallel}$, and the vector $\mathbf{r}$ is $\mathbf{r} = \mathbf{v}_i - \mathbf{v}_j$. The distance between $\mathbf{v}_i$ and the line joining $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ is given by $d_{\perp}(\mathbf{v}_i, \mathbf{v}_j) = |\mathbf{q}_{\perp} \cdot \mathbf{r}|/|\mathbf{q}_{\perp}|$, where $(.) \cdot (.)$ represents the dot product between two vectors. The projection of vector $\mathbf{r}$ on the line joining $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ is given by $r_{\parallel}(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{q}_{\parallel} \cdot \mathbf{r}/|\mathbf{q}_{\parallel}|$. Note that if $0 \leq r_{\parallel} \leq 1$, the projection of $\mathbf{v}_i$ on the line linking $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ is on the edge joining the two previous nodes. Thus, we define the second cost function as follows.

$$\Phi_{\mathbf{edge}}(\mathbf{v}_i) = \sum_{j \neq i, i-1} \Psi_{\text{edge}}(\mathbf{v}_i, \mathbf{v}_j),$$

where $\Psi_{\text{edge}}(\mathbf{v}_i, \mathbf{v}_j) = 0$ if $r_{\parallel}(\mathbf{v}_i, \mathbf{v}_j) < 0$ or $r_{\parallel}(\mathbf{v}_i, \mathbf{v}_j) > 1$ or $d_{\perp}(\mathbf{v}_i, \mathbf{v}_j) > d$. Otherwise, $\Psi_{\text{edge}}(\mathbf{v}_i, \mathbf{v}_j) = (d_{\perp} - d)^2$.

The two previous cost functions are combined as follows

$$\Phi_{\text{constraint}}(\mathbf{v}) = \sum_i \left(\Phi_{\text{vertex}}(\mathbf{v}_i) + \Phi_{\text{edge}}(\mathbf{v}_i)\right). \quad (11)$$

The value of $\alpha_{\max}$, introduced in Section IV-B, is computed for each iteration k using (11). Let $\mathbf{v}^k$ be the current vertices positions and $\mathbf{s}^k$ the normalized search direction. We then compute

$$\min_j \left\{ j \in \{1 \dots W\} \,:\, \Phi_{\text{constraint}}(\mathbf{v}^k + j\,\mathbf{s}^k) > 0 \right\}. \quad (12)$$

Given vertices $\mathbf{v}^k$, we search along direction $\mathbf{s}^k$ in discrete steps (with precision of one pixel) for the minimum value of j such the new vertices positions approach too much other vertices or edges. The previous restriction can be computed indeed in a fast manner. Once computed, we set $\alpha_{\max} = j-1$, the maximum step the line search algorithm can use. If, for a given k the result of (12) is null, we set $\alpha_{\max} = W$. In the experiments we set $W = 30$ and $d = 5$. That is, the vertices may move at most 30 pixels from one iteration to the next and the band is set to a width of 5 pixels.

V. Experimental Results

In this section, we present the experimental results for the three models of CAC method on synthetic and real images. We compare the performance with six different level set based methods implemented in the software Creaseg [32]. The interested reader can evaluate the CAC algorithm, together with the algorithms in Creaseg, for new images, using the code available in the authors website[1].

In all the experiments shown below, we use uniform sampling and we constrain edges to not cross between each other as described in section IV. Cage vertices are automatically placed in an uniform way around the initial evolving contour.

A. Synthetic Images

In this section, we perform an experiment to show the influence of the number of cage vertices and the distance between the cage and the evolving interface. We consider two synthetic images consisting of a uniform square and star in a uniform background, see Fig. 4. We give as input the set Ω_1, whose associated contour (i.e. the evolving interface) is drawn white, whereas the cage is drawn in black. In the left column of Fig. 4, the initial configuration of the cage and the evolving interface is shown. For the square (resp. star) image, from top to bottom, cages with 4 and 8 (resp. 8, 16 and 16) vertices are shown. Recall that the set Ω_1 is made up of all interior pixels, whereas Ω_2 was made up of all external pixels. In the right column, the segmentation results, using the mean model, are shown. As can be seen, the number of cage vertices influences the type of deformation that can be applied to the evolving interface. Indeed, the cage with only 4 (resp. 8) vertices is not able to correctly fit the square (resp. star) boundary, whereas the cage with 8 (resp. 16) vertices is able to properly deform the evolving interface to get closer to the real boundary of the

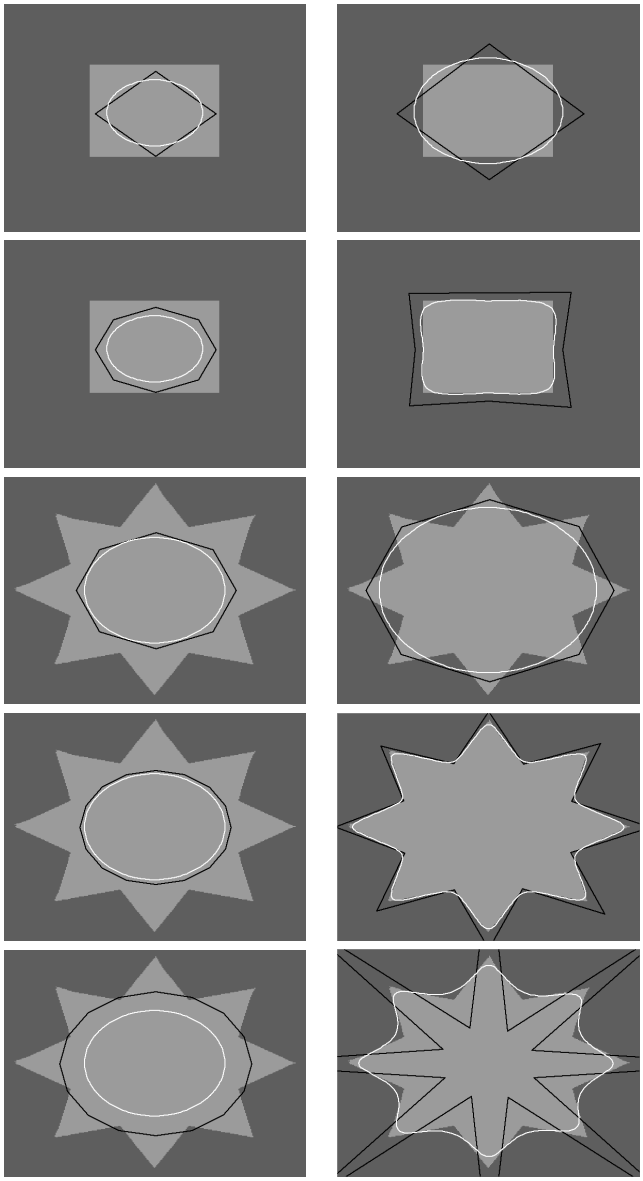

Fig. 4. Results on two synthetic images for the mean model. The left column shows the initialization, whereas the right shows the segmentation result. From top to bottom, the number of cage vertices is 4, 8, 8, 16 and 16. The distance of the cage vertices to the evolving interface is, respectively, 10, 10, 5, 5 and 25 pixels.

[1] http://www.maia.ub.es/~lgarrido/ieeeipcac2015

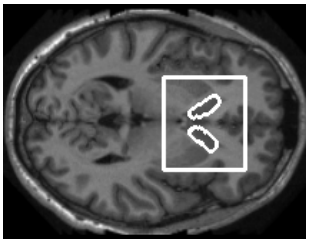

Fig. 5. MRI image with the Caudate Nuclei marked in white. The atlas-based crop is also marked in white.

square (resp. star).

We show the influence of the distance between the cage and the evolving interface in last two rows of Fig. 4. In both cases, the radius of the evolving interface at the initial stage is set to 70 pixels, but the distance of the cage is 5 and 25 pixels, respectively. As illustrated previously in Fig. 2, we can observe that the distance of the cage vertices to the evolving interface plays an important role as a regularization. The nearer the cage to the evolving interface is the greater the interface can be deformed. And, the greater the distance is the smaller the deformation that can be applied to the curve. For the last row, the algorithm stopped at the iteration at which the vertices were about to cross with each other.

B. Real Images

In these experiments, we apply CAC method to real images. We present comparison experiments, preliminary results of a multiresolution scheme which is part of our future work, and experiments to illustrate the energy evolution behavior.

We would like to point out that our objective in this work is not to improve current level set implementations. Rather, we would like to show that our parametric active contour approach provides a framework which allows to introduce energies that have been usually only tackled with level set approaches.

In the first experiment, we use Creaseg software [32] to quantitatively compare the CAC method with the mean model and six different level set methods. In particular, as is described in [32], the methods are classical contour-based approaches (Caselles [2]) and region-based approaches (Chan and Vese [3]), as well as more recent methods such as localized (Lankton [7] and Li [8]), parameterized (Bernard [9]), and discrete approximation-based techniques (Shi [10]).

We consider five images, four real images and a medical image. Real images are collected from Grabcut Dataset[2]: Castanet, Elephant, Trees and Grave images, see Fig. 7. The medical image is a Magnetic Resonance Image (MRI), see Fig. 5, which corresponds to one of the slices of a 3D MRI brain image from the MICCAI 2012 challenge[3]. The objective in this challenge was to segment different subcortical structures, as the Caudate Nuclei, marked in white in Fig. 5. Atlas information is commonly used to localize these structures in MRI. We use this information to define a crop of the image containing the target structures, marked also in white in the figure.

[2]http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm

[3]https://masi.vuse.vanderbilt.edu/workshop2012

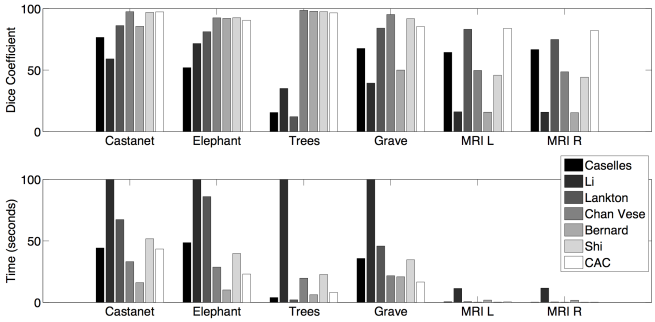

Fig. 6. Comparative results with 6 images (Castanet, Elephant, Trees, Grave, MRI L, MRI R). Top shows the Dice coefficient values and bottom shows the execution time in seconds.

The obtained segmentations are compared with the hand-labeled segmentations using the Dice coefficient [10]. This overlapping measure is defined as

$$DC(\Omega_A, \Omega_H) = 100 \cdot \frac{2 \cdot Area(\Omega_A \cap \Omega_H)}{Area(\Omega_A) + Area(\Omega_H)}$$

where Ω_A and Ω_H denote the automatic and hand-labeled segmentations. The DC varies from 0 to 100, where 100 means perfect agreement between segmentations and 0 means segmentations are completely different.

Fig. 6 summarizes the results of the seven methods on these five images. Note that, each hemisphere's Caudate structure is segmented separately, thus, we show the left and right result indicated as MRI L and MRI R. Top plot shows the Dice coefficient results and bottom one shows the execution time, in seconds. We consider the same initial evolving contour to initialize all the methods for a particular image (see first row of Fig. 7). In MRI segmentation, atlas-based strategies can be used to find contours near to the solution. Thus, to simulate this situation here, we build the initial curve by means of a morphological dilation of circular structuring element of radius 2 of the hand-labeled segmentation. For the six level set algorithms, we use the parameters setting by default defined in Creaseg software, except for the maximum number of iterations. This parameter is originally set to 200, and we increase it to 1000 to allow the convergence of all the methods. In particular, Chan and Vese needs more than 200 iterations to converge, but less than 1000 for all tested images. However, Li method does not converge before this limit and reaches the maximum number of iterations for all the real images. For visualization purposes, in the bottom plot of Fig. 6 we set the abscise limit to 100 seconds, even if Li method exceeds this limit. The cropped MRI image is much smaller than the others, thus, their segmentation requires much less time. We set the CAC parameters as follows, for the four real images: the distance of the cage points to the initial contour is 10 pixels and the number of control points is 16. For MRI image, we set the CAC parameters as follows: the distance of the control points to the initial contour is 20 pixels and the number of control points is 4. In addition, $W = 3$ since the image is small. As can be seen, CAC method is among the four best

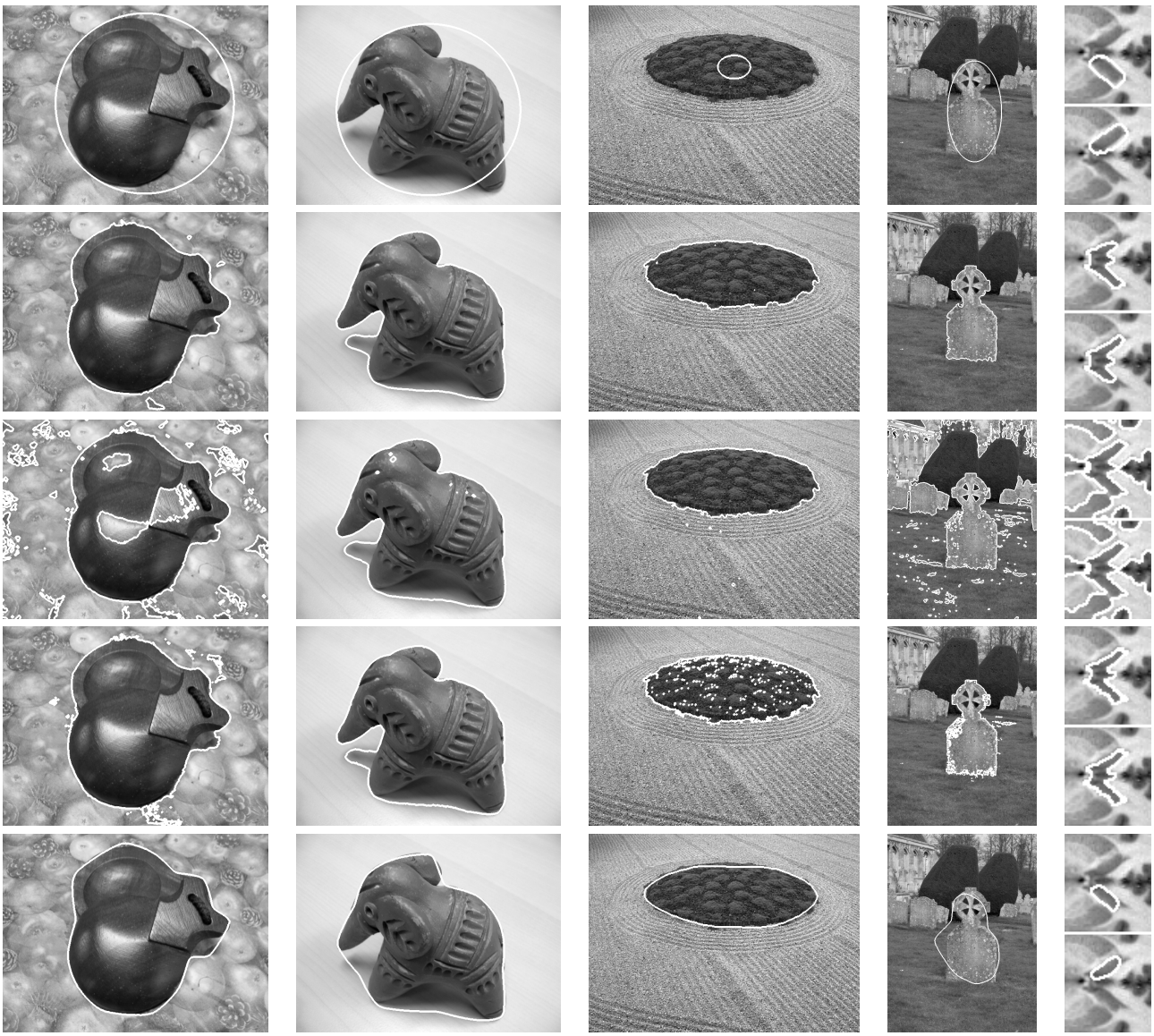

Fig. 7. Comparative results. From top to bottom: initialization curve, results of Chan and Vese, Bernard, Shi and CAC methods on Castanet, Elephant, Trees, Grave and the crop of MRI image.

methods in all cases and is the best for the MRI example.

Fig. 7 qualitatively shows the segmentation results obtained by Chan and Vese, Bernard, Shi and CAC methods, from second to fifth row respectively, on Castanet, Elephant, Trees, Grave and the MRI image. The first row show the initialization curve used for each image. As can be seen, for the real images, the obtained segmentation results are similar among the different methods. CAC result does not present spurious regions, but it does not properly define the more sharpen shapes, as it is the case of the Elephant legs. This is due to the regularization imposed by means of the cage-based parametrizaion. This property can be useful in medical image segmentation, as it is the case of the MRI example, where the structure to be segmented is a convex connected component. Note that the CAC method obtains the best segmentation result for the Caudate Nuclei, compared with the hand-labeled segmentation (Fig. 5).

In the next experiment, Fig. 8, we show an example using the Gaussian model and different number of cage vertices. In the top-left corner of the figure, the initialization for the cage of 8 vertices is shown. In the top-right corner, the result for the segmentation is shown. As can be seen, the evolving contour is not able to properly adapt to the contour of the squirrel due to the low number of cage vertices. In the bottom row, the result for the segmentation with 16 and 32 vertices is shown. The initialization is the same as the one with 8, but with a cage of 16 or 32 vertices. Increasing the number of vertices allows the evolving interface to better adapt to the contour of the squirrel. This idea may lead to multiresolution techniques. The multiresolution is performed in the cage vertices space rather than in the pixel space.

Fig. 9 presents an example to show the interest of this technique: we start with the result obtained with a cage made up of 16 vertices (bottom-left image of Fig. 8). By using this segmentation we construct a new cage around the interface. The initial cage now has 64 cage vertices, and can

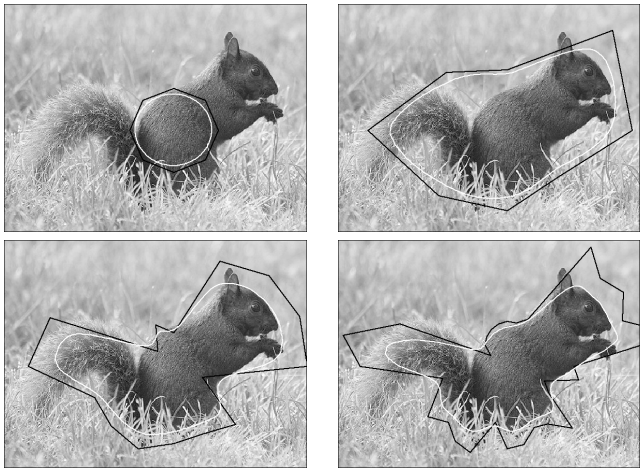

Fig. 8. Experiment with the Squirrel image using the Gaussian model and different number of cage vertices. Top shows initialization (left) and result (right) using 8 cage vertices. Bottom shows result with 16 (left) and 32 (right) cage vertices. Public domain image taken from http://en.wikipedia.org/wiki/File:Blacksquirrelrev.jpg.

Fig. 9. Experiment with the Squirrel image using the Gaussian model and a multiresolution technique. The algorithm starts with a cage made up of 16 vertices (see Fig. 8) and then uses the resulting segmentation to continue with a larger cage. The left shows the initial second-stage cage made up of 64 vertices. The right shows the resulting image.

be seen in Fig. 9 left. The interface is then evolved (using only the second step of the gradient descent) to result in the segmentation shown at the right which is more accurate than before. The multiresolution technique shown here is just a preliminary result. Part of our future work will concentrate on the usefulness of such technique.

Fig. 10 shows the result for the Castanet image using different initializations. All cages have 16 vertices. On the left the initialization is shown and on the right the result is shown. Observe that different initializations lead to similar results. As can be seen, our proposed method is robust with respect the cage initialization.

Our last experiment shows the result for the histogram model, see Fig. 11. For comparison purposes, we refer the reader to [5]. As can be seen, the algorithm has been able to properly adapt to the shape of the object using only the histogram energy model, that is, without using any additional energy term such as the length of the evolving contour as is the case for the level sets.

VI. Conclusion

In this paper, we have presented a parametrized active contour approach for two dimensional segmentation problems, the Cage Active Contour. The evolving interface is evolved by moving a set of cage points. Mean value coordinates are used to parametrize the points of the space with respect to the cage points. Our framework is suitable for the implementation of discrete energies in a unified framework, both region-based and edge-based terms, although we have shown here only the application to region-based energies. Regularization of the curve is obtained implicitly with the distance of the cage points to the curve. The versatility of our approach opens the door to new applications, such as medical ones, where target objects has only one smooth connected component.

Our future work will focus on the extension of the proposed approach to multicomponent images, on localized active contours, the improvement of the computational cost and the

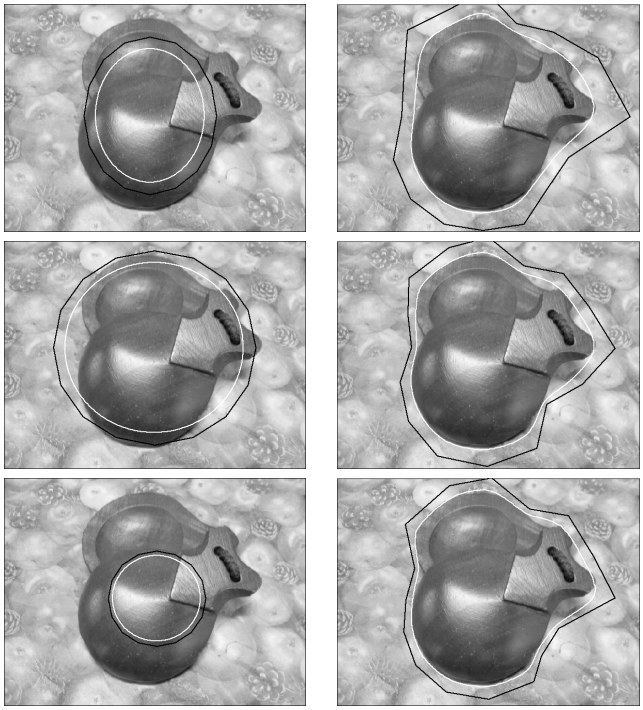

Fig. 10. Experiment with the Castanet image showing the result for different initializations using the Gaussian model. All cages have 16 vertices.

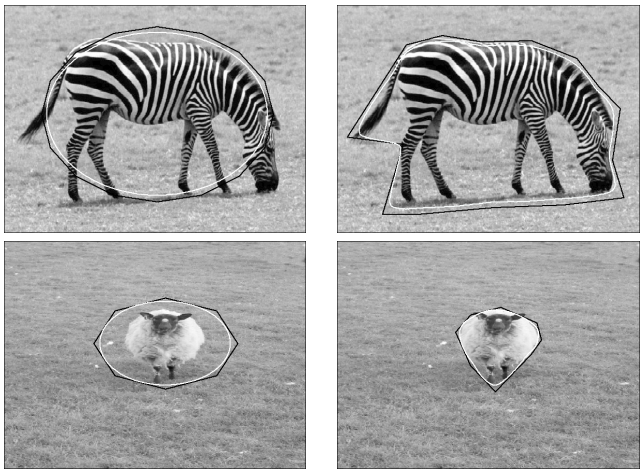

Fig. 11. Experiment with the zebra (top) and sheep (bottom) (from Microsoft Grabcut Dataset) using the histogram model and with a cage made up of 16 and 8 vertices, respectively. Left, original image, right, resulting segmentation.

extension to three-dimensional problems. Other issues, such as the introduction of new energy models, will be also tackled.

ACKNOWLEDGMENT

Authors would like to acknowledge MICINN projects, reference MTM2012-30772, TIN2012-38187-C03-01 and TIN2013-43478-P, and Narcís Coll of the Universitat de Girona for his helpful comments.

REFERENCES

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1988.
[2] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, pp. 61–79, 1997.
[3] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, feb 2001.
[4] M. Rousson and R. Deriche, "A variational framework for active and adaptative segmentation of vector valued images," in *IEEE Proceedings of the Workshop on Motion and Video Computing*, 2002, pp. 56–61.
[5] O. Michailovich, Y. Rathi, and A. Tannenbaum, "Image segmentation using active contours driven by the Bhattacharyya gradient flow," *IEEE Trans. on Image Processing*, vol. 16, no. 11, pp. 2787–2801, Nov. 2007.
[6] J. Mille and L. Cohen, "A local normal-based region term for active contours," in *Int. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, Aug. 2009, pp. 168–181.
[7] S. Lankton and A. Tannenbaum, "Localizing region-based active contours," *IEEE Transactions on Image Processing*, vol. 17, no. 11, pp. 2029–2039, Nov. 2008.
[8] C. Li, C. Kao, J. C. Gore, and Z. Ding, "Minimization of region-scalable fitting energy for image segmentation," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1940–1949, October 2008.
[9] O. Bernard, D. Friboulet, P. Thévenaz, and M. Unser, "Variational B-spline level-set: A linear filtering approach for fast deformable model evolution," *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1179–1191, June 2009.
[10] Y. Shi and W. C. Karl, "A real-time algorithm for the approximation of level-set-based curve evolution," *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 645–656, May 2008. [Online]. Available: http://dx.doi.org/10.1109/TIP.2008.920737
[11] M. Jacob, T. Blu, and M. Unser, "Efficient energies and algorithms for parametric snakes," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1231–1244, Sep. 2004.
[12] F. Precioso, M. Barlaud, T. Blu, and M. Unser, "Robust real-time segmentation of images and videos using a smoothing-spline snake-based algorithm," *IEEE Transaction on Image Processing*, vol. 14, no. 7, pp. 910–924, Jul. 2005.
[13] D. Barbosa, T. Dietenbeck, J. Schaerer, J. D'hooge, D. Friboulet, and O. Bernard, "B-spline explicit active surfaces: an efficient framework for real-time 3D region-based segmentation," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 241–251, Jan. 2012.
[14] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Dynamic free-form deformations for animation synthesis," *IEEE Trans. on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 201–214, Jul. 1997.
[15] S. Coquillart, "Extended free-form deformation: a sculpturing tool for 3d geometric modeling," *SIGGRAPH*, vol. 24, no. 4, pp. 187–196, Sep. 1990.
[16] D. Rueckert, L. Sonoda, C. Hayes, D. Hill, M. Leach, and D. Hawkes, "Nonrigid registration using free-form deformations: application to breast MR images," *IEEE Transactions on Medical Imaging*, vol. 18, no. 8, pp. 712–721, Aug. 1999.
[17] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19–27, Mar. 2003.
[18] Q. Xue, L. Igual, A. Berenguel, M. Guerrieri, and L. Garrido, "Active contour segmentation with affine coordinate-based parametrization," in *Int. Conference on Computer Vision Theory and Applications*, 2014.
[19] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12 – 49, 1988.
[20] V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours," *Numerische Mathematik*, pp. 694–6999, 1993.
[21] R. Malladi, J. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 158–175, 1995.
[22] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 623 – 629, June 1971.
[23] K. Hormann and G. Greiner, "Continuous shading of curved surfaces," *Curves and Surfaces Proceedings*, pp. 152 – 163, 2000.
[24] J. Schreiner, A. Asirvatham, M. Praun, and H. Hoppe, "Inter-surface mapping," *SIGGRAPH*, vol. 23, pp. 870–877, 2004.
[25] K. Kobayashi and K. Ootsubo, "t-ffd: free-form deformation by using triangular mesh," in *ACM Symp. on Solid Modeling and Applications*, 2003, pp. 226–234.
[26] P. Joschi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation," in *SIGGRAPH*, 2007.
[27] Y. Lipman, D. Levin, and D. Cohen-Or, "Green coordinates," in *SIGGRAPH*, 2008.
[28] K. Hormann and M. Floater, "Mean value coordinates for arbitrary planar polygons," *ACM Transactions on Graphics*, vol. 25, no. 4, pp. 1424–1441, Oct. 2006.
[29] J. J. Moré and D. J. Thuente, "Line search algorithms with guaranteed sufficient decrease," *ACM Transactions on Mathematical Software*, vol. 20, no. 3, pp. 286–307, Sep. 1994.
[30] L. Cohen, "On active contour models and balloons," *Image Understanding*, vol. 53, pp. 211–218, 1991.
[31] H. Delingette and J. Montagnat, "Topology and shape constraints in parametric active contours," Institut National de Recherche en Informatique et en Automatique, Tech. Rep., 2000.
[32] T. Dietenbeck, M. Alessandrini, D. Friboulet, and O. Bernard, "Creaseg: A free software for the evaluation of image segmentation algorithms based on level-set." in *ICIP*. IEEE, 2010, pp. 665–668.

Lluís Garrido obtained his PhD in telecommunications at the Polytechnic University of Catalunya (UPC) in 2002. From 2003 to 2010 he worked within the Departament of Information and Communication Technologies of the University Pompeu Fabra (UPF). He currently is associate professor at the University of Barcelona (UB).

Marité Guerrieri received her computer science degree at the University CAESE of Buenos Aires in 1985. She obtained her PhD in computer science at the Polytechnic University of Catalunya in 2009. Currently she is a research member of the GGG (Geometry and Graphics Group) at the University of Girona.

Laura Igual received the degree in Mathematics from the University of Valencia in 2000. She obtained her Ph.D. Thesis in 2006 from the University Pompeu Fabra (UPF) and since then she is a research member at the Computer Vision Center of Barcelona. Since 2009, she is a lecturer at University of Barcelona (UB).