STATIONARY SUBDIVISION SNAKES FOR CONTOUR DETECTION

Rafael Díaz Fuentes, Javier Pino Torres, Victoria Hernández Mederos*, and Jorge C. Estrada Sarlabous
Instituto de Cibernética, Matemática y Física, La Habana, Cuba.

ABSTRACT

In this paper we propose a method for computing the contour of an object in an image using a snake represented as a subdivision curve. The evolution of the snake is driven by its control points which are computed minimizing an energy that pushes the snake towards the boundary of the interest region. Our method profits from the hierarchical nature of subdivision curves, since the unknowns of the optimization process are the few control points of the subdivision curve in the coarse representation and, at the same time, good approximations of the energies and their derivatives are obtained from the fine representation. We develop the theory assuming that the subdivision scheme generating the snake is linear stationary and uniform. To illustrate the performance of our method we develop a computational tool called SubdivisionSnake, which computes the snakes associated with two classical subdivision schemes: the four point scheme and the cubic B-spline. Our experiments using synthetic and real images with SubdivisionSnake confirm that the proposed method is fast and successful.

KEYWORDS: subdivision, snakes, object segmentation.

MSC: 68U10, 65D18, 65D07

RESUMEN

En este artículo se propone un método para calcular el contorno de un objeto en una imagen utilizando una curva snake representada como una curva de subdivisión. La evolución de la curva snake depende de sus puntos de control, los cuales se calculan minimizando una energía que empuja la curva hacia la frontera de la región de interés. El método saca provecho de la naturaleza jerárquica de las curvas de subdivisión, ya que las incógnitas del proceso de optimización son los pocos puntos de control que se utilizan en la representación gruesa de la curva, mientras que la representación fina de la curva permite obtener buenas aproximaciones de las energías y de sus derivadas. La teoría se desarrolla asumiendo que el esquema de subdivisión que genera la curva snake es lineal, estacionario y uniforme. Para ilustrar el desempeño del método se desarrolló una herramienta computacional nombrada SubdivisionSnake, la cual calcula la curva snake correspondiente a dos esquemas clásicos de subdivisión: el esquema de cuatro puntos y el esquema B-spline cúbico. Los experimentos utilizando imágenes sintéticas y reales confirman que el método propuesto es rápido y funciona éxitosamente.

PALABRAS CLAVE: subdivisión, curvas snakes, segmentación de objetos .

[*] vicky@icimaf.cu

1. INTRODUCTION

Active contours or snakes were introduced by Kass et al. in [28] as curves that slither within an image from some initial position towards the contour of the object of interest. Snakes have become very popular in segmentation and tracking applications [7], [18] since they are very flexible and efficient.
The evolution of the snake is formulated as a minimization problem and the corresponding objective functions is usually known as snake energy. During the optimization process, the snake is iteratively updated from a starting position until it reaches the minimum of the energy function. This energy measures the distance between the snake and the boundary of the object. It also controls some desirable properties of the final snake, such as the smoothness, the interpolation of distinguished points, etc. The quality of segmentation is determined by the choice of the energy terms and the starting position of the snake.

Kass et al. [28] originally formulated the snake energy as a linear combination of three terms: the *image energy*, which only depends on the image, the *internal energy*, which ensures the smoothness of the snake, and the *constraint energy*, which allows that the user interacts with the snake. The specific definition of these energies depends on the application, on the nature of the image and also on the representation of the snake. The image energy guides the snake to the boundary of the interest object and it is the most important energy. It is usually defined as a weighted sum of a gradient based energy [28], [33], that provides a good approximation of the contour of the object, and a region based energy [22], [35], that distinguishes different homogeneous regions within the image. Gradient based energies have a narrow zone of attraction in comparison with region based energies. Hence, the success of the segmentation depends on the selection of the weight.

Snakes differ not only in the choice of the energy function but also in the representation of the curve. According to the representation, snakes may be classified as point snakes [28], geodesic snakes [9], [1], [38] and parametric snakes [32], [22],[6], [15]. Point-snakes simply consist of an ordered collection of points. This representation depends on a large number of parameters (the snake points), which makes the optimization expensive. Geodesic snakes are described as the zero level set of a higher-dimensional manifold. This type of active contours is very flexible topologically. In consequence, it is suitable for segmenting objects that have very variable shapes. A drawback of geodesic snakes is that they are expensive from computational point of view. Parametric snakes are smooth curves written as a linear combination of a basis of functions. The coefficients in this representation, known as control points, are few. This makes faster the optimization process. The downside of parametric snakes is that the parametrization restricts the shapes that can be approximated.

In this paper we focus on a particular class of parametric snakes: those obtained from linear and stationary subdivision schemes. Subdivision schemes generate a curve starting from an initial set of control points which, by the iterative application of refinement rules, becomes continuous in the limit. Depending on the particular choice of the subdivision mask, the continuous limit curve may have different degrees of smoothness. In the linear and stationary subdivision schemes, the limit curve is a parametric curve, which can be written as a linear combination of the integer shifts of a basic limit function, that is actually a scaling function. Subdivision curves have the multiresolution property, which means that they can be easily represented in different scales. As a consequence, snakes constructed as subdivision curves provides representations of the contour of a shape with varying resolutions, a property that most of parametric snakes do not have.

1.1. Related work

In the context of image segmentation the most common parametric snakes are those producing B-spline type curves, see for instance: [22], [6], [27] and [17]. But it is important to recall that these papers, and many other where snakes are B-spline curves, never use the fact that B-spline are scaling functions. Therefore, in these articles there is no notion of multiresolution representation of the snake. The use of subdivision curves for segmentation was first proposed in [25], where the so called *tamed snake* is introduced. This snake is generated by the classical four point subdivision scheme [19]. The method incorporates image information considering the control points of the subdivision curve as mass points attracted by edges of the image. The four point subdivision scheme is also used in [32] in combination with the gradient vector flow. After every step of subdivision, the region energy of the subdivision polygon is reversely computed and a local adaptive compensation is carried out, in such a way that regions with high curvature are further subdivided, while flat regions remain unrefined.

In [29] a segmentation method called *SketchSnakes* is proposed. The method combines a general subdivision curve snake with an initialization process based on few sketch lines drawn by the user across the width of the target object. External image forces are computed at the points of the finer level curve and then distributed, using weights derived from the original subdivision rules, among the points of the coarse level. The positions of the control points are updated, new external forces are calculated and the process is repeated until an accurate solution is reached.

More recently, exponential B-spline have been introduced to construct snakes that reproduce circular and elliptical shapes [15], [16], [17], [13], [14]. In [3] subdivision snakes are obtained in a generic way using a multiscale approach to speed up the optimization process and improve robustness. Depending on the selected admissible subdivision mask, the snake may be interpolatory or reproduce trigonometric or polynomial curves. The multiscale approach facilitates to increase the number of points describing the curve as the algorithm progresses to the solution and, at each step, the scale of the image feature is matched to the density of the sample of the curve.

1.2. Our contribution

The proposed image energy, which is completely data driven, is a linear combination of a well known edge-based term using gradient information to detect contours (see [3], [15], [26], [27]) and a new region-based term which uses statistical information to discriminate different homogeneous regions.

Compared with the region-based energy used in [15],[26], [27], [3], in this paper we introduce a new region energy designed to maximize the contrast between the average intensity of the image within the snake and over the complement of the snake in a bounding box. This energy is simpler and computationally cheaper than other similar energies proposed in the literature [10],[26], [27], [35],[15], [4] because the bounding box, containing the object to be segmented, does not change during the optimization. Moreover, the average intensity inside and outside the snake have neither to be estimated a priori, nor to be included among the optimization parameters. Furthermore, we propose also an algorithm to obtain a pixel-level discretization of the snake (see section 4.1.2.). This algorithm allows to compute robust approximations of the region energy for images of either low or high resolution. Finally, as part of this research, we develop a friendly and flexible computational application for object segmentation, using two types of subdivision snakes: cubic B-spline and four points subdivision curves.

2. SUBDIVISION CURVES

2.1. Linear uniform stationary subdivision schemes

Denote by $\mathbf{P}^0$ a polygon in the plane. A *subdivision scheme* is a procedure that refines $\mathbf{P}^0$ producing a sequence of polygons $\mathbf{P}^1, \mathbf{P}^2, \ldots$ with an increasing number of vertices. A *linear, stationary, uniform* and *binary* subdivision scheme is based on the application of a refinement rule

$$\mathbf{p}_i^{k+1} = \sum_{j\in\mathbb{Z}} a_{i-2j}\mathbf{p}_j^k \tag{2.1}$$

that computes the vertices $\mathbf{p}_i^k = (x_i^k, y_i^k)$ of the polygon $\mathbf{P}^k$ in the step k as a linear function of the vertices of $\mathbf{P}^{k-1}$. The coefficients $\mathbf{a} = \{a_i \in \mathbb{R}, i \in \mathbb{Z}\}$ in (2.1)) are a called *subdivision mask*. In practice, only a finite number of coefficients are different from zero. The subdivision scheme converges if the sequence of piecewise linear functions $\mathbf{f}^k(t)$ which satisfies the interpolation conditions

$$\mathbf{f}^k\left(\frac{i}{2^k}\right) = \mathbf{p}_i^k, \quad i \in \mathbb{Z} \tag{2.2}$$

converges uniformly. Denote by $\mathbf{r}(t) = (x(t), y(t))$ the continuous limit function

$$\mathbf{r}(t) = \lim_{k\to\infty} \mathbf{f}^k(t). \tag{2.3}$$

This limit exists as long as the subdivision scheme applied to the functional data $\delta = \{\delta_{i,0}, i \in \mathbb{Z}\} = \{\ldots, 0, 0, 1, 0, 0, \ldots\}$ converges. The corresponding limit function $\varphi(t)$ is called *basic limit function* and satisfies the *refinement equation*

$$\varphi(t) = \sum_{j\in\mathbb{Z}} a_j \varphi(2t - j). \tag{2.4}$$

Due to the linearity of the subdivision rules, the subdivision curve $\mathbf{r}(t)$ can be written as a linear combination of the integer shifts of $\varphi(t)$,

$$\mathbf{r}(t) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^0 \varphi(t - j). \tag{2.5}$$

This representation can be used to define also the tangent vector to the curve (as well as the normal vector),

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}(t) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^0 \frac{\mathrm{d}}{\mathrm{d}t}\varphi(t - j). \tag{2.6}$$

Even more, after (2.1), for any $k \geq 0$ the subdivision curve may be expressed as,

$$\mathbf{r}(t) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^k \varphi(2^k t - j). \tag{2.7}$$

From (2.7) we observe that,

$$\mathbf{r}\left(\frac{i}{2^k}\right) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^0 \varphi\left(\frac{i}{2^k} - j\right) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^k \varphi(i-j). \tag{2.8}$$

In case of the interpolatory subdivision schemes, as $\varphi(i-j) = \delta_j^i$, then it holds that,

$$\mathbf{r}\left(\frac{i}{2^k}\right) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^k \delta_j^i = \mathbf{p}_i^k. \tag{2.9}$$

Also, according to (2.2) and (2.3) it holds,

$$\mathbf{r}\left(\frac{i}{2^k}\right) = \lim_{q\to\infty} \mathbf{f}^{k+q}\left(\frac{2^q i}{2^{k+q}}\right) = \lim_{q\to\infty} \mathbf{p}_{2^q i}^{k+q}, \qquad q \in \mathbb{N}. \tag{2.10}$$

In case of interpolatory schemes the center and right hand expressions are limits of constant sequences that converge to the point $\mathbf{p}_i^k$ as stated in (2.9). Otherwise, the right hand sequence allows to analyze the exact value of $\mathbf{r}\left(i/2^k\right)$, as it is done in (2.19).

Remark 2.1. *A subdivision curve $\mathbf{r}(t)$ is usually represented by a polygon $\mathbf{P}^k$ whose vertices are obtained after some refinements of an initial polygon $\mathbf{P}^0$. As k increases the polygon $\mathbf{P}^k$ provides a better approximation of $\mathbf{r}(t)$. In this work we approximate $\mathbf{r}(t)$ by the polygon $\{\mathbf{r}(i/2^k)\}$ whose vertices are on the curve. For interpolatory subdivision schemes both $\mathbf{P}^k$ and $\{\mathbf{r}(i/2^k)\}$ polygons are the same, but they are different in the case of non-interpolatory schemes.*

The expression (2.8) allows two options for the evaluation of the curve in dyadic parameters. The first one uses the initial control polygon $\mathbf{P}^0$ and computes the evaluations $\varphi\left(\frac{i}{2^k} - j\right)$. The second subdivides the initial polygon k times and uses the polygon $\mathbf{P}^k$. We choose the first option, since the evaluation of $\varphi\left(\frac{i}{2^k} - j\right)$ for any $i, j \in \mathbb{Z}$ and $k \in \mathbb{N}$ can be done *a priori* storing the results in a lookup table. The evaluation of the basic function at dyadic parametric values $\varphi\left(\frac{i}{2^k} - j\right)$ for any $i, j \in \mathbb{Z}$ and $k \in \mathbb{N}$ can be computed as the subdivision of the polygon with vertices $\mathbf{P}^0 = \left\{(i, \delta_i^0),\ i \in \mathbb{Z}\right\}$ by k times (see Figures 1 and 2).

In this work we are interested in *closed curves* that approximate the boundary of a region in a digital image. Hence, if the curve $\mathbf{r}(t)$ has a control polygon with M vertices $\mathbf{p}_i^0$, $i = 0, \ldots, M-1$, then the polygon is periodically extended assuming that $\mathbf{p}_{i+M}^0 = \mathbf{p}_i^0$ for all $i \in \mathbb{Z}$. Under this assumption, we obtain from (2.5))

$$\begin{aligned} \mathbf{r}(t+M) &= \sum_{j\in\mathbb{Z}} \mathbf{p}_j^0 \varphi(t+M-j) = \sum_{i\in\mathbb{Z}} \mathbf{p}_{i+M}^0 \varphi(t-i) \\ &= \sum_{i\in\mathbb{Z}} \mathbf{p}_i^0 \varphi(t-i) = \mathbf{r}(t), \end{aligned} \tag{2.11}$$

i.e., the subdivision curve is periodic with period M. For more details about subdivision schemes, see [20].

2.2. Exact evaluation of linear uniform stationary subdivision schemes

To illustrate the performance of our method we discuss in details two classical subdivision schemes: the four point and the cubic B-spline. The first one is interpolatory while the second is non-interpolatory. Since the interpolation of the points provided by the user is the more natural starting point for the snake, we explain in the case of the cubic B-spline how to compute the initial control polygon in such a way that the B-spline curve passes through the given set of points.
In the next sections we describe the subdivision schemes chosen to generate the snake curves. Moreover, we provide the expressions needed to evaluate the subdivision curves and their derivatives at dyadic parameters.

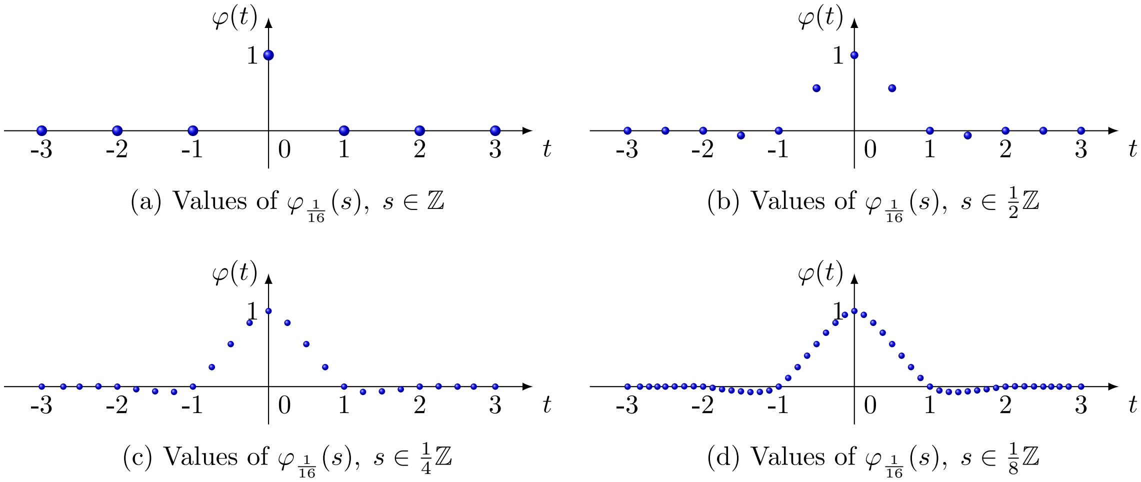
2.2.1. Four points subdivision scheme

The four point subdivision scheme [19], also known as DLG, is a linear, stationary and uniform subdivision scheme, depending on a tension parameter ω. The rules that define this scheme are

$$\mathbf{p}_{2i}^{k+1} = \mathbf{p}_i^k \tag{2.12}$$

$$\mathbf{p}_{2i+1}^{k+1} = \left(\omega + \frac{1}{2}\right)\left(\mathbf{p}_i^k + \mathbf{p}_{i+1}^k\right) - \omega\left(\mathbf{p}_{i-1}^k + \mathbf{p}_{i+2}^k\right) \tag{2.13}$$

The scheme is *interpolatory* since the rule (2.12) implies that the set of points of the step $k+1$ contains the points of the previous step. Hence, the control points $\mathbf{p}_i^0$ are interpolated by functions $\mathbf{f}^k(t)$ for all $k \geq 0$ and thus they belong to the limit curve that we denote by $\mathbf{r}_\omega(t)$, to recall that it depends on the free parameter ω. This curve is continuous if ω is in the interval $(0, \frac{1}{4})$ and it has a continuous tangent vector when $\omega \in \left(0, \frac{\sqrt{5}-1}{8}\right)$. The basic limit function φ_ω of the DLG scheme has support $[-3, 3]$, as it is shown in Figure 1 for $\omega = \frac{1}{16}$.



(a) Values of $\varphi_{\frac{1}{16}}(s),\ s \in \mathbb{Z}$

(b) Values of $\varphi_{\frac{1}{16}}(s),\ s \in \frac{1}{2}\mathbb{Z}$

(c) Values of $\varphi_{\frac{1}{16}}(s),\ s \in \frac{1}{4}\mathbb{Z}$

(d) Values of $\varphi_{\frac{1}{16}}(s),\ s \in \frac{1}{8}\mathbb{Z}$

Figure 1: Generating the values of the basic function for the 4-point subdivision scheme.

Since the subdivision rules (2.12)-(2.13) depend on 4 points to get a closed curve we need a closed polygon composed by $M+3$ points $\mathbf{P}^0 = \{\mathbf{p}_{-1}^0, \mathbf{p}_0^0, \ldots, \mathbf{p}_{M-1}^0, \mathbf{p}_M^0, \mathbf{p}_{M+1}^0\}$, where $\mathbf{p}_{-1}^0 = \mathbf{p}_{M-1}^0$,

$\mathbf{p}_M^0 = \mathbf{p}_0^0$ and $\mathbf{p}_{M+1}^0 = \mathbf{p}_1^0$. Hence, if the initial polygon has M vertices, then expression (2.5) for the subdivision curve is reduced to

$$\mathbf{r}_\omega(t) = \sum_{j=-1}^{M+1} \mathbf{p}_j^0 \varphi_\omega(t-j), \quad 0 \le t \le M \tag{2.14}$$

where $\varphi_\omega(t)$ is the basic limit function of the four point subdivision scheme with parameter ω.

If we denote by $\mathbf{t}_i^k$ the tangent vector to the subdivision curve at $\mathbf{p}_i^k$, then it holds:

$$\mathbf{t}_i^k = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}_\omega(t)\Big|_{t=\frac{i}{2^k}} = \frac{2^k}{1-4\omega}\left(\frac{1}{2}(\mathbf{p}_{i+1}^k - \mathbf{p}_{i-1}^k) - \omega(\mathbf{p}_{i+2}^k - \mathbf{p}_{i-2}^k)\right). \tag{2.15}$$

Remark 2.2. *In practice, we use* (2.15) *only to compute* $\varphi'(i/2^k)$*, obtained with* $\{\mathbf{p}_i^0 = (i, \delta_i^0)\}$*. The tangent vector for any subdivision curve is computed by using* (2.6)*.*

Recall that for $i = 0, \dots, 2^k M - 1$ the subdivision curve $\mathbf{r}_\omega(t)$ of the DLG scheme satisfies

$$\mathbf{r}_\omega\left(\frac{i}{2^k}\right) = \mathbf{p}_i^k, \qquad \frac{d\mathbf{r}_\omega}{dt}\left(\frac{i}{2^k}\right) = \mathbf{t}_i^k. \tag{2.16}$$

In what follows we use the notation $\mathbf{t}_i^k = (tx_i^k, ty_i^k)$.
The best value for the parameter ω with respect to the regularity of the limit curve is $\omega = \frac{1}{16}$ [21]. In the rest of this paper we consider only this case.

2.2.2. Cubic B-spline subdivision scheme

This linear, stationary and uniform subdivision scheme is defined by the rules:

$$\mathbf{p}_{2i}^{k+1} = \tfrac{1}{8}\mathbf{p}_{i-1}^k + \tfrac{6}{8}\mathbf{p}_i^k + \tfrac{1}{8}\mathbf{p}_{i+1}^k \tag{2.17}$$

$$\mathbf{p}_{2i+1}^{k+1} = \tfrac{1}{2}\mathbf{p}_i^k + \tfrac{1}{2}\mathbf{p}_{i+1}^k \tag{2.18}$$

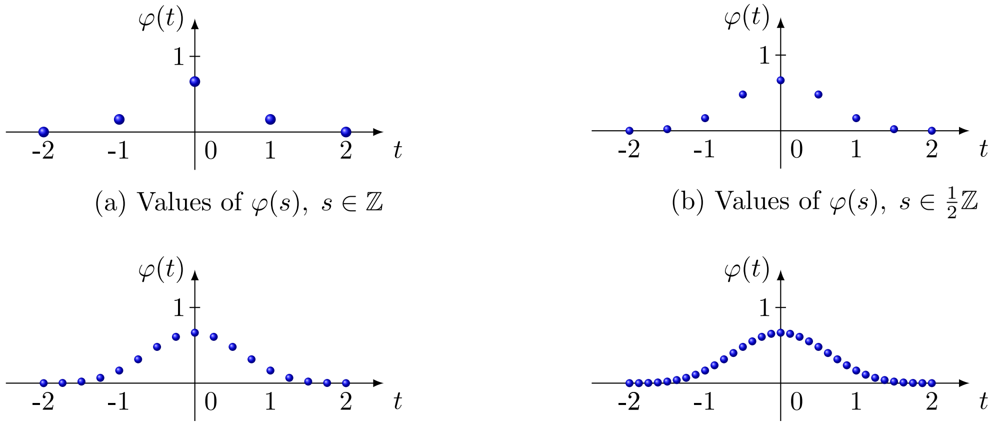
and generates as limit a cubic B-spline curve that is C^2-continuous. The basic limit function φ for this scheme has support $[-2, 2]$, as it is shown in Figure 2.
Since this scheme is not interpolatory, the points in $\mathbf{P}^k$ don't belong to the limit curve. Nevertheless, following (2.10) it can be proved [34] that:

$$\mathbf{r}\left(\frac{i}{2^k}\right) = \tfrac{1}{6}\mathbf{p}_{i-1}^k + \tfrac{4}{6}\mathbf{p}_i^k + \tfrac{1}{6}\mathbf{p}_{i+1}^k. \tag{2.19}$$

Remark 2.3. *Recall that in this work, we don't compute the values* $\mathbf{r}\left(i/2^k\right)$ *using* (2.19) *for any polygon* $\mathbf{P}^0$ *and its refinements. Instead, we store in a lookup table the pre-computed values of* $\varphi\left(\frac{i}{2^k}\right)$ *for* $i \in \mathbb{Z}$ *and a previous fixed value of* $k \in \mathbb{N}$*, obtained from the initial data* $\mathbf{P}^0 = \left\{(i, \delta_i^0),\ i \in \mathbb{Z}\right\}$*. Then, we use:*

$$\mathbf{r}\left(\frac{i}{2^k}\right) = \sum_{j\in\mathbb{Z}} \mathbf{p}_j^0 \varphi\left(\frac{i}{2^k} - j\right).$$

Taking into account that $\mathbf{P}^0$ *changes during the optimization of the snake the previous strategy reduces the computational cost.*



(a) Values of $\varphi(s),\ s \in \mathbb{Z}$

(b) Values of $\varphi(s),\ s \in \frac{1}{2}\mathbb{Z}$

(c) Values of $\varphi(s),\ s \in \frac{1}{4}\mathbb{Z}$

(d) Values of $\varphi(s),\ s \in \frac{1}{8}\mathbb{Z}$

Figure 2: Generating the values of the basic function for the cubic B-spline subdivision scheme.

Since the subdivision rules (2.17)-(2.18) depend on 3 points to get a closed curve we need a closed polygon composed by $M+2$ points $\mathbf{P}^0 = \{\mathbf{p}^0_{-1}, \mathbf{p}^0_0, \dots, \mathbf{p}^0_{M-1}, \mathbf{p}^0_M\}$, where $\mathbf{p}^0_{-1} = \mathbf{p}^0_{M-1}$ and $\mathbf{p}^0_M = \mathbf{p}^0_0$. Hence, if the initial polygon has M vertices, then the expression (2.5) for the subdivision curve is reduced to,

$$\mathbf{r}(t) = \sum_{j=-1}^{M} \mathbf{p}^0_j \varphi(t-j), \quad 0 \leq t \leq M \tag{2.20}$$

where $\varphi(t)$ is the basic limit function of cubic B-spline subdivision scheme.
If we denote by $\mathbf{t}^k_i$ the tangent vector to the subdivision curve at $\mathbf{p}^k_i$, then,

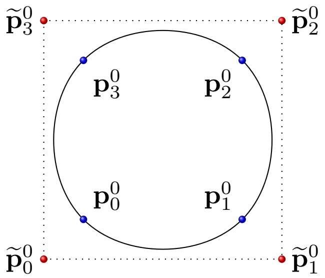
$$\mathbf{t}^k_i = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}(t)\bigg|_{t=\frac{i}{2^k}} = \frac{1}{2}\left(\mathbf{p}^k_{i+1} - \mathbf{p}^k_{i-1}\right). \tag{2.21}$$

The use of this expression follows the same argument as in *Remark 2.2.*.

Approximating subdivision curves do not interpolate their control points. Since control points are the degrees of freedom, this property makes approximating subdivision snakes less intuitive in interactive segmentation than interpolating subdivision snakes. To overcome this limitation, we explain now how to compute the control points $\widetilde{\mathbf{P}}^0$ of the cubic B-spline curve interpolating the vertices of the initial polygon introduced by the user. This new strategy, different from other works as [6], unifies the treatment of approximating and interpolating subdivision snakes, making the initialization and interaction more user-friendly.
The control polygon $\widetilde{\mathbf{P}}^0$ depends linearly on the polygon $\mathbf{P}^0$ as it is shown in the following.

Theorem 2.4. *The subdivision curve generated by the cubic B-spline scheme that interpolates the set of points $\mathbf{P}^0$ has control points $\widetilde{\mathbf{P}}^0$ given by,*

$$\widetilde{\mathbf{P}}^0 = \mathcal{A}\mathbf{P}^0, \tag{2.22}$$



$\widetilde{\mathbf{p}}_3^0$
$\widetilde{\mathbf{p}}_2^0$
$\mathbf{p}_3^0$
$\mathbf{p}_2^0$
$\mathbf{p}_0^0$
$\mathbf{p}_1^0$
$\widetilde{\mathbf{p}}_0^0$
$\widetilde{\mathbf{p}}_1^0$

Figure 3: Interpolation of given set of points $\mathbf{P}^0$ by the cubic B-spline subdivision scheme

where the elements of matrix $\mathcal{A} = [a_{s,t}]_{s=0,t=0}^{M-1,M-1}$ *are given by:*

$$a_{s,t} = \begin{cases} \frac{1}{M} + \frac{3}{M}\cos(s\pi) + \frac{2}{M}\displaystyle\sum_{j=1}^{\frac{M}{2}-1}\left(\frac{2}{3}+\frac{1}{3}\cos(2j\pi/M)\right)^{-1}\cos(2(s-t)\pi/M), & \textit{for} \quad M \mod 2 = 0, \\ \frac{1}{M} + \frac{2}{M}\displaystyle\sum_{j=1}^{\lfloor\frac{M}{2}\rfloor}\left(\frac{2}{3}+\frac{1}{3}\cos(2j\pi/M)\right)^{-1}\cos(2(s-t)\pi/M), & \textit{for} \quad M \mod 2 = 1. \end{cases}$$

See the proof in *Appendix A*. Figure 3 shows the points $\mathbf{P}^0$ to be interpolated by the cubic B-spline and the control polygon $\widetilde{\mathbf{P}}^0$ computed using (2.22).

3. SNAKE ENERGIES

In the rest of the paper we denote by Γ the object to be segmented in the given digital image and by $\partial\Gamma$ its boundary. Our main goal is to compute a snake that approximates automatically $\partial\Gamma$. In the literature, the evolution of the snake is driven by the minimization of several energies that measure the proximity between the snake and $\partial\Gamma$ and also some desirable properties of the snake like the smoothness, the interpolation of distinguished points and so on.

Since our snake is a subdivision curve, the total energy E_{snake}, depends on the initial control polygon $\mathbf{P}^0$ of the snake in the representation (2.5). The control polygon $\mathbf{P}_*^0$ of the optimal snake is computed as:

$$\mathbf{P}_*^0 = arg \min_{\mathbf{P}^0} E_{\text{snake}}(\mathbf{P}^0). \tag{3.1}$$

Here we assume that the region of interest Γ to be segmented is dark in comparison to the background. Hence, the energy functionals related with the image are designed to detect dark objects on a brighter background. This is not a limitation of the proposed method, since it can be easily adapted to the contrary case.

All the energies are defined by integrals of functions which are computed approximately. To obtain good approximations we use a large sample of points on the subdivision curve. In the following sections we develop the expressions for each energy.

Remark 3.1. *Images are represented in a system of coordinates defined by rows and columns, like the indexing of a matrix. Thus, if a pixel has coordinates* (x, y)*, the* x*-coordinate refers to the row and* *the y-coordinate refers to the column (see for example Figure 4). This does not affect the definition and use of the subdivision schemes, since each coordinate in* (2.5) *works independently.*

3.1. Gradient energy

If $I(x,y)$ denotes the image intensity at a pixel with coordinates (x,y) and $\mathbf{r}(t) = (x(t), y(t))$ is a parametric curve living on the image for $t \in [0, M]$, the simplest image energy is the *gradient magnitude energy* E_{mag} given by:

$$E_{mag}(\mathbf{r}(t)) = -\int_0^M \|\nabla I(\mathbf{r}(t))\|^2 \, dt \tag{3.2}$$

where $\|\nabla I(\mathbf{r}(t))\|^2 = \left(\frac{\partial I}{\partial x}(x(t), y(t))\right)^2 + \left(\frac{\partial I}{\partial y}(x(t), y(t))\right)^2$. Since the gradient magnitude energy only depends on the magnitude of the gradient vector, the minimization of (3.2) can misguide the snake to a neighboring object if the initial approximation is not very close to the boundary of interest. To overcome this limitation, several alternative energies has been proposed, like balloon forces [12], gradient vector-fields [37], [26],[27] or multiresolution approaches [6].

In this paper we use the gradient based image energy E_{grad} proposed in [26]. The idea behind this approach is the following. If we travel around the ground truth boundary curve $\partial\Gamma$ in counterclockwise direction, then Γ is always on the "left", i.e in the direction of $-\nabla I$. Hence, we pull the snake in the direction of $\partial\Gamma$, requiring the normal to snake at any point to be parallel to $-\nabla I$ at the same point. More precisely, if we denote by $\mathbf{n}(t)$ the inward unit normal to snake at the point $\mathbf{r}(t)$, then the new energy E_{grad}, which takes into account not only the magnitude of the image gradient but also its direction is given by:

$$E_{grad}(\mathbf{r}(t)) = -\int_0^M \langle \nabla I(\mathbf{r}(t)), \left\| \frac{\mathrm{d}\mathbf{r}(t)}{\mathrm{d}t} \right\| \mathbf{n}(t) \rangle \, dt \tag{3.3}$$

where $\langle \cdot, \cdot \rangle$ is the usual scalar product and $\frac{\mathrm{d}\mathbf{r}(t)}{\mathrm{d}t}$ denotes the tangent to $\mathbf{r}(t)$. Expanding (3.3) we obtain

$$E_{grad}(\mathbf{r}(t)) = -\int_0^M \left(\frac{\partial I}{\partial x}(x(t), y(t)) \frac{\mathrm{d}y(t)}{\mathrm{d}t} - \frac{\partial I}{\partial y}(x(t), y(t)) \frac{\mathrm{d}x(t)}{dt} \right) \, dt. \tag{3.4}$$

To obtain good approximations of the energies (and their derivatives with respect to the coordinates of control points) we use a large sample of points on the subdivision curve. More precisely, given the initial polygon $\mathbf{P}^0 = \{\mathbf{p}_0^0, \ldots, \mathbf{p}_{M-1}^0\}$, we select k (in our experiments we take $k = 4$ or $k = 5$) and we use (2.5) to generate $2^k M$ points $\mathbf{r}(i/2^k)$, $i = 0, \ldots, 2^k M - 1$ on the subdivision curve. Moreover, we apply bilinear interpolation on the gradient of the image to compute $\nabla I(\mathbf{r}(i/2^k))$. Finally, we approximate the energy substituting the integral in (3.4) by the average of values of the integrand over the sample of $2^k M$ points on the subdivision curve corresponding to parameter values $\frac{i}{2^k}$, $i = 0, \ldots, 2^k M - 1$.

Taking into account (2.5) we obtain[1] the following approximation of (3.4):

$$E_{grad}(\mathbf{P}^0) \approx \frac{1}{2^k M} \sum_{i=0}^{2^k M - 1} \left[\frac{\partial I}{\partial y} \left(\sum_{j \in \mathbb{Z}} \mathbf{p}_j^0 \varphi(\tfrac{i}{2^k} - j) \right) tx_i^k - \frac{\partial I}{\partial x} \left(\sum_{j \in \mathbb{Z}} \mathbf{p}_j^0 \varphi(\tfrac{i}{2^k} - j) \right) ty_i^k \right] \tag{3.5}$$

[1] Recall that the indices of the inner summations depend on the choice of the subdivision scheme, see (2.14) and (2.20).

where $tx_i^k = \frac{\mathrm{d}x}{\mathrm{d}t}\left(i/2^k\right)$ and $ty_i^k = \frac{\mathrm{d}y}{\mathrm{d}t}\left(i/2^k\right)$, so that $\left(tx_i^k, ty_i^k\right) = \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t}\left(i/2^k\right)$.
It should be noticed that the right hand side of (3.5) is a function of the coordinates of the initial control points $\mathbf{P}^0$.

3.2. Region energy

The main disadvantage of gradient based energy (3.3) is that its zone of attraction is limited, since the gradient is small as long as we move away from $\partial\Gamma$. To face this problem several region energies have been introduced in the literature [33], [11], [10],[35], [15]. Some of them use statistical information to identify different regions [26], [27],[35], [4].
Inspired in the energies proposed in [10], [15] and [35] we introduce here a simple region energy E_{reg}, designed to maximize the contrast between the average intensity of the pixels within the snake and the average intensity in the region outside the snake and inside a given bounding box. We denote by Ω the region enclosed by the snake $\mathbf{r}(t) = (x(t), y(t))$, $t \in [0, M]$. Moreover, we assume that a rectangular region R, containing in its interior the object Γ to be segmented and also the initial control polygon $\mathbf{P}^0$ of the snake, has been selected. Let $|R|$ be the area of R and let $|\Omega|$ be the area of Ω (which may vary while the snake evolves). Then the new region energy E_{reg} to be minimized is given by,

$$E_{reg}(\mathbf{P}^0) := -\left(\frac{\int\int_{\Omega} I(x,y)\mathrm{d}x\mathrm{d}y}{|\Omega|} - \frac{\int\int_{R\setminus\Omega} I(x,y)\mathrm{d}x\mathrm{d}y}{|R| - |\Omega|}\right)^2. \tag{3.6}$$

Recall that minimizing E_{reg} is equivalent to maximize the difference between the average intensity inside Ω and the average intensity in the complement of Ω in R.
The region energy (3.6) may be written as,

$$E_{reg}(\mathbf{P}^0) = -\left(\frac{I_\Omega}{|\Omega|} - \frac{I_R - I_\Omega}{|R|-|\Omega|}\right)^2. \tag{3.7}$$

where

$$I_\Omega = \int\int_\Omega I(x,y)\mathrm{d}x\mathrm{d}y \quad \text{and} \quad I_R = \int\int_R I(x,y)\mathrm{d}x\mathrm{d}y$$

denote the average intensity of the image inside $\mathbf{r}(t)$ and R respectively.
Observe that even when our region energy (3.6) is similar to the energy proposed in [15], there are some substantial differences. First, in [15] the enclosing ellipse $\mathbf{r}_\lambda$ (boundary of the region enclosing the object to be segmented) is a dilatation of the best elliptical approximation $\mathbf{r}_e$ to the active contour $\mathbf{r}$. As the active contour $\mathbf{r}$ evolves, $\mathbf{r}_e$ and $\mathbf{r}_\lambda$ change, thus in each iteration of the optimization process, the area $|\Omega|$, the average intensity I_Ω of the image inside $\mathbf{r}$, and the average intensity of the image in the complement of the region enclosed by $\mathbf{r}$ with respect to the region enclosed by $\mathbf{r}_\lambda$ must be updated. In comparison, our approach based on region energy (3.6) is computationally less expensive because the enclosing reference region (containing the active contour and the object) is a rectangle R, selected initially by the user, that remains fixed during the evolution of the snake. As a consequence, its area $|R|$ and the average intensity I_R of the image inside R are computed only one time. Hence, as the active contour $\mathbf{r}$ evolves, in each iteration of the optimization process, only $|\Omega|$ and I_Ω must be updated in (3.7).

Since region energies are usually expressed as integrals of a function over the domain Ω enclosed by the snake, some authors propose the use of Green's theorem to rewrite the 2D integrals as line integrals along the snake [15],[17],[27]. In particular, if we apply it to the function $I(x,y)$ we obtain

$$I_\Omega = \int\int_\Omega I(x,y)\mathrm{d}x\mathrm{d}y = \int_{\partial\Omega} I_1(x,y)\mathrm{d}y = -\int_{\partial\Omega} I_2(x,y)\mathrm{d}x \tag{3.8}$$

where

$$I_1(x,y) = \int_{-\infty}^{x} I(\tau,y)\mathrm{d}\tau \quad \text{and} \quad I_2(x,y) = \int_{-\infty}^{y} I(x,\tau)\mathrm{d}\tau. \tag{3.9}$$

Thus, if $\partial\Omega$ is parametrized by $\mathbf{r}(t) = (x(t), y(t)),\ 0 \le t \le M$, then from (3.8)) it holds

$$I_\Omega = \int_0^M I_1(x(t),y(t))\,\frac{\mathrm{d}y(t)}{\mathrm{d}t}\mathrm{d}t = -\int_0^M I_2(x(t),y(t))\,\frac{\mathrm{d}x(t)}{\mathrm{d}t}\mathrm{d}t. \tag{3.10}$$

This approach reduces significantly the computational cost, but in our experiments we have found that large errors may be introduced when we use it to compute the integrals in (3.6), in a digital images context. In papers based on this approach, line integrals (3.9) are approximated using a sample of points on the snake and summing up the contributions of column or row image pixel strips corresponding to each point on the snake. But even if the snake is parametrized by a multiple of the arc length, the distribution on the image of the sample of points may be very irregular. For instance, if the image has low resolution then some points may belong to the same pixel overestimating the value of the integral. On the contrary, if the image has high resolution then those rows or columns of Ω without any point of the sample do not contribute to the computation producing an underestimate of the integral. We propose instead a sort of *rasterization* of $\partial\Omega$ in order to describe it and compute then (3.6) by means of the pixels in Ω and their values of intensity (see Figure 8). It should be noticed that the subdivision curve $\mathbf{r}(t)$ is represented as a polygon with vertices in $\{\mathbf{r}(i/2^k), i = 0,\ldots,2^kM-1\}$ living on the image. Then, it should be observed that $\mathbf{r}(i/2^k) = (x_i^k, y_i^k)$ is represented on the image by the pixel with coordinates $(\lceil x_i^k \rceil, \lceil y_i^k \rceil)$.

In our rasterization algorithm of the snake (detailed in Section 4.1.2.), the integral of the intensity is computed approximately summing up (with sign) the contribution of each horizontal image strip intersected by Ω, see Figure 4. The value l_j^i is the index of the column of the pixel that results from the intersection of the edge $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$ with the j-th row of the image. Then,

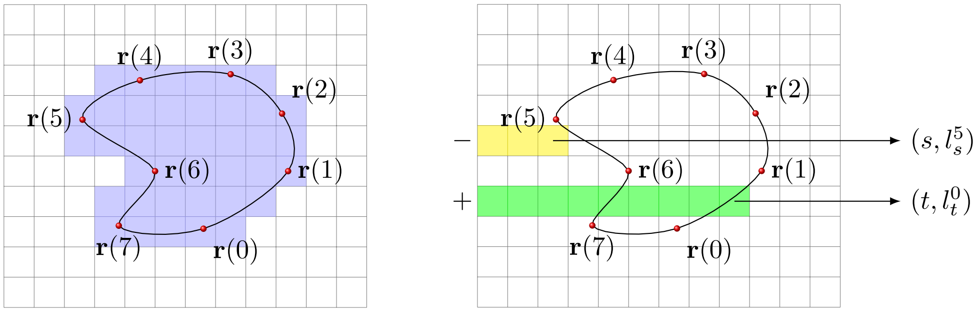
$$I_\Omega = \int\int_\Omega I(x,y)\ \ dx\ dy \approx \sum_{i=0}^{2^kM-1} \mathrm{sign}(x_i^k - x_{i+1}^k) \sum_{j=\lceil x_i^k \rceil}^{\lceil x_{i+1}^k \rceil} \sum_{l=1}^{l_j^i} I(l,j). \tag{3.11}$$

The values of $\sum_{l=1}^{l_j^i} I(l,j)$ can be pre-computed in a lookup table to speed up the implementation.

In particular, the approximation of the area of Ω enclosed by the subdivision curve is,

$$|\Omega| = \int\int_\Omega \ dx\ dy \approx \sum_{i=0}^{2^kM-1} \mathrm{sign}(x_i^k - x_{i+1}^k) \sum_{j=\lceil x_i^k \rceil}^{\lceil x_{i+1}^k \rceil} l_j^i. \tag{3.12}$$

Finally, (3.11) and (3.12) are substituted in (3.7) to obtain the approximation of the region energy.



(a) Representing Ω (b) The strips used to approximate the integral in E_{reg}

Figure 4: Pixels considered to compute the *region-based energy*. Observe that pixel's coordinates are in the coordinate system of the image *(row, column)*.

3.3. Optimization

To obtain the optimal position of the control points of the snake we minimize the total energy given by,

$$E_{\text{snake}}(\mathbf{P}^0) = \alpha E_{grad}(\mathbf{P}^0) + (1-\alpha)E_{reg}(\mathbf{P}^0). \tag{3.13}$$

The optimization problem is solved using the BFGS Quasi-Newton method with a cubic line search procedure. This method requires the gradient of the snake energy with respect to the variables of our problem: the coordinates (x_j^0, y_j^0) of the control points $\mathbf{p}_j^0$, $j = 0, \ldots, M-1$. In this section we give the expressions of the approximations of partial derivatives of each energy with respect to each coordinate x_j^0 and y_j^0.

3.3.1. Derivatives of gradient energy

From (3.4) we obtain (see *Appendix B* for more details),

$$\frac{\partial E_{grad}}{\partial x_j^0} = \int_0^M \left(\left(\frac{\partial^2 I}{\partial x \partial y} \frac{\partial x}{\partial t} - \frac{\partial^2 I}{\partial x^2} \frac{dy}{dt} \right) \frac{\partial x}{\partial x_j^0} + \frac{\partial I}{\partial y} \frac{\partial \left(\frac{dx}{dt}\right)}{\partial x_j^0} \right) dt. \tag{3.14}$$

Substituting the integral in (3.14) by the average of the integrand evaluated in the parameter values $i/2^k$, $i = 0, \ldots, 2^k M - 1$ we obtain (see more details in *Appendix B*) the following approximation for the partial derivative of gradient energy with respect to x_j^0,

$$\frac{\partial E_{grad}}{\partial x_j^0} \approx \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \left(\frac{\partial^2 I}{\partial x \partial y} \left(\mathbf{r}\left(\frac{i}{2^k}\right)\right) t_{ix}^k - \frac{\partial^2 I}{\partial x^2} \left(\mathbf{r}\left(\frac{i}{2^k}\right)\right) t_{iy}^k \right) \varphi\left(\frac{i}{2^k} - j\right) + \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \frac{\partial I}{\partial y} \left(\mathbf{r}\left(\frac{i}{2^k}\right)\right) \varphi^{'}\left(\frac{i}{2^k} - j\right). \tag{3.15}$$

Proceeding in a similar way, from (3.4) we obtain,

$$\frac{\partial E_{grad}}{\partial y_j^0} = \int_0^M \left(\left(\frac{\partial^2 I}{\partial y^2} \frac{\partial x}{\partial t} - \frac{\partial^2 I}{\partial x \partial y} \frac{dy}{dt} \right) \frac{\partial y}{\partial y_j^k} + \frac{\partial I}{\partial x} \frac{\partial \left(\frac{dy}{dt}\right)}{\partial y_j^k} \right) \text{dt}. \tag{3.16}$$

Discretizing the integral with the same procedure, from (3.16) we obtain the following approximation for the partial derivative of gradient energy with respect to y_j^0,

$$\frac{\partial E_{grad}}{\partial y_j^0} \approx \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \left(\frac{\partial^2 I}{\partial y^2} \left(\mathbf{r} \left(\frac{i}{2^k} \right) \right) t_{ix}^k - \frac{\partial^2 I}{\partial x \partial y} \left(\mathbf{r} \left(\frac{i}{2^k} \right) \right) t_{iy}^k \right) \varphi \left(\frac{i}{2^k} - j \right) + \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \frac{\partial I}{\partial x} \left(\mathbf{r} \left(\frac{i}{2^k} \right) \right) \varphi^{'} \left(\frac{i}{2^k} - j \right). \quad (3.17)$$

3.3.2. Derivatives of region energy

In order to find the optimal control polygon we have to compute the partial derivatives of E_{reg} with respect to the coordinates (x_j^0, y_j^0) of the control points $\{\mathbf{p}_j^0, j = 0, \ldots, M-1\}$. Since I_R is constant, from (3.7) we obtain,

$$\frac{\partial E_{reg}}{\partial x_j^0} = -2D \left(\frac{\partial A}{\partial x_j^0} - \frac{\partial B}{\partial x_j^0} \right) \quad (3.18)$$

where

$$A := \frac{I_\Omega}{|\Omega|}, \quad B := \frac{I_R - I_\Omega}{|R| - |\Omega|} \quad \text{and} \quad D := A - B.$$

Proceeding as it is shown in *Appendix B* it can be proved that (3.18) is equals to,

$$\frac{\partial E_{reg}}{\partial x_j^0} = -2D \int_0^M \left(G - H\, I(\mathbf{r}(t)) \right) \varphi(t-j) y'(t) \mathrm{d}t \quad (3.19)$$

where

$$G := \frac{I_\Omega}{|\Omega|^2} + \frac{I_R - I_\Omega}{(|R| - |\Omega|)^2} \quad \text{and} \quad H := \frac{1}{|\Omega|} + \frac{1}{|R| - |\Omega|}.$$

Proceeding in a similar way and deriving in the second equality of (3.8) it is easy to check that

$$\frac{\partial E_{reg}}{\partial y_j^0} = 2D \int_0^M \left(G - H\, I(\mathbf{r}(t)) \right) \varphi(t-j) x'(t) \mathrm{d}t. \quad (3.20)$$

In practice, we approximate (3.19) and (3.20) by

$$\frac{\partial E_{reg}}{\partial x_j^0} \approx -\frac{\widetilde{D}}{2^{k-1} M} \sum_{i=0}^{2^k M-1} \left[\widetilde{G} - \widetilde{H} I \left(\mathbf{r} \left(\frac{i}{2^k} \right) \right) \right] \varphi \left(\frac{i}{2^k} - j \right) y' \left(\frac{i}{2^k} \right)$$

$$\frac{\partial E_{reg}}{\partial y_j^0} \approx \frac{\widetilde{D}}{2^{k-1} M} \sum_{i=0}^{2^k M-1} \left[\widetilde{G} - \widetilde{H} I \left(\mathbf{r} \left(\frac{i}{2^k} \right) \right) \right] \varphi \left(\frac{i}{2^k} - j \right) x' \left(\frac{i}{2^k} \right)$$

where $\widetilde{D}, \widetilde{G}$ and $\widetilde{H}$ denote the approximations of D, G and H respectively obtained from the approximated values of $I_R, I_\Omega, |\Omega|$ and $|R|$ in (3.11) and (3.12).

4. IMPLEMENTATION

In this section we give some details about the computation of the energies previously introduced. Moreover, we describe the main features of the application SubdivisionSnake, which is able to compute the subdivision snakes produced by cubic B-spline and 4-point subdivision curves.

4.1. Details about the energies

For the implementation of energies it is necessary to define how to compute the gradient of an image in a point, the area enclosed by a curve, and others details. In the following we discuss these themes.

4.1.1. Gradient energy

Since the image I is only defined in points with integer coordinates, the evaluation of I and its partial derivatives in a point $(x, y) \in \mathbb{R}^2$ is approximated using bilinear interpolation. In particular, ∇I in 3.5 is approximated as,

$$\begin{aligned}\nabla I(x,y) := \nabla I(\lfloor x \rfloor, \lfloor y \rfloor)\,(1-\{x\})(1-\{y\}) + \nabla I(\lfloor x+1 \rfloor, \lfloor y \rfloor)\,\{x\}(1-\{y\}) +& \\ \nabla I(\lfloor x \rfloor, \lfloor y+1 \rfloor)\,(1-\{x\})\{y\} + \nabla I(\lfloor x+1 \rfloor, \lfloor y+1 \rfloor)\{x\}\{y\},& \quad (4.1)\end{aligned}$$

where $\{x\} = x - \lfloor x \rfloor$ denotes the fractional part of x.
The gradient of the image in a pixel can be approximated using different filters such as Prewitt and Sobel [24] (see Figure 5). Since we evaluate the gradient in points that belong to the snake, it is convenient to extend the width of the filter in order to increase the region of attraction of gradient energy (see Figure 6). Consequently, we use a generalization of the Prewitt filter of $(2q + 1) \times (2q + 1)$ pixels, to compute the gradient in those pixels with distance greater or equal to $q > 0$ (see Figure 7) to the boundary of the image. For the rest of the pixels we use Sobel filter to approximate the gradient. The constant value q depends on the image dimensions.

-1	-1	-1
0	0	0
1	1	1

$\frac{\partial I}{\partial x}$

-1	0	1
-1	0	1
-1	0	1

$\frac{\partial I}{\partial y}$

(a) Prewit: for $\frac{\partial I}{\partial x}$ (left) and $\frac{\partial I}{\partial y}$ (right)

-1	-2	-1
0	0	0
1	2	1

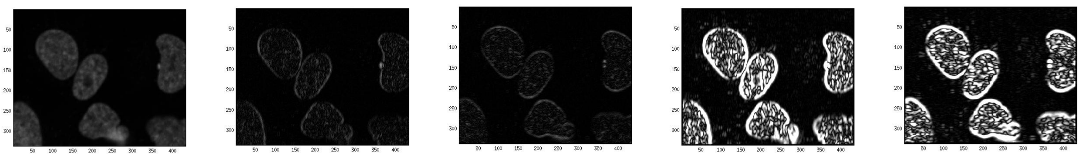
$\frac{\partial I}{\partial x}$

-1	0	1
-2	0	2
-1	0	1

$\frac{\partial I}{\partial y}$

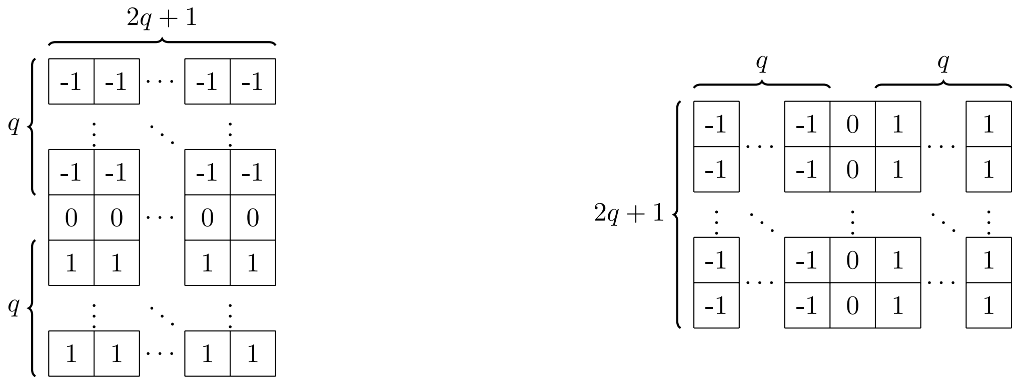
(b) Sobel: for $\frac{\partial I}{\partial x}$ (left) and $\frac{\partial I}{\partial y}$ (right)

Figure 5: Some known filters to compute approximations of the gradient in a pixel.



(a)Image I (b)Filter Sobel: $\frac{\partial I}{\partial x}$ (c)Filter Sobel: $\frac{\partial I}{\partial y}$ (d)Filter 7×7: $\frac{\partial I}{\partial x}$ (e)Filter 7×7: $\frac{\partial I}{\partial y}$

Figure 6: Partial derivatives computed using filters of different sizes. a) Original images, b) and c) Sobel's filter, d) and e) the proposed 7×7 filter.



$$\begin{array}{c|cc c cc|}
 & \multicolumn{5}{c}{\overbrace{\hphantom{-1\ -1\ \cdots\ -1\ -1}}^{2q+1}} \\
 & -1 & -1 & \cdots & -1 & -1 \\
q & & \vdots & \ddots & \vdots & \\
 & -1 & -1 & & -1 & -1 \\
 & 0 & 0 & \cdots & 0 & 0 \\
 & 1 & 1 & & 1 & 1 \\
q & & \vdots & \ddots & \vdots & \\
 & 1 & 1 & \cdots & 1 & 1
\end{array}
\qquad
\begin{array}{c|ccc c ccc|}
 & \multicolumn{3}{c}{\overbrace{\hphantom{-1\ \cdots\ -1}}^{q}} & & \multicolumn{3}{c}{\overbrace{\hphantom{1\ \cdots\ 1}}^{q}} \\
 & -1 & \cdots & -1 & 0 & 1 & \cdots & 1 \\
 & -1 & & -1 & 0 & 1 & & 1 \\
2q+1 & \vdots & \ddots & & \vdots & & \ddots & \vdots \\
 & -1 & \cdots & -1 & 0 & 1 & \cdots & 1 \\
 & -1 & & -1 & 0 & 1 & & 1
\end{array}$$

Figure 7: Proposed filter to compute approximations of the gradient of an image: for $\frac{\partial I}{\partial x}$ (left) and $\frac{\partial I}{\partial y}$ (right).

The gradient of the image in each pixel is pre-computed and stored in a lookup table, so that the evaluations in (3.5), (3.15) and (3.17) use the stored values.

4.1.2. Region energy

The first step to compute the integrals (3.6) defining the region energy is to obtain a sequence of pixels that approximates the snake, that is represented by the polygon with vertices $\{\mathbf{r}(i/2^k) = (x_i^k, y_i^k), i = 0, \ldots, 2^k M - 1\}$. The problem is reduced to the rasterization of each edge of that polygon. Rasterization algorithms provide the pixels that are intersected by a straight line (see Figure 8.1). Since these are more pixels than the ones needed to describe the region Ω enclosed by a closed polygon, we select for each horizontal line only one pixel per edge of the polygon [2]. To obtain these pixels, called *boundary pixels*, we determine for the horizontal line j the pixels (j, l_j^i) that are simultaneously on the line and on the edge $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$. If the result of the previous operation is more than one pixel, then we select the outer pixel with respect to the region enclosed by the subdivision curve (see Figure 8.2). Consequently, we proceed as follows.

We classify the edge $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$ as *downhill, horizontal* or *uphill* if the sign of $x_i^k - x_{i+1}^k$ is negative, zero or positive, respectively [3]. To compute approximately the integrals in (3.6) it is necessary to chose, for a given edge $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$, one pixel with coordinates (j, l_j^i) for each image row j, with $\min\{\lceil x_i^k \rceil, \lceil x_{i+1}^k \rceil\} \leq j \leq \max\{\lceil x_i^k \rceil, \lceil x_{i+1}^k \rceil\}$. The value of l_j^i depends on the previous edge classification as follows. Let $r_i(x)$ be the equation of the line passing through the pixels $(\lceil x_i^k \rceil, \lceil y_i^k \rceil)$ and $(\lceil x_{i+1}^k \rceil, \lceil y_{i+1}^k \rceil)$, then,

$$r_i(x) = \lceil y_i^k \rceil + \frac{\lceil y_{i+1}^k \rceil - \lceil y_i^k \rceil}{\lceil x_{i+1}^k \rceil - \lceil x_i^k \rceil}(x - \lceil x_i^k \rceil).$$

If $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$ is a downhill edge (see Figure 8.2, edge $[\mathbf{r}(4), \mathbf{r}(5)]$) then,

$$l_j^i = \min\left\{\lceil r_i(j) \rceil, \lceil r_i(j+1) \rceil\right\}, \qquad j \in \left[\lceil x_i^k \rceil, \lceil x_{i+1}^k \rceil\right] \tag{4.2}$$

[2] We choose the horizontal direction without loss of generality, the same result is obtained if the vertical direction is chosen.

[3] Remember that we are using the system of coordinates defined by $(row, column)$.

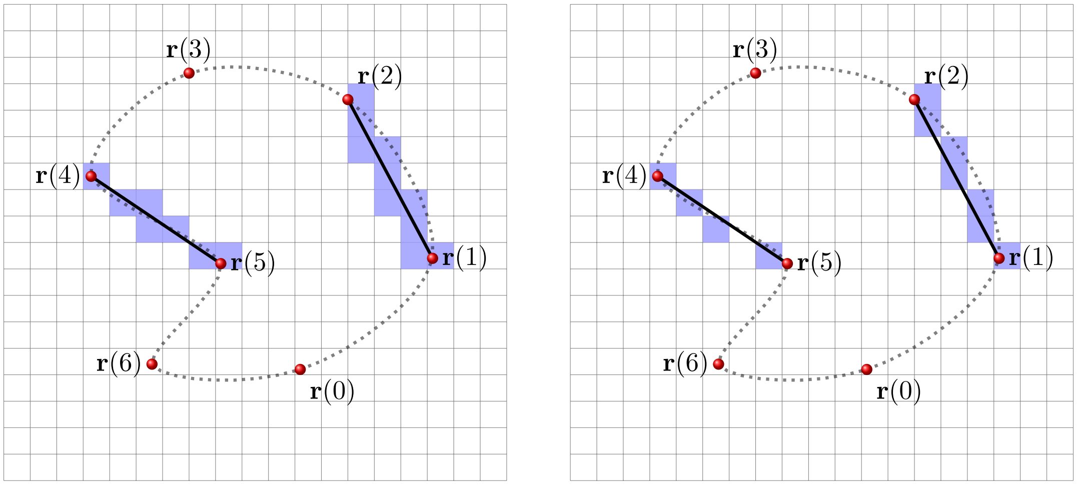


Figure 8.1: Rasterization of straight lines.

Figure 8.2: Boundary pixels to describe the boundary.

Figure 8: Pixel discretization of a straight line for a left edge and a right edge describing the boundary of a region.

If $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$ is a uphill edge (see Figure 8.2, edge $[\mathbf{r}\,(1)\,, \mathbf{r}\,(2)]$) then,

$$l^i_j = \max\left\{\lceil r_i(j)\rceil, \lceil r_i(j+1)\rceil\right\}, \qquad j \in \left[\left\lceil x^k_{i+1}\right\rceil, \left\lceil x^k_i\right\rceil\right] \tag{4.3}$$

Finally, if $[\mathbf{r}\left(i/2^k\right), \mathbf{r}\left((i+1)/2^k\right)]$ is a horizontal edge, then there is no need to define the value of l^i_j as $\operatorname{sign}(x^k_i - x^k_{i+1}) = 0$ in (3.11) and (3.12). In this case, the description of the boundary makes use of the neighboring edges.

In order to describe the boundary of the region enclosed by the subdivision curve, we store pairs of *boundary pixels* with respect to each horizontal. The amount of pairs on each horizontal line depends on the convexity of the curve (see Figure 9). It should be noticed that a boundary pixel may be simultaneously the right pixel of a pair and left pixel of the next pair in the same horizontal line, see for example the pixel corresponding to $\mathbf{r}(7)$ in Figure 9.

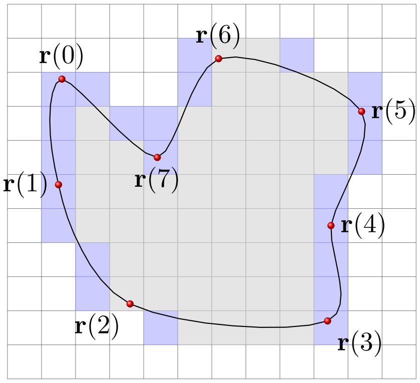
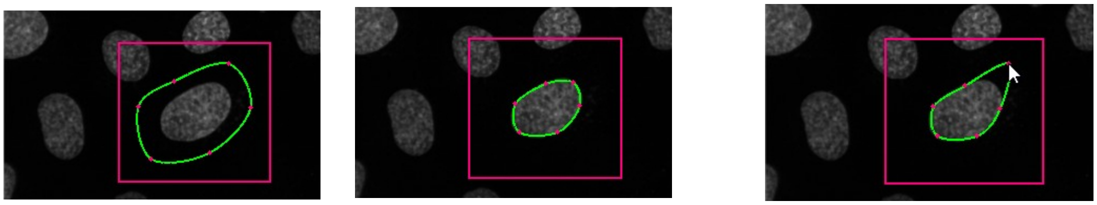


Figure 9: Description of the boundary of a region with pairs of *boundary pixels*.

4.2. SubdivisionSnake Application

The segmentation technique proposed in this paper has been implemented in C# with `.NET` platform (version 4.5). The optimization step, based on Limited memory BFGS method [8], is done using the library `Optimization.dll` of `Accord.NET` [2]. The resulting application is called SubdivisionSnake and it is independent of any imaging hardware.

Currently, SubdivisionSnake application is able to compute two types of subdivision snakes: cubic B-spline (2.17)-(2.18) and four points subdivision curves (2.12)-(2.13). Other subdivision curves could be easily added to the application if a procedure for evaluating the curve and its partial derivatives at dyadic parametric values is included. The interaction with the user is very simple and only requires an initial polygon approximating the boundary of the object to be segmented and a bounding box containing the object to be segmented and the initial polygon. As illustrated in Figure 10, the position of any control point can be intuitively manipulated on the image with simple mouse actions. The snake is updated in real-time since control points have local influence and therefore only a small region of the snake has to be recomputed. The resulting tool is a semi-automatic and intuitive segmentation algorithm based on the position of the control points and consisting of three fundamental steps: initialization, optimization and correction.



Initialization.

Segmentation

Interaction with the control points

Figure 10: Interaction of the user with SubdivisionSnake application.

SubdivisionSnake application has a main window to load the image and set the free parameters. This window contains the following options:

- Load and save image: this option load the target image and save the segmented image. The application accepts `.jpg`, `.gif` , `.bmp`, `.png` and `.tif` images. Color images are transformed in gray images at the beginning of the processing.

- Target color object: using this option the user says if the object to be segmented is darker than the background (default option) or the contrary.

- Control points: the application offers several options related to the control points of the snake.

  Clicking on the image, the initial position of control points can be defined. It is also possible to delete a control point or to change its position (dragging the mouse). Finally, control points can be saved or loaded to be reused.

- α: this options allows the selection of a value in $[0, 1]$ for the parameter α.

By default, we use $\alpha = 0.1$ at the beginning of the optimization and $\alpha = 0.9$ in the last steps of the optimization since gradient based energies have a narrow zone of attraction in comparison with region based energies. This means that in the default option the region energy controls initially the movement of the snake inducing his fast displacement. When the position of the snake stabilizes the value of α changes automatically to $\alpha = 0.9$ and then the gradient energy pushes the snake to the boundary of the object.

Otherwise, the selected α does not change during all the optimization process.

- Visualize: this option is used to visualize the snake and its control polygon.

5. RESULTS

To illustrate the performance of SubdivisionSnake we experiment with two group of images. The first group is composed by synthetic images. These images are created in such a way that the boundary of the object to be segmented is clear e intuitive. Some synthetic images used in this paper were produced filling the interior of closed subdivision curves with a color that makes a good contrast with the background. Other synthetic images were obtained using `.seg` files of Berkeley data base [5]. From these files it is possible to know which pixels belong to each object to be segmented in the image. The syntectic image is obtained assigning a specific color to these pixels and a contrasting color to the rest of pixels. In general, synthetic images are simpler than the real images included in the second group. In our experiment we also use real images from Berkeley data base and from other sources. All the examples shown in this section are the direct result of the optimization process, without any previous denoising, smoothing or interactive correction. Color images are transformed to gray level images using the standard perceptual weightings for the three-color components [24].

5.1. Quantitative evaluation of results

When we work with synthetic images the ground-truth region, composed by pixels belonging to the object Γ, is known. In some real images of Berkeley database, the ground-truth is also given. In all these cases it is possible to validate quantitatively the quality of the results using the Jaccard distance J between Γ and the region Ω enclosed by the snake, given by

$$J = 1 - \frac{|\Omega \cap \Gamma|}{|\Omega \cup \Gamma|}$$

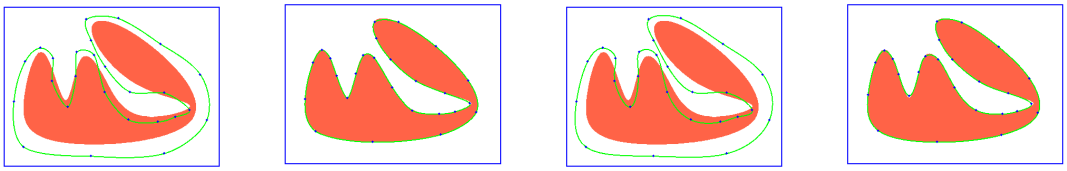
where $|G|$ denotes the area of region G. Observe that $0 \leq J \leq 1$ and a value of J close to 0 indicates a good segmentation.

Table 1 shows the segmentation results obtained for images in Figures 11, 12 and 13, with the default selection of the parameter α. We use the same sequence of control points $\mathbf{P}^0$ to initialize the snake based on cubic B-splines and on 4-point subdivision scheme. In the case of the B-splines, we compute initially the sequence of points $\widetilde{\mathbf{P}}^0$ (2.22) such that the corresponding cubic B-spline interpolates the points $\mathbf{P}^0$. Columns 2 and 3 of Table 1 contain the Jaccard distance between Γ and the region Ω enclosed by the cubic B-spline snake in the initialization step and after convergence, respectively.

Image	# control points	cubic B-spline		4-point	
		Initialization	Segmentation	Initialization	Segmentation
Synthetic image	26	0.4019	0.0168	0.3935	0.0196
Airplane	23	0.3137	0.0837	0.3099	0.0772
Japanese garden	8	0.5091	0.0471	0.5134	0.0463

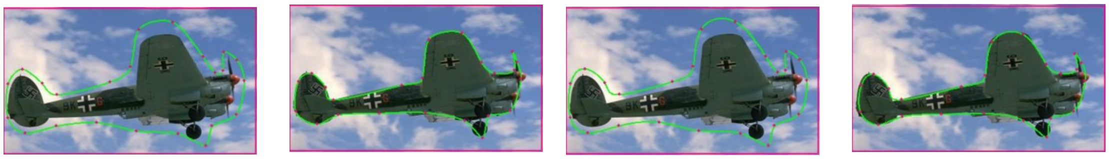
Table 1: Jaccard distance for images in Figures 11, 12 and 13.

Similarly, columns 4 and 5 contain the Jaccard distance for the 4 points subdivision snake. We observe that despite of the different nature of the images, the Jaccard distance in the optimum is very small for both subdivision snakes, in correspondence with a good segmentation of the target objet.



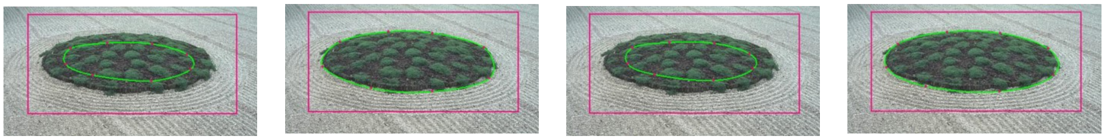
Initialization Segmentation Initialization Segmentation

Figure 11: Synthethic image corresponding to the results reported in Table 1. a) and b) 4 points snake, c) and d) cubic B-spline snake.



Initialization Segmentation Initialization Segmentation

Figure 12: Airplane image corresponding to the results reported in Table 1. a) and b) 4 points snake, c) and d) cubic B-spline snake.



Initialization Segmentation Initialization Segmentation

Figure 13: Japanese garden image corresponding to the results reported in Table 1. a) and b) 4 points snake, c) and d) cubic B-spline snake.

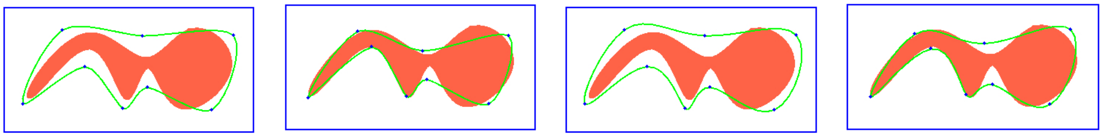
5.2. Influence of the number of control points

The number of control points has a strong influence in the quality of the segmentation. In general, increasing the number of control points improves the quality of segmentation, but makes higher the

# control points	cubic B-spline		4-point	
	Initialization	Segmentation	Initialization	Segmentation
8	0.3630	0.1653	0.3581	0.1873
10	0.3682	0.0637	0.3539	0.0788
12	0.3681	0.0388	0.3411	0.0589

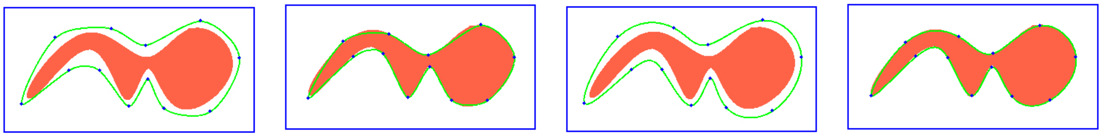
Table 2: Influence of the number of control points in the quality of the segmentation measured by the Jaccard distance.

computational cost. Table 2 shows that if one selects initial polygons with different number of control points but approximately the same Jaccard distance, then the better segmentation corresponds to the snake with the highest number of control points. This result is valid for both subdivision snakes. Figures 14 and 15 correspond to the results in Table 2 for 8 and 12 control points respectively. The improvement of the segmentation is evident when we compare Figure 14 b) with Figure 15 b) and Figure 14 d) with Figure 15 d).



Initialization Segmentation Initialization Segmentation

Figure 14: Images corresponding to the results in Table 2 with 8 control points. a) and b) 4 points snake, c) and d) cubic B-spline snake.



Initialization Segmentation Initialization Segmentation

Figure 15: Images corresponding to the results in Table 2 with 12 control points. a) and b) 4 points snake, c)and d) cubic B-spline snake.

5.3. Some results with real images

To show the potential of the subdivision snakes to segment objects in real images, we include in this section the results corresponding to three images: a cerebral hemorrhage (Figure 16), a cell (Figure 17) and a hail (Figure 18). In general, processing real images is more involved than synthetic images, since several factors may affect the segmentation procedure. In some cases the boundary of the object of interest may be blurred, see for instance Figures 16 and 17). In other cases, the object to be segmented has inhomogeneous intensity values and poor contrast with the background, see Figure 17. Moreover, sometimes a non-uniform illumination makes difficult to capture the boundary of the object, see Figure 18. Despite these difficulties, our method provides reasonable segmentation results, even without image enhancing in the preprocessing.

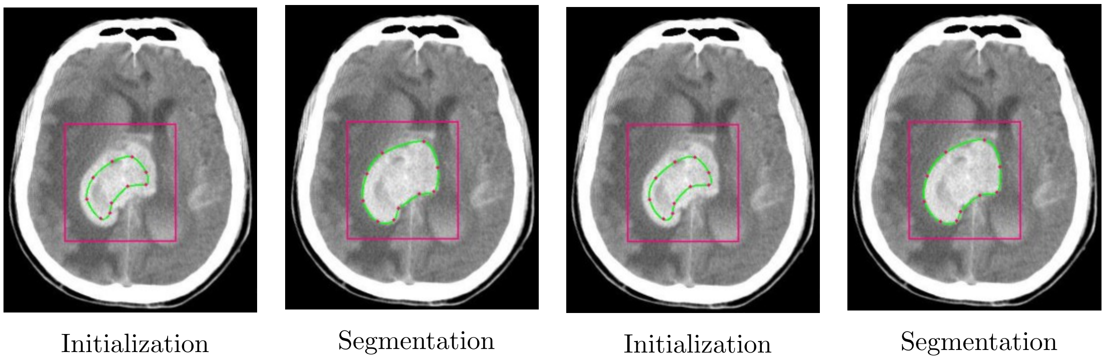

Initialization Segmentation Initialization Segmentation

Figure 16: Segmentation of a cerebral hemorrhage.a) and b) 4 points snake, c) and d) cubic B-spline snake.

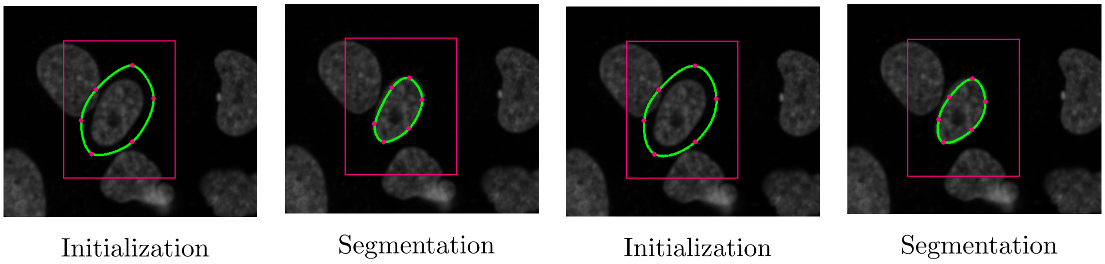

Initialization Segmentation Initialization Segmentation

Figure 17: Segmentation of a cell in a sample. a) and b) 4 points snake, c) and d) cubic B-spline snake.

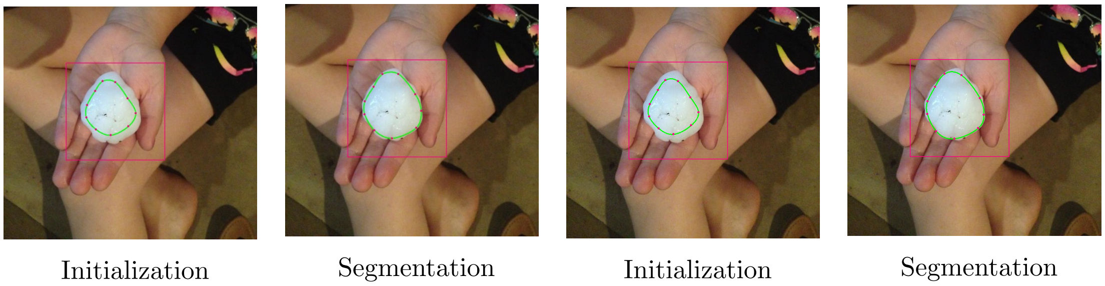

Initialization Segmentation Initialization Segmentation

Figure 18: Segmentation of a hail.a) and b) 4 points snake, c) and d) cubic B-spline snake.

6. CONCLUSIONS

We have proposed a method for computing the contour of an object in an image using a snake represented as a subdivision curve. Our method is based on linear and stationary subdivision schemes. We illustrate its performance developing the computational tool SubdivisionSnake, that computes the snakes associated with two classical subdivision schemes: the four points scheme and the cubic

B-spline. The method profits from the hierarchical nature of subdivision curves, since the unknowns of the optimization process are the few control points of the subdivision curve in the coarse representation, while good approximations of the energies and their derivatives are obtained from the fine representation.

The evolution of the snake is driven by its control points that are computed minimizing an energy which is combination of contour-based and region-based energies. We have introduced a new region energy that guides the snake, maximizing the contrast between the average intensity of the image within the snake and the average intensity over the complement of the snake in a fixed bounding box. Explicit expressions of the new region energy functional and its partial derivatives have been provided and an accurate pixel discretization was also discussed.

Our experiments using synthetic and real images confirm that the proposed method is fast and successful. Our flexible computational framework SubdivisionSnake facilitates the interaction with the snake by letting the user to move directly the control points with the mouse and and to control the weights associated to the combination of both energy functionals.

The proposed methodology and implementation of our method, which is illustrated in the case of two popular linear and stationary subdivision schemes, may be extended to non-linear and non-stationary subdivision schemes, in particular to those subdivision schemes with known formulae for the position of the limit points and their tangent vectors, see for instance [30], [31]. We could also implement a multiscale optimization strategy [3], where the snake is optimized in a coarse-to-fine fashion.

An improved version of the computational tool SubdivisionSnake including some non-linear and non-stationary subdivision schemes and a multiscale optimization strategy will be developed and the link of the corresponding open source code will be made public.

RECEIVED: JULY, 2019.
REVISED: SEPTEMBER, 2020.

REFERENCES

[1] APPELTON, B. and TALBOT, H. [2005]: ”Globally optimal geodesic active contours”, **J. Math. Imaging Vis.**, 23(1), 67–86.

[2] **Accord.NET Machine Learning Framework** [2008 - 2015], version 3.0. http://accord-framework.net.

[3] BADOUAL, A., SCHMITTER, D., UHLMANN, V., UNSER, M. [2017]: ”Multiresolution Subdivision Snakes”, **IEEE Transactions on Image Processing** 26 (3), 1188-1201.

[4] BADOUAL, A., UNSER, M. and DEPEURSINGE, A. [2019]: ”Texture-driven parametric snakes for semi-automatic image segmentation ”, **Computer Vision and Image Understanding** 188, 102793.

[5] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[6] BRIGGER, P., HOEG, J. and UNSER, M. [2000]: ”B-spline snakes: A flexible tool for parametric contour detection”, **IEEE Trans. Image Processing** 9, 1484–1496.

[7] BRITTO, A. P. and RAVINDRAN, G. [2007]: ”Review of deformable curves-A retro analysis”, **Inf. Technol. J.**, 6 (1), 26–36.

[8] BYRD, R., LU, P., NOCEDAL, J. [1995]: ”A limited memory algorithm for bound constrained optimization”, **SIAM Journal on Scientific and Statistical Computing**, 16(5),1190–1208.

[9] CASELLES, V., KIMMEL, R. and SAPIRO, G. [1997]: ”Geodesic active contours”, **Int. J. Comput. Vision**, 22(1), 61–79.

[10] CHAN, T. F., VESE, L. A. [2001]: ”Active contours without edges”, **IEEE Trans. Image Processing**, 10, 266–276.

[11] CHESNAUD, C., REFREGIER, P., BOULET, V. [1999]: ”Statistical region snake-based segmentation adapted to different physical noise models”, **IEEE Trans. Pattern Anal. Machine Intell.**, 21, 1145–1157.

[12] COHEN, L. D. [1991]: ”On active contour models and balloons”, **CVGIP: Imag. Understand.**, 53, 211–218.

[13] CONTI, C. , ROMANI, L. and UNSER M. [2015]: ”Ellipse-preserving Hermite interpolation and subdivision”, **J. Math. Anal. Appl**. 426, 211-227.

[14] CONTI, C., COTRONEI, M. and ROMANI L. [2017]: ”Beyond B-splines: exponential pseudo-splines and subdivision schemes reproducing exponential polynomials ”, **Dolomites Research Notes on Approximation**, Vol. 10, 31-42 .

[15] DELGADO, R., THEVENAZ, P., SEELEMANTULA, C., UNSER, M. [2012]: ”Snakes with an ellipse-reproducing property”, **IEEE Transactions on Image Processing** 21, 1258–1271.

[16] DELGADO, R., THEVENAZ, P., UNSER, M. [2012]: ”Exponential splines and minimal-support bases for curve representation”, **Comput. Aided Geometric Design** 29, 109–128.

[17] DELGADO, R., UNSER, M. [2013]: ”Spline-based framework for interactive segmentation in biomedical imaging”, **IRBM**, 34 (3), 235–243.

[18] DELGADO, R., UHLMANN, V., SCHMITTER, D., UNSER, M. [2015]: ”Snakes on a Plane: A perfect snap for bioimage analysis”. **IEEE Signal Process. Mag.** 32 (1), 41–48.

[19] DYN, N., LEVIN, D., GREGORY, J. A. [1987]: ”A 4-point interpolatory subdivision scheme for curve design”, **Computer Aided Geometric Design** 4, 257–268.

[20] DYN, N., LEVIN, D. [2002]: ”Subdivision schemes in geometric modelling”, **Acta Numerica** 11, 73–144.

[21] DYN, N. [2009]: ”Linear and Nonlinear Subdivision Schemes in Geometric Modeling”, in Foundations of computational mathematics”, Hong Kong 2008, **London Math. Soc. Lecture Notes** Cambridge: Cambridge Univ. Press, 363, 68-92.

[22] FIGUEIREDO, M. A., LEITÃO, J. M. and JAIN, A. K. [2000]: ”Unsupervised contour representation and estimation using B-splines and a minimum description length criterion”, **IEEE Trans. Image Processing** 9(6), 1075–1087.

[23] FRAZIER, M. W. [1999]: ”An Introduction to Wavelets Through Linear Algebra”, **Springer Verlag**.

[24] GONZALEZ, R. C.,WOODS, R. E., EDDINS, S. L. [2009]: ”Digital Image Processing using MATLAB”, **Gatesmark**, 2nd ed.

[25] HUG, J., BRECHBÜHLER, C., SZÈKELY, G. [1999]: ”Tamed Snake: A Particle System for Robust Semi-Automatic Segmentation”, MICCAI ’99 Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention, **Springer-Verlag London, UK**, 106-115.

[26] JACOB, M., BLU, T., UNSER, M. [2001]: ”A unifying approach and interface for spline-based snakes”, in **Proc. SPIE Int. Symp. Medical Imaging: Image Processing**, 4322, 340–347.

[27] JACOB, M., BLU, T., UNSER, M. [2004]: ”Efficient energies and algorithms for parametric snakes”, **IEEE Transactions on Image Processing** 13, 1231–1244.

[28] KASS, M., WITKIN, A., TERZOPOULOS, D. [1987]: ”Snakes: Active contour models”, **Int. J. Comput. Vis.**, 1, 321–332.

[29] MCINERNEY, T. [2008]: ”SketchSnakes: Sketch-Line initialized snakes for efficient interactive medical image segmentation”, **Computerized medical imaging and graphics** 32 (5), 331–352.

[30] ROMANI, L., HERNANDEZ, V. and ESTRADA, J. [2016]: ”Exact evaluation of a class of non-stationary approximating subdivision algorithms and related application ”, **IMA J Numerical Analysis**. Vol. 36, Issue 1,380-399.

[31] ROMANI L., BADOUAL, A. and UNSER, M. [2019]: ”Normal-based interpolating subdivision for the geometric representation of deformable models”, **IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)**, 8–11.

[32] SHI, Z., LI, F. [2011]: ”Active subdivision snake scheme”, **Applied Mechanics and Materials**, 44-47, 3917–3921.

[33] STAIB, L. H., DUNCAN, J. S. [1992]: ”Boundary fitting with parametrically deformable models”, **IEEE Trans. Pattern Anal. Machine Intell.** 14, 1061–1075.

[34] STOLLNITZ, E. J., DEROSE, A. D., SALESIN, D. H. [1996]: ”Wavelets for Computer Graphics: Theory and Applications”, **Morgan Kaufmann**.

[35] THEVENAZ, P., DELGADO, R., UNSER, M. [2011]: ”The ovuscule”, **IEEE Transactions on Pattern Analysis and Machine Intelligence** 33, 382–393.

[36] UHLMANN, V., FAGEOT, J. and UNSER, M. [2016]: " Hermite Snakes With Control of Tangents", **IEEE Transactions on Image Proccessing**, Vol. 25, No. 6, 2803–2816.

[37] XU, C., PRINCE, J. (1998) "Snakes, shapes, and gradient vector flow", **IEEE Trans. Image Processing** 7,359–369.

[38] ZHANG, H., BIAN, Z., GUO, Y., FEI, B. and YE, M. (2003) "An efficient multiscale approach to level set evolution", **Proc. 25th Annu. Int. Conf. IEEE Engineering in Medicine and Biological Society**, 694–697.

A MAKING INTERPOLATORY THE CUBIC B-SPLINE SUBDIVISION SCHEME

The rule in (2.19)) not only allows to evaluate the curve on dyadic parameter values, but also to impose the interpolatory condition to the scheme (see Fig. 3). In fact, starting from $\mathbf{P}^0$ a polygon $\widetilde{\mathbf{P}}^0$ can be computed such that if,

$$\mathbf{r}(t) = \sum_{j\in\mathbb{Z}} \widetilde{\mathbf{p}}_j^0 \varphi\left(t-j\right), \tag{A1}$$

then $\mathbf{r}(i) = \mathbf{p}_i^0$.

Let $\mathbf{P}^0 = \left\{\mathbf{p}_i^0 \in \mathbb{R}, \quad \mathbf{p}_{i+M}^0 = \mathbf{p}_i^0\right\}$ be an M-periodic sequence of points. From (A1) and (2.19) it is clear that,

$$\mathbf{p}_i^0 = \mathbf{r}(i) = \frac{1}{6}\widetilde{\mathbf{p}}_{i-1}^0 + \frac{4}{6}\widetilde{\mathbf{p}}_i^0 + \frac{1}{6}\widetilde{\mathbf{p}}_{i+1}^0, \qquad \text{for} \quad i \in \mathbb{Z}. \tag{A2}$$

This expression can be written in matrix terms as[4]

$$\mathbf{P}^0 = S^\infty \widetilde{\mathbf{P}}^0 \tag{A3}$$

where S^∞ is the circulant matrix with first column b given by,

$$b = \begin{bmatrix} S^\infty{}_{0,0} & S^\infty{}_{1,0} & \dots & S^\infty{}_{M-1,0} \end{bmatrix}^T = \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & 0 & \dots & 0 & \frac{1}{6} \end{bmatrix}^T. \tag{A4}$$

Hence from (A3) we obtain

$$\widetilde{\mathbf{P}} = (S^\infty)^{-1}\mathbf{P}^0$$

and the problem is reduced to compute the inverse of the matrix S^∞.

From Fourier Analysis [23] it is known that since S^∞ is a circulant matrix, it is diagonalized by the Fourier basis. More precisely,

$$S^\infty = \left(F^M\right)^{-1} D F^M, \tag{A5}$$

where D is a diagonal matrix and F^M is the matrix with elements,

$$F_{s,k}^M = e^{-2sk\pi \mathrm{i}/M}, \qquad s,k = 0,\dots,M-1. \tag{A6}$$

It can be easily verified that,

$$\left(F^M\right)^{-1} = \frac{1}{M}\overline{F^M}$$

[4] The symbol S^∞ comes from the limit of the subdivision operator S.

where $\overline{F^M}$ is the conjugated matrix of F^M. Hence, from (A5) it follows,

$$(S^\infty)^{-1} = \frac{1}{M}\overline{F^M}\ D^{-1}\ F^M. \tag{A7}$$

It is also known [23], that the diagonal matrix D has in its diagonal the values of the Fourier transform of b,

$$\widehat{b} = \frac{2}{3}F^M_{\cdot,0} + \frac{1}{6}F^M_{\cdot,1} + \frac{1}{6}F^M_{\cdot,M-1}, \tag{A8}$$

where $F^M_{\cdot,k}$, $k = 0, \ldots, M-1$ are the column vectors with elements defined in (A6). Thus,

$$\begin{aligned}\widehat{b}_s &= \frac{2}{3}F^M_{s,0} + \frac{1}{6}F^M_{s,1} + \frac{1}{6}F^M_{s,M-1}\\ &= \frac{2}{3} + \frac{1}{6}e^{-2s\pi\mathrm{i}/M} + \frac{1}{6}e^{-2(M-1)\pi\mathrm{i}/M}\\ &= \frac{2}{3} + \frac{1}{3}\cos(2s\pi/M) = \widehat{b}_{M-s}\end{aligned} \tag{A9}$$

Since the inverse of a non-singular circulant matrix is also a circulant matrix and as $\widehat{b}_s \neq 0$ for all s, the matrix representing $(S^\infty)^{-1}$ is also circulant. Therefore, we only need to compute its first column $(S^\infty)^{-1}_{\cdot,0}$. From (A7) and (A9) we obtain,

$$\begin{aligned}(S^\infty)^{-1}_{s,0} &= \frac{1}{M}\sum_{t=0}^{M-1}\widehat{b}_s^{-1}e^{2st\pi\mathrm{i}/M}\\ &= \begin{cases}\frac{1}{M} + \frac{3}{M}\cos(s\pi) + \frac{2}{M}\sum\limits_{t=1}^{\frac{M}{2}-1}\left(\frac{2}{3}+\frac{1}{3}\cos(2t\pi/M)\right)^{-1}\cos(2st\pi/M), & \text{for} \quad M \mod 2 = 0\\ \frac{1}{M} + \frac{2}{M}\sum\limits_{t=1}^{\lfloor\frac{M}{2}\rfloor}\left(\frac{2}{3}+\frac{1}{3}\cos(2t\pi/M)\right)^{-1}\cos(2st\pi/M), & \text{for} \quad M \mod 2 = 1\end{cases}\end{aligned} \tag{A10}$$

Remark A1. *To compute the entries for the whole matrix we just need to remember that it is circulant and use* (A10).

B COMPUTING THE GRADIENTS OF THE ENERGIES

With the aim to simplify the exposition, the deduction of the expressions for the gradient of the energies are showed here. Being a similar process to deduce both partial derivatives, we only show the deduction of the partial derivatives with respect to each x^0_j, $j = 0, \ldots, M-1$.

B1. Gradient of *gradient energy*

Deriving directly in (3.4) with respect to x^0_j we obtain,

$$\begin{aligned}\frac{\partial E_{grad}}{\partial x^0_j} = &-\int_0^M\left(\left(\frac{\partial^2 I}{\partial x^2}\frac{\partial x}{\partial x^0_j} + \frac{\partial^2 I}{\partial x\partial y}\frac{\partial y}{\partial x^0_j}\right)\frac{\mathrm{d}y(t)}{\mathrm{d}t} + \frac{\partial I}{\partial x}\frac{\partial\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)}{\partial x^0_j}\right)\mathrm{d}t\\ &+\int_0^M\left(\left(\frac{\partial^2 I}{\partial x\partial y}\frac{\partial x}{\partial x^0_j} + \frac{\partial^2 I}{\partial y^2}\frac{\partial y}{\partial x^0_j}\right)\frac{\mathrm{d}x(t)}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\partial\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)}{\partial x^0_j}\right)\mathrm{d}t.\end{aligned} \tag{B1}$$

Taking into account that $y(t)$ and $\frac{\mathrm{d}y(t)}{\mathrm{d}t}$ don't depend on x_j^0, from (B1) we get,

$$\frac{\partial E_{grad}}{\partial x_j^0} = \int_0^M \left(\left[\frac{\partial^2 I}{\partial x \partial y} \frac{\partial x}{\partial t} - \frac{\partial^2 I}{\partial x^2} \frac{\mathrm{d}y}{\mathrm{d}t} \right] \frac{\partial x}{\partial x_j^0} + \frac{\partial I}{\partial y} \frac{\partial \left(\frac{\mathrm{d}x}{\mathrm{d}t} \right)}{\partial x_j^0} \right) \mathrm{d}t \tag{B2}$$

From (2.5) and (2.6) it follows,

$$\frac{\partial x(t)}{\partial x_j^0} = \varphi(t-j), \quad \frac{\partial y(t)}{\partial y_j^0} = \varphi(t-j) \tag{B3}$$

$$\frac{dx(t)}{dt} = \sum_{j=-1}^{M+1} x_j^0 \varphi^{'}(t-j), \quad \frac{\mathrm{d}y(t)}{\mathrm{d}t} = \sum_{j=-1}^{M+1} y_j^0 \varphi^{'}(t-j) \tag{B4}$$

where $\varphi^{'}$ denotes $\frac{d\varphi}{dt}$. Evaluating the last expressions in $t = \frac{i}{2^k}, \;\; i = 0, \ldots, 2^k M - 1$ it holds

$$\left. \frac{\partial x(t)}{\partial x_j^0} \right|_{t=\frac{i}{2^k}} = \varphi\left(\frac{i}{2^k} - j \right) = \left. \frac{\partial y(t)}{\partial y_j^0} \right|_{t=\frac{i}{2^k}} \tag{B5}$$

$$\left. \frac{\partial \left(\frac{dx}{dt} \right)}{\partial x_j^0} \right|_{t=\frac{i}{2^k}} = \varphi^{'}\left(\frac{i}{2^k} - j \right) = \left. \frac{\partial \left(\frac{dy}{dt} \right)}{\partial y_j^0} \right|_{t=\frac{i}{2^k}} \tag{B6}$$

Substituting the integral in (B2) by the average of the integrand evaluated in the parameter values $i/2^k, \;\; i = 0, \ldots, 2^k M - 1$ and using (B5)and (B6) we obtain the following approximation for the partial derivative of gradient energy with respect to x_j^0,

$$\frac{\partial E_{grad}}{\partial x_j^0} \approx \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \left(\frac{\partial^2 I}{\partial x \partial y} \left(\mathbf{r}\left(\frac{i}{2^k} \right) \right) t_{ix}^k - \frac{\partial^2 I}{\partial x^2} \left(\mathbf{r}\left(\frac{i}{2^k} \right) \right) t_{iy}^k \right) \varphi\left(\frac{i}{2^k} - j \right) + \frac{1}{2^k M} \sum_{i=0}^{2^k M-1} \frac{\partial I}{\partial y} \left(\mathbf{r}\left(\frac{i}{2^k} \right) \right) \varphi^{'}\left(\frac{i}{2^k} - j \right). \tag{B7}$$

In a similar way, deriving (3.4) with respect to y_j^0 and taking into account that $x(t)$ and $\frac{dx(t)}{dt}$ don't depend on x_j^0, we obtain the expression for $\frac{\partial E_{grad}}{\partial y_j^0}$.

B2. Gradient of *region energy*

Recalling that,

$$\frac{\partial E_{reg}}{\partial x_j^0} = -2D \left(\frac{\partial A}{\partial x_j^0} - \frac{\partial B}{\partial x_j^0} \right) \tag{B8}$$

where

$$A := \frac{I_\Omega}{|\Omega|} \tag{B9}$$

$$B := \frac{I_R - I_\Omega}{|R| - |\Omega|} \tag{B10}$$

$$D := A - B. \tag{B11}$$

Deriving directly in (B9) and (B10) we obtain,

$$\frac{\partial A}{\partial x_j^0} = \frac{1}{|\Omega|}\frac{\partial I_\Omega}{\partial x_j^0} - \frac{I_\Omega}{|\Omega|^2}\frac{\partial |\Omega|}{\partial x_j^0} \tag{B12}$$

$$\frac{\partial B}{\partial x_j^0} = \frac{-1}{|R|-|\Omega|}\frac{\partial I_\Omega}{\partial x_j^0} + \frac{I_R - I_\Omega}{(|R|-||\Omega|)^2}\frac{\partial |\Omega|}{\partial x_j^0}. \tag{B13}$$

Substituting (B12) and (B13) in (B8) we get,

$$\frac{\partial E_{reg}}{\partial x_j^0} = -2D\left[\left(\frac{1}{|\Omega|} + \frac{1}{|R|-|\Omega|}\right)\frac{\partial I_\Omega}{\partial x_j^0} - \left(\frac{I_\Omega}{|\Omega|^2} + \frac{I_R - I_\Omega}{\left(|R|-|\Omega|\right)^2}\right)\frac{\partial |\Omega|}{\partial x_j^0}\right]. \tag{B14}$$

Now we compute the partial derivatives involved in (B14) using the Green Theorem as stated in (3.8) and (3.9).

Deriving in the first equality of (3.8) it follows,

$$\frac{\partial I_\Omega}{\partial x_j^0} = -\int_0^M \frac{\partial I_1}{\partial x}\frac{\partial x(t)}{\partial x_j^0} y'(t)\mathrm{d}t.$$

Taking into account that, according to Leibniz's rule in (3.9) (for differentiation under the integral sign), $\frac{\partial I_1}{\partial x} = I(x(t), y(t))$ and $\frac{\partial x(t)}{\partial x_j^0} = \varphi(t-j)$, from the previous expression we obtain,

$$\frac{\partial I_\Omega}{\partial x_j^0} = -\int_0^M I(\mathbf{r}(t))\varphi(t-j)y'(t)\, dt. \tag{B15}$$

Since $|\Omega| = \int\int_\Omega \mathrm{d}x\mathrm{d}y$ from (B15) it is clear that,

$$\frac{\partial |\Omega|}{\partial x_j^0} = -\int_0^M \varphi(t-j)y'(t)\mathrm{d}t. \tag{B16}$$

Finally, substituting (B15) and (B16) in (B14) and grouping similar terms we obtain

$$\frac{\partial E_{reg}}{\partial x_j^0} = -2D\int_0^M \left[G - H\, I(\mathbf{r}(t))\right]\varphi(t-j)y'(t)\mathrm{d}t$$

where

$$G := \frac{I_\Omega}{|\Omega|^2} + \frac{I_R - I_\Omega}{(|R|-|\Omega|)^2} \quad \text{and} \quad H := \frac{1}{|\Omega|} + \frac{1}{|R|-|\Omega|}.$$

We proceed in a similar way to compute $\dfrac{\partial E_{reg}}{\partial y_j^0}$.