



## Segmentation with Active Contours

Fabien Pierre, Mathieu Amendola, Clémence Bigeard, Timothé Ruel,  
Pierre-Frédéric Villard

### ► To cite this version:

Fabien Pierre, Mathieu Amendola, Clémence Bigeard, Timothé Ruel, Pierre-Frédéric Villard.  
Segmentation with Active Contours. Image Processing On Line, 2021, 11, pp.120 - 141.  
10.5201/ipol.2021.298 . hal-03235096

**HAL Id: hal-03235096**

**<https://hal.science/hal-03235096>**

Submitted on 25 May 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Published in Image Processing On Line on 2021-05-24.  
 Submitted on 2020-04-03, accepted on 2021-03-05.  
 ISSN 2105-1232 © 2021 IPOL & the authors CC-BY-NC-SA  
 This article is available online with supplementary materials,  
 software, datasets and online demo at  
<https://doi.org/10.5201/ipol.2021.298>

# Segmentation with Active Contours

Fabien Pierre, Mathieu Amendola, Clémence Bigeard, Timothé Ruel,  
 Pierre-Frédéric Villard

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France  
[{fabien.pierre,pierre-frederic.villard}@univ-lorraine.fr](mailto:{fabien.pierre,pierre-frederic.villard}@univ-lorraine.fr)

## Abstract

Active contours (also known as *snakes*) have shown their ability to introduce regularity on image segmentation. In contrast with level-set approaches, the active contours techniques based on a contour parameterization are able to maintain the initial topology of the area of interest. For this reason, it has been used in recent medical research for diaphragm segmentation. Most of the on-line codes for 2D/3D segmentation, as well as built-in Matlab toolboxes are based on level-set methods. Moreover, in the literature, the implementation details of active contours methods with meshes in three dimensions are tight, making tedious any reproduction of these techniques. In this paper, we give some details of the implementation of active contours in 2D/3D with meshes, especially about the choice of the use of a 2D/3D mesh and its refinement. We also explore the choice of the parameters with a quantitative study of their influence on the segmentation results. The 3D segmentation method has been applied to CT scan images of the lungs.

## Source Code

The reviewed Matlab source code and documentation for this algorithm are available at [the web page of this article](#)<sup>1</sup>. See the `README.txt` file for usage instructions in the archive.

**Keywords:** active contours; image segmentation; medical imaging

## 1 Introduction

Medical training is more and more done using computer-based simulators. They offer a wide variety of training procedures without being invasive to any real patient and provide useful metrics to evaluate trainee's performance. Companies like Mentice<sup>2</sup> or Simbionix<sup>3</sup> are well known actors on the market. However those commercial simulators use generic geometries for the anatomy, i.e. based on conceptual organ geometries not based on real patients, so they cannot be used for treatment planning. Recent

<sup>1</sup><https://doi.org/10.5201/ipol.2021.298>

<sup>2</sup><http://www.mentice.com> [last access: May 24, 2021]

<sup>3</sup><http://www.simbionix.com> [last access: May 24, 2021]

research work is precisely focusing on filling this gap by building patient-based simulators [21, 9, 12]. When other segmentation techniques fail and when deep-learning techniques lack of labeled data, segmenting 3D organs with low contrast is still an open problem. Various approaches exist and state-of-the-art techniques are atlas-based models [10] and machine-learning based methods [20]. They rely on a consequent database that is not always available. In the following we will only focus on techniques efficient on a small amount of data.

Considering that the image is defined on a regular grid of pixels in 2D (resp. voxels in 3D), the aim of the image segmentation is the computation of  $N$  sets of pixels (resp. voxels) such that each set corresponds to an area of the image (these sets are called classes). These areas are computed considering contours, shapes, or statistics of their pixels (resp. voxels).

In the literature, two types of image segmentation methods can be distinguished, based on the discretization of the data. The first one is based on the natural regular grid definition of the image, in pixels or in voxels. The second one is based on the discretization of the boundary of the pixels sets. The first type has been widely used in the seminal segmentation approaches, especially with thresholding methods (see, e.g., [18] for a review of such techniques). One of the most known methods is Otsu's method, based on histograms [15]. Avoiding the dependency of the pixel discretization, the freely deformable models are based on a discretization of the boundary of the pixel classes. Kass et al. [8] have introduced such models for image segmentation in a method called *snakes*, based on the energy minimization. The minimizer provides a trade-off between a regular shape and a data-driven one. The concept has been extended in 3 dimensions by Terzopoulos et al. [19].

The level-set approaches have been introduced by Osher and Sethian [14] and popularized for image analysis and computer vision by Malladi et al. [11]. In these methods, the boundary of the shape of interest is defined as the level-set of a function, which is considered as the minimizer of an energy. For these approaches, two variants exist, which are variational approaches (see, e.g., [4]) and a PDE approach [14]. These methods favor a regular level-set and are able to naturally tackle the issue of topology variations, whereas the free deformable models preserve the initial topology of the shape. While the preservation of the topology can be seen as an advantage, especially when the interest object is known (e.g., the lungs), the free-deformable approach has been extended to manage some changes of topology [13]. In this paper, we will focus on the free deformable model with fixed topology.

In the following we will focus on the 2D formalization in Section 2, and the numerical computation of a solution in Section 3. It will be then extended to the 3D case in Section 5. The numerical results as well as the influence of the parameters will be studied in Section 4 (2D) and in Section 6 (3D).

## 2 Model for Active Contours in 2D

Active contour models aim to extract the boundary of an area contained in an image. To compute it, the main idea consists in the definition of a curve as the minimizer of a functional. This functional is composed of a sum of energies to perform a trade-off between the regularity of the shape and the contours defined by the gradient of the image. In our framework, the source code of the 2D implementation of active contours is in the `release2D` folder. It is composed of the main Matlab script `MAIN.EvolutionSnake2D.m` and a `functions` folder, which contains all the utilities functions.

## 2.1 Model Definition

In the following,  $I$  denotes an image on a domain  $\Omega \subset \mathbb{R}^2$ . The active contour model is defined as a functional applied to a curve in the image domain  $\Omega$ . Formally, we aim to compute a path  $v$

$$\begin{aligned} v : [0, 1] &\rightarrow \Omega \subset \mathbb{R}^2 \\ s &\mapsto v(s) = (x(s), y(s)), \end{aligned} \quad (1)$$

such that  $v(0) = v(1)$  and  $v \in \mathcal{C}^2$ . At the position  $s$ , the active contour is represented by  $v(s) = (x(s), y(s))$  where  $x$  and  $y$  are *snake* point coordinate vectors, defined on  $\Omega$ .

Let us consider the following energy

$$E(s) = \int_0^1 E_{\text{int}}(s) + E_{\text{ext}}(s) \, ds, \quad (2)$$

where  $E_{\text{int}}$  denotes the internal energy and  $E_{\text{ext}}$  the external energy. The internal energy penalizes non-smooth curves and its minimization favors the regularity of  $v$ . This is the a priori part of the functional. The external energy depends on the image and enables the curve to fit the area of interest. This last term can be seen as the data fidelity term of the functional.

**Internal energy.** To regularize the shape of the *snake*, the internal energy can be written as

$$E_{\text{int}} = \frac{1}{2} \left( \alpha(s) \|v'(s)\|^2 + \beta(s) \|v''(s)\|^2 \right), \quad (3)$$

where  $\alpha(s)$  and  $\beta(s)$  are weights, which control the *snake* regularity. In the following,  $\alpha$  and  $\beta$  are considered constant to simplify the problem for a numerical application. The minimization of the term  $E_{\text{int}}$  enforces the regularity of the curve.

**External energy.** The external energy represents the data-fidelity term. It is defined as a potential  $P$  on the image domain  $\Omega$

$$P : \Omega \rightarrow \mathbb{R} \quad (4)$$

$$(x, y) \mapsto P(x, y), \quad (5)$$

such that the minimum of  $P$  is the contour of interest in the image. In case of detection of a black contour in a white background, the potential can be the image itself. For smooth images, we consider that the external energy is only represented by the image gradient. Indeed, the active contour must be brought to high gradient areas (which represent image edges) to make the *snake* close to the contours. For this purpose, the image gradient has to be as high as possible. The external energy can therefore be represented by  $E_{\text{ext}} = -P(v(s))$ , where

$$P(x, y) = \left\| \left( \frac{\partial I}{\partial x}(x, y), \frac{\partial I}{\partial y}(x, y) \right) \right\|^2. \quad (6)$$

This potential has an analogous definition with the Canny edges detector [2].

**Global energy.** Combining Equations (2), (3) and (6), our problem can be written as

$$\min_v \int_0^1 G(s, v, v', v'') \, ds, \quad (7)$$

with

$$G(s, v, v', v'') = \frac{1}{2} \left( \alpha \|v'(s)\|^2 + \beta \|v''(s)\|^2 \right) - P(v(s)). \quad (8)$$

The computation of the optimal condition for the minimization of this energy based on the calculus of the variations is detailed in the next section.

## 2.2 Optimal Condition

Let us now write the optimal condition to minimize the functional (7).

The aim is to find the active contour which minimizes the previous equation. To solve it, we use Euler-Lagrange's theorem, which is given by Equation (9)

$$\min_v \int_0^1 G(s, v, v', v'') ds \Leftrightarrow \frac{\partial G}{\partial v} - \frac{\partial}{\partial s} \frac{\partial G}{\partial v'} + \frac{\partial^2}{\partial s^2} \frac{\partial^2 G}{\partial v''} = 0. \quad (9)$$

By analogy with our problem we write the following equalities

$$\frac{\partial G}{\partial v} = -\nabla(P(v(s))), \quad \frac{\partial}{\partial s} \frac{\partial G}{\partial v'} = \alpha v^{(2)}(s), \quad \text{and} \quad \frac{\partial^2}{\partial s^2} \frac{\partial^2 G}{\partial v''} = \beta v^{(4)}(s). \quad (10)$$

Summing up, we obtain the optimal condition

$$\forall s \in [0, 1], \quad \alpha v^{(2)}(s) - \beta v^{(4)}(s) - \nabla(P(v(s))) = 0. \quad (11)$$

Follow the numerical computation of the *snakes*, let us consider the discretization of the curve  $v$  in the next section.

## 3 Snake Discretization and Numerical Scheme

In the following, a discrete active contour, (as known as *snake*)  $V$  is a discretized version of a continuous path  $v$ , composed of  $N$  points (also called vertices) indexed from 1 to  $N$ , represented by the coordinates  $(v_1, \dots, v_N) \in (\mathbb{R}^2)^N$ . The curve may have self-intersections. The snake  $V$  is a regular discretization given by  $v_i = v((i-1)/N)$ . For all  $i$ ,  $v_i$  denotes the coordinates  $(x_i, y_i) \in \mathbb{R}^2$ .

**Discretization of the snake derivatives.** In the case of a discrete active contour (i.e., when Equation (11) is discretized), we have to define  $v^{(2)}$  and  $v^{(4)}$ .

To discretize the derivatives, we use the Euler's method for the second order

$$\frac{\partial^2 v}{\partial s^2} \simeq v_{k+1} - 2v_k + v_{k-1}, \quad (12)$$

as well as the fourth order

$$\frac{\partial^4 v}{\partial s^4} \simeq v_{k+2} - 4v_{k+1} + 6v_k - 4v_{k-1} + v_{k-2}. \quad (13)$$

Let us remark that the usual Euler's method is  $\frac{\partial v}{\partial s} \simeq N \cdot (v_{k+1} - v_k)$ . Nevertheless, removing the factor  $N$  of all the computation makes the parameters of the model independent of the resolution of the discretization. Notice that to simplify the presentation and without loss of generality, Equations (12) and (13) are normalized to the case of constant arc length equal to 1.

These derivatives of discrete *snakes* can be seen as linear operator on the *snakes*. As boundary conditions, we expect the active contour to be closed, therefore we define  $v_0 := v_N$ . With this cyclic boundary conditions, let  $A_2$  denote the tridiagonal matrix (second derivative) and  $A_4$  the pentagonal matrix (fourth derivative)

$$\frac{\partial^2 v}{\partial s^2} \simeq \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -2 & 1 \\ 1 & 0 & 0 & \dots & 1 & -2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix} = A_2 v, \quad (14)$$

and

$$\frac{\partial^4 v}{\partial x_k^4} \simeq \begin{pmatrix} 6 & -4 & 1 & \dots & 1 & -4 \\ -4 & 6 & -4 & \dots & 0 & 1 \\ 1 & -4 & 6 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \dots & 6 & -4 \\ -4 & 1 & 0 & \dots & -4 & 6 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix} = A_4 v. \quad (15)$$

To simplify the notations, let us denote  $A = -\alpha A_2 + \beta A_4$ . The computation of the matrix  $A$  can be performed by `release2D/functions/computeA.m`. To help the implementation,  $A_4$  is approximated by  $A_2^2$  (line 42 of `computeA.m`).

The cubic spline refinement detailed in the paragraph “Adaptive discretization” aims at this reparameterization in order to avoid degeneration of the curve.

**Temporal discretization for the numerical scheme.** Let us denote by  $V$  the discretized path such as  $(v_1, \dots, v_N) \in (\mathbb{R}^2)^N$  and

$$\nabla(P(V)) \in (\mathbb{R}^2)^N, \quad (16)$$

the gradient of the potential computed in the coordinates given in  $V$ . From Equation (11), the optimal condition can be now written as

$$AV - \nabla(P(V)) = 0. \quad (17)$$

We aim to discretize temporally the equation to obtain an iterative algorithm. In this way, the *snake* is going to get closer, step-by-step, to the image contours.

Let  $V_t$  denote the vector  $V$  at time  $t$  and let us consider the following fixed-point problem

$$AV - \nabla(P(V)) = 0 \iff AV - \nabla(P(V)) + V = V. \quad (18)$$

Let us denote  $g(V) = AV - \nabla(P(V))$  and  $\gamma$  the step size of Euler’s temporal discretization. Using Euler implicit method for  $AV$  and explicit method for  $-\nabla(P(V))$ , we finally obtain the following numerical scheme

$$AV_t - \nabla(P(V_{t-1})) = -\frac{1}{\gamma}(V_t - V_{t-1}). \quad (19)$$

By inverting the matrix  $A + \gamma I_d$ , we can obtain  $V_t$  depending on  $V_{t-1}$

$$V_t = (\gamma A + I_d)^{-1} (V_{t-1} + \gamma \nabla(P(V_{t-1}))). \quad (20)$$

The implementation of Equation (20) is done in `release2D/functions/EvolutionSnake2D.m` at line 147, with a pre-computation of  $(\gamma A + I_d)^{-1}$  at line 120. The properties of the functional (7) does not enable us to prove the convergence of such scheme. Moreover, since some flatnesses may appear in the functional, some terms have to be artificially added to numerically ensure the evolution of the numerical scheme in some cases.

**Balloon force in 2D.** To avoid stationary effects during the minimization process due to constant parts of the image, we have added a balloon force as proposed in [6, 5].

The balloon force aims to naturally contract/expand the *snake*, even if the image potential does not attract it [5]. This enables the *snake* to get close to the image when it could not have been achieved by the regularity forces. This balloon force is built from the curve normal vectors, computed in each

point of the active contour from the previous and the following point. For instance, for the point  $v_i$ , let us consider the tangential vector  $\vec{t}_i$ , computed by averaging the previous and the next edges

$$\vec{t}_i = \frac{1}{2} ((v_{i+1} - v_i) + (v_i - v_{i-1})) = \begin{pmatrix} t_x \\ t_y \end{pmatrix}. \quad (21)$$

For the balloon force, we naturally choose the orthogonal vector  $\vec{n}_i$  to expand the active contour

$$\vec{n}_i = \frac{\vec{t}_i^\top}{\|\vec{t}_i\|} = \frac{1}{\sqrt{t_y^2 + t_x^2}} \begin{pmatrix} -t_y \\ t_x \end{pmatrix}. \quad (22)$$

By computing  $\vec{n}$  for each point of the *snake*, we obtain a force

$$F_B = \kappa_B \vec{n}, \quad (23)$$

with the sign of  $\kappa_B$  controlling the expansion or contraction of the active contours. This force is integrated in the numerical scheme in the following way:

$$V_t = (\gamma A + I_d)^{-1} (V_{t-1} + \gamma [\nabla(P(V_{t-1})) + F_B]). \quad (24)$$

This force is computed by `release2D/functions/balloonForce.m`.

In order to obtain a balloon force oriented outside of the polygon, it is oriented clockwise thanks to the function `polygonParity.m` which uses a parity test to check the sense of the polygon once the user draws it. In our code, a negative coefficient  $\kappa_B$  means that the balloon force is inner-oriented whereas a positive coefficient means an outer orientation of the force.

In the following we detail the discretization of  $\nabla(P(V_{t-1}))$ .

**Potential discretization and smoothing.** To compute explicitly  $\nabla(P(V_{t-1}))$ , the image gradient has to be discretized. The following formulas have been taken from [1]. Let  $N$  denote the number of points and  $g$  a discrete function and define the gradient

$$\frac{\partial g_{x,y}}{\partial x} = \begin{cases} \frac{g_{x+1,y} - g_{x-1,y}}{2} & \text{if } 1 < x < N, \\ \frac{g_{2,y} - g_{1,y}}{2} & \text{if } x = 1, \\ \frac{g_{N,y} - g_{N-1,y}}{2} & \text{if } x = N. \end{cases} \quad (25)$$

Likewise in the  $y$  direction we have

$$\frac{\partial g_{x,y}}{\partial y} = \begin{cases} \frac{g_{x,y+1} - g_{x,y-1}}{2} & \text{if } 1 < y < N, \\ \frac{g_{x,2} - g_{x,1}}{2} & \text{if } y = 1, \\ \frac{g_{x,N} - g_{x,N-1}}{2} & \text{if } y = N. \end{cases} \quad (26)$$

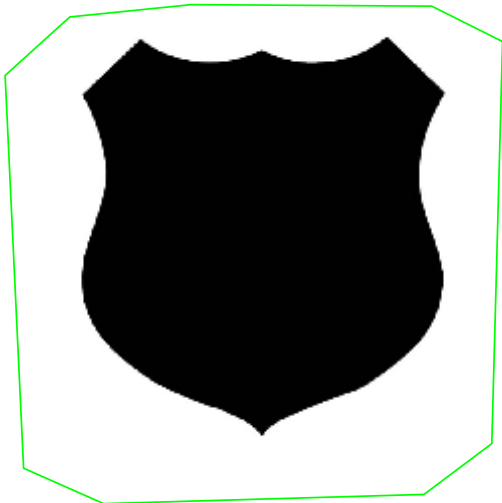
The computation of this gradient is implemented in `release2D/functions/gradientCentered.m`. To compute  $\nabla(P(x, y))$ , we apply these formulas directly on the image with an interpolation available in `release2D/functions/interpSnake2.m`. We then compute pixel-wise the norm of the gradients. Finally, we apply these formulas again to compute  $P$ . To avoid some problems with the noise, the image is blurred with a Gaussian kernel with a standard deviation equal to  $\sigma$ . The parameter of this kernel as well as its influence on the result is discussed in Section 4.



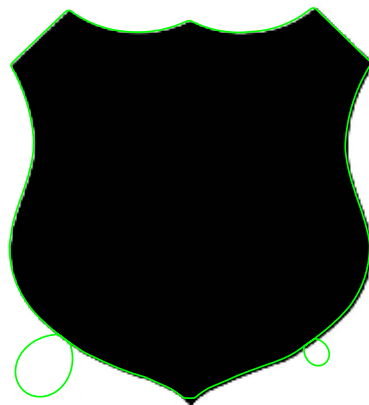
**Stopping criterion.** The computation of the active contour being an iterative process, it has to be stopped. The model being highly non-convex the stopping criterion can only be heuristic. Our strategy is inspired by the convergent algorithm. When the difference between two iterations  $V_t$  and  $V_{t+1}$  is small enough, the algorithm is stopped. To compute the difference between the two vectors, the standard L2-norm is used. In practice, when  $\|V_t - V_{t+1}\| \leq \varepsilon$ , the algorithm is stopped. For practical cases, we have experimentally used  $\varepsilon = 9 \cdot 10^{-2}$ , which is implemented with an interpolation available in `release2D/functions/EvolutionSnake2D.m` at lines 134 and 176. Since there is no convergence guaranty, the algorithm also stops when a maximum number of iterations is reached.

In Section 4, a discussion about the influence of the smoothness of the image and the time-step parameter is proposed.

**Mickey Mouse Ears failure case.** In this paragraph, we describe a failure case that may happen with some set of parameters with some specific initialization values. In the case of an initialization with a polygonal snake, the topology of the initial contour can be modified, and some blobs, looking as the ears of Mickey Mouse can appear.



(a) Initialization of the algorithm.



(b) Mickey Mouse Ears failure.

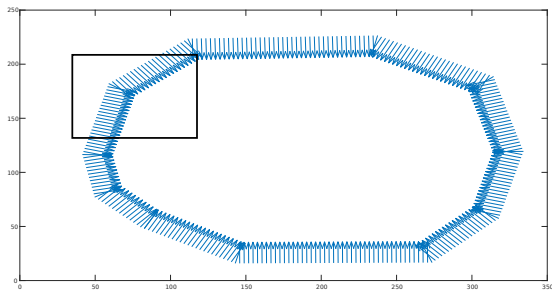
Figure 1: In some conditions, the evolution of the snake can produce some double points on the curve, even if the initialization does not contain one. This image has been produced with  $\alpha = 1750$ ,  $\beta = 700$ ,  $\kappa_B = -90$ ,  $\gamma = 0.00005$ ,  $\sigma = 1$  and 10000 iterations.

For instance in Figure 1, whereas the initial curve is a polygon with no multiple points, the iterative algorithm produces some double points (the curve crosses itself). To understand the origin of this problem, let us display locally the balloon forces around the snake, and focus on the angle of the snake (see, e.g., Figure 2).

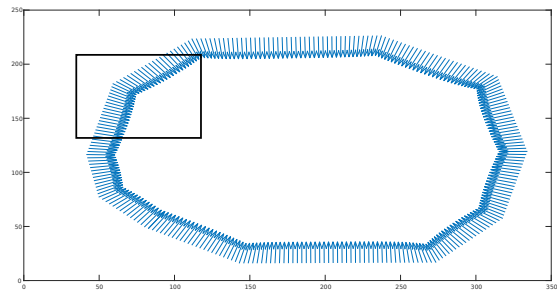
One can see the balloon forces are crossed, that will produce a crossing of the points of the curves, dealing with a small blob. Moreover, after one crossing, the balloon force will continue to favor an increase of the blob.

In order to avoid this effect, the smoothing of the balloon force has been proposed. To this aim, the convolution of the balloon force with a normalized 1D Gaussian kernel centered in 0 and with

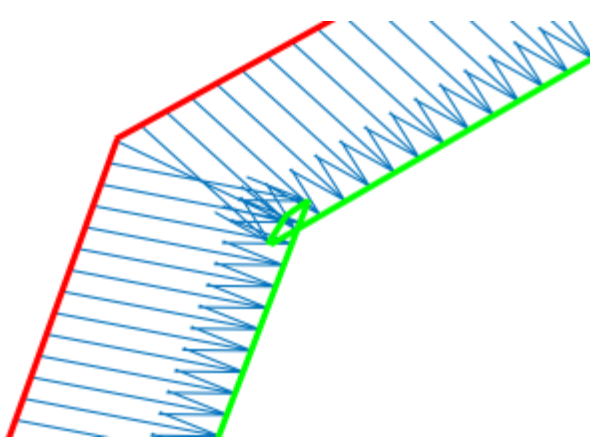




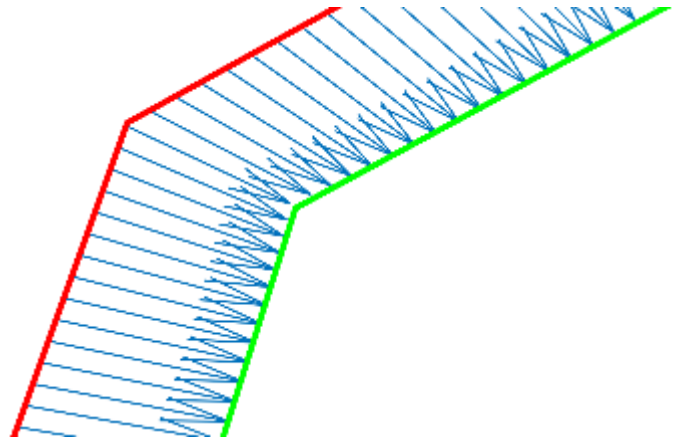
(a) Field of balloon forces applied to a snake.



(b) The same field with a Gaussian smoothing.



(c) Zoom on (a). Initial snake (red) and resulting snake (green).



(d) Zoom on (b). Initial snake (red) and resulting snake (green).

Figure 2: When the balloon force is large enough, and the snake (in red in Figure (c)) is modified with respect to this force, a loop occurs in the resulting snake (in green in Figure (c)). With the Gaussian smoothing, this phenomenon is avoided (see, e.g., (d)).

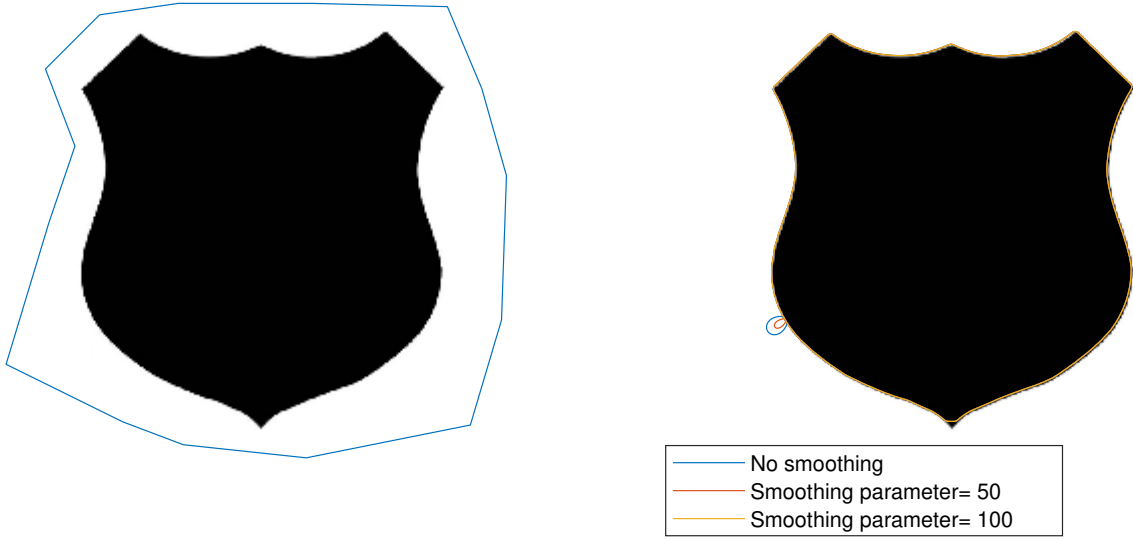
a standard-deviation denoted by  $\sigma_B$  is used to regularize this force. This convolution, considered with cyclic boundary conditions, is applied directly on each coordinates of the  $F_B$  vector defined in Equation (23). The smoothing version of  $F_B = \begin{pmatrix} F_{B,x} \\ F_{B,y} \end{pmatrix}$  is defined as

$$\tilde{F}_{B,i} = \begin{pmatrix} (G_{\sigma_B} * F_{B,x})_i \\ (G_{\sigma_B} * F_{B,y})_i \end{pmatrix}. \quad (27)$$

The smoothing of the balloon force is implemented in `release2D/functions/smoothForces.m`. The computation of the convolution is done through FFT transformation in order to make the circular conditions easier to compute. The balloon force with and without regularization can be seen in Figure 2. With the Gaussian smoothing the crossing of the force is lower. In Figure 3 the smoothing of the balloon force shows some positive effect since it can remove the blob from the resulting snakes.

The computation of the contour following the snake evolution model including a smooth balloon force is described in Algorithm 1. A hack has been added in order to cancel the balloon force when the force  $P'(V)$  deriving from the external energy is strong enough.

**Adaptive discretization.** In order to enable the snake to fit some small structures, a refinement of the discretization of the active contour is proposed. To this aim, the points of the snake are interpolated with cubic splines. These splines are defined as follows: the splines connect two adjacent



(a) Initialization of the algorithm.

(b) Result after convergence.

Figure 3: The smoothing of the balloon force has a valuable effect on the reduction of the blobs. With  $\sigma_B = 100$ , there is no blob. Is the smoothing is low  $\sigma_B = 50$  or without smoothing, some blob appears in the bottom-left side. The others parameters are the same as in Figure 1.

points of the vector defining the active contours; the splines should be connected smoothly in the sense that the second order derivative of the spline has to be equal in the connecting points. The method and the implementation is widely inspired by [17], Chapter 3. Let us define a discretized path in  $\mathbb{R}^2$  defined by the points

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x(t_i) \\ y(t_i) \end{pmatrix}, \quad (28)$$

with  $i = 1, \dots, N$ . Let us assume the following conventions:  $t_{N+1} = t_1$ ,  $x_{N+1} = x_1$ , and  $y_{N+1} = y_1$ . In the interval  $t \in [t_i, t_{i+1}]$ , the linear interpolation can be written as

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = A \begin{pmatrix} x_i \\ y_i \end{pmatrix} + B \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}, \quad (29)$$

with  $A = \frac{t_{i+1} - t}{t_{i+1} - t_i}$  and  $B = 1 - A$ . In order to get a smooth path, we aim to define the second derivative at the connection points

$$\frac{\partial^2 x}{\partial t^2}(t_i) = x''_i, \quad (30)$$

$$\frac{\partial^2 y}{\partial t^2}(t_i) = y''_i, \quad (31)$$

with  $x''_i$  and  $y''_i$  given values. As done before, we use the conventions  $x''_{N+1} = x''_1$ , and  $y''_{N+1} = y''_1$ . In order to constraint the smoothness of the curve, the interpolation is written as a third order polynomial

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = A \begin{pmatrix} x_i \\ y_i \end{pmatrix} + B \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} + C \begin{pmatrix} x''_i \\ y''_i \end{pmatrix} + D \begin{pmatrix} x''_{i+1} \\ y''_{i+1} \end{pmatrix}, \quad (32)$$

---

**Algorithm 1:** Algorithm to return a polygon given by a snake evolution model.

---

```

Input      : input image  $I$  (image domain:  $\Omega \subset \mathbb{R}^2$ ), initial contour  $V_{\text{init}} \in (\mathbb{R}^2)^n$ 
Output    : final contour  $V \in (\mathbb{R}^2)^n$ 
 $V \leftarrow V_{\text{init}}$                                      // Contour initialization
 $I \leftarrow G_\sigma * I$ , with  $G_\sigma$  a 2D Gaussian kernel with standard-deviation  $\sigma$ .
// Computation of the regularization matrix
Computing  $A_2$  defined in Equation (14)
 $A \leftarrow -\alpha A_2 + \beta A_2^2$                          // Equations (15) and (20)
 $P' \leftarrow \nabla \|\nabla I\|^2$                          // Contribution of external energy with Equation (6)
 $C \leftarrow (\text{true})^n \in (\{\text{true}, \text{false}\})^n$     // Hack : enable the balloon force
 $nb_{\text{iteration}} \leftarrow 1$ ;  $error \leftarrow 1$         // Stopping conditions
// Iterative process
while  $nb_{\text{iteration}} \leq n_{\text{max}}$  AND  $error > 9 \cdot 10^{-2}$  do
     $\tilde{F}_B$  computed from Equation (27)                 // Computation of the balloon forces
    // Hack : disable balloon force when External energy is large enough
    if  $\|\tilde{F}_{B,i}\| \leq \|P'(V)_i\|$  then
         $C_i = \text{false}$ 
    if NOT  $C_i$  then
         $\tilde{F}_{B,i} \leftarrow 0$ 
     $V_1 \leftarrow (\gamma A + I_d)^{-1} \left( V + \gamma \left( P'(V) + \tilde{F}_{B,i} \right) \right)$  // Numerical scheme Equation (24)
    // Cubic spline refinement
    if  $nb_{\text{iteration}} \bmod 100 = 0$  then
         $V_1 \leftarrow$  refinement of  $V_1$ , update of  $n$ .
         $C \leftarrow (\text{true})^n \in (\{\text{true}, \text{false}\})^n$  // Hack : enable the balloon force
         $A \leftarrow -\alpha A_2 + \beta A_2^2$                  // Equations (15) and (20)
    // Computation of the stopping criteria
     $nb_{\text{iteration}} \leftarrow nb_{\text{iteration}} + 1$ 
    if  $V$  and  $V_1$  have the same size then
         $error \leftarrow \|V - V_1\|_2^2$ 
     $V \leftarrow V_1$ 
return  $V$ 

```

---

with  $C = \frac{1}{6}(A^3 - A)(t_{i+1} - t_i)^2$  and  $D = \frac{1}{6}(B^3 - B)(t_{i+1} - t_i)^2$ . The second order derivative is thus a linear interpolation of the constraints when  $t \in [t_i, t_{i+1}]$

$$\frac{\partial^2 x}{\partial t^2}(t) = Ax''_i + Bx''_{i+1}, \quad (33)$$

$$\frac{\partial^2 y}{\partial t^2}(t) = Ay''_i + By''_{i+1}. \quad (34)$$

We have now  $2N$  unknowns  $x''_i$  and  $y''_i$  that we have to compute. To this aim, we assume the continuity of the first order derivative. With some rearrangements, for  $i = 1, \dots, N$

$$\frac{x_{i+1} - x_i}{t_{i+1} - t_i} - \frac{x_i - x_{i-1}}{t_i - t_{i-1}} = \frac{t_i - t_{i-1}}{6}x''_{i-1} + \frac{t_{i+1} - t_{i-1}}{3}x''_i + \frac{t_{i+1} - t_i}{6}x''_{i+1} \quad (35)$$

$$\frac{y_{i+1} - y_i}{t_{i+1} - t_i} - \frac{y_i - y_{i-1}}{t_i - t_{i-1}} = \frac{t_i - t_{i-1}}{6}y''_{i-1} + \frac{t_{i+1} - t_{i-1}}{3}y''_i + \frac{t_{i+1} - t_i}{6}y''_{i+1}, \quad (36)$$

with the convention  $t_0 = t_N$ ,  $x_0 = x_N$ ,  $y_0 = y_N$ ,  $x_0'' = x_N$ , and  $y_0'' = y_N$ . We then get  $2N$  equations with  $2N$  unknowns. Solving this linear system gives us the second order derivative constraints at each connection points. To solve it, the variable  $t$  is discretized with a step equal to  $\frac{1}{N}$ . Equation (32) gives then the values of the spline between connecting points.

Figure 4 shows an example. The set of points  $(x_i, y_i)$  are the red circles, which are connected by smooth curves.

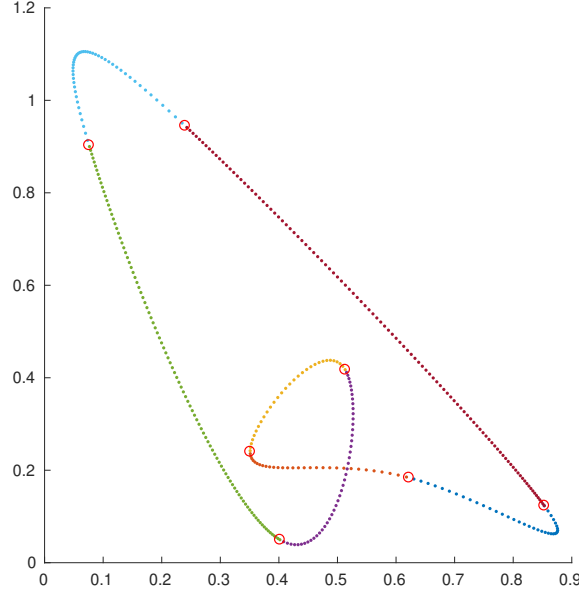


Figure 4: The cubic splines connect the points (red circles) with smooth (polynomial) interpolations. This curve can be regularly sampled with respect to  $t$  (colored dots). Notice that, theoretically, this spline interpolation can change the curve topology, but it is not the case on our practical experiments.

Once the coefficients of Equation (32) have been computed, the splines are uniformly sampled. To this aim, a numerical computation of the length of each spline is done with a left Riemann sum to compute the following path length with a subdivision of the  $[0, 1]$  interval

$$\int_0^1 \left\| \left( \frac{\partial x}{\partial t}(t), \frac{\partial y}{\partial t}(t) \right)^t \right\| dt. \quad (37)$$

The left Riemann approximation can be written as

$$\frac{1}{N} \sum_{i=0}^{N-1} \left\| \left( \frac{\partial x}{\partial t}(i/N), \frac{\partial y}{\partial t}(i/N) \right)^t \right\|. \quad (38)$$

Then, the spline is homogeneously sampled: the complete path defined on  $[0, 1]$  is subdivided relatively to the lengths of each spline. To this aim, the interval  $[0, 1]$  is uniformly sampled, and for each time step in this interval it can be determined to which spline of the path it corresponds. The point of the path at this time step of the interval can be computed from Equation (32). All these computations are listed in Algorithm 2 and performed in `release2D/functions/splinesInterpolation2D.m`.

The refinement strategy, as well as the influence of the splines refinement on a numerical example, are studied in Section 4.

---

**Algorithm 2:** Algorithm to return a polygon with a different number of points after refinement with cubic splines.

---

**Input** : Closed path  $V$  defined by points  $(X(i), Y(i))$  with  $i \in [0, N]$   
**Output** : New path interpolated with cubic splines  $V_2$  with  $N_2$  points of coordinates  $(X_2, Y_2)$ .  
**Parameters:**  $N_2$  the new number of wished points

```

// 1- Compute second order derivatives of the interpolating function, both
// are computed from a system with  $2N$  equations and  $2N$  unknown
Given  $X$  and  $Y$ , compute  $X''$  and  $Y''$  as defined in Equation (35) and (36).
// 2- Compute the arc length of each spline
 $M \leftarrow 100$  // left Riemann sum resolution.
for  $i = 1$  to  $N$ , with  $i + 1 = 1$  when  $i = N$  do
    Compute  $S'_i$  with respect to  $t$  between  $(X(i), Y(i))$  and  $(X(i + 1), Y(i + 1))$  from the  $X''$ 
    and  $Y''$  values (the derivative of the spline  $S_i$  is formally computed from Equation (32)).
    Sample the function  $S'_i$  at  $M$  points  $t_j$  uniformly distributed between 0 and 1.
     $path\_length(i) \leftarrow \frac{\sum_j \|S'_i(t_j)\|}{M \cdot N}$  // Left Riemannian sum to approximate
    Equation (38).
// 3- Sample the splines to get the points of the new path
Compute  $T_2$ , a vector of  $N_2$  uniformly distributed between 0 and  $\sum_{j=1}^N path\_length(j)$ 
for  $i = 1$  to  $N$ , with  $i + 1 = 1$  when  $i = N$  do
    for  $t \in T_2$  such as being between  $\sum_{j=1}^i path\_length(j)$  and  $\sum_{j=1}^{i+1} path\_length(j)$  do
         $A \leftarrow \frac{\sum_{j=1}^{i+1} path\_length(j) - t}{path\_length(i+1)}$ 
         $B \leftarrow 1 - A$ 
         $C \leftarrow 1/6(A^3 - A)(i/N - (i - 1)/N)^2$ 
         $D \leftarrow 1/6(B^3 - B)(i/N - (i - 1)/N)^2$ 
         $X_2 \leftarrow AX(i) + BX(i + 1) + CX''(i) + DX''(i + 1)$  // Equation (32)
         $Y_2 \leftarrow AY(i) + BY(i + 1) + CY''(i) + DY''(i + 1)$  // Equation (32)
        Add  $(X_2, Y_2)$  to  $V_2$ .
return  $V_2$ 

```

---

## 4 2D Segmentation Results

In this section, we propose to study the influence of the parameters of the snakes model in the two dimensions examples. Let us study the following parameters: the smoothing terms  $\alpha$ ,  $\beta$ ,  $\sigma$ , the time-step  $\gamma$ , and the cubic spline refinement. For all the tested images, the initial contour  $V_{t=0}$  is given by coarsely defining manually various points outside of the target as illustrated by Figure 3(a).

**Influence of  $\alpha$  and  $\beta$ .** In the internal energy defined in (3), the two parameters  $\alpha$  and  $\beta$  tune the regularity of the snake shape. The  $\alpha$  parameter rules the stretching between the points of the snake, whereas the  $\beta$  parameter rules the second-order derivative of the path.

The influence of the  $\alpha$  parameter can be seen in Figure 5. The results have been provided with various  $\alpha$  values. The higher  $\alpha$ , the lower the perimeter of the curve. This result matches well with the theoretical aspect: when  $\alpha$  is high, the differences between the points are penalized, thus they are closer to their neighbors. In case when  $\alpha$  is too low, some Mickey Mouse Ears effect can occur as in Figure 3.

In order to see the influence of the  $\beta$  parameter, let us compare two results, one with a large



Figure 5: Influence of the parameter  $\alpha$  on the segmentation. The results are produced with  $\beta = 500$ ,  $\kappa_B = -50$ ,  $\gamma = 0.0001$ ,  $\sigma = 1$ ,  $\sigma_B = 70$ . The initialization is displayed on the left-hand side. The stopping criterion is defined as  $\|V_t - V_{t+1}\| \leq 9 \cdot 10^{-2}$  and the initialization snake is chosen as in Figure 1.

value and another with a small value. The results of the algorithm with  $\beta = 4$  and with  $\beta = 40000$  can be seen in Figure 6. With a small value of  $\beta$ , the contour has sharp edges, since the term based on the second-order derivative doesn't penalize much the curvature of the path. With a large value of  $\beta$ , large curvature values (the angular parts of the path) are penalized, thus the result is smooth.



(a)  $\beta = 4$ .

(b)  $\beta = 40000$ .

Figure 6: Influence of the parameter  $\beta$  on the segmentation. The results are produced with  $\alpha = 17.5$ ,  $\kappa_B = -50$ ,  $\gamma = 0.0002$ ,  $\sigma = 1$ ,  $\sigma_B = 70$ . The stopping criterion is defined as  $\|V_t - V_{t+1}\| \leq 9 \cdot 10^{-2}$  and the initialization snake is chosen as in Figure 3.

**Influence of  $\sigma$  and the step time  $\gamma$ .** Since there is no convergence guaranty for the numerical scheme, let us numerically study the behavior of the iterative algorithm.

The original problem (7) is not convex, but the convolution of the potential with a Gaussian filter changes the properties of the energy. Since this filtering has a low-pass effect, the energy is smoother when  $\sigma$  is higher.

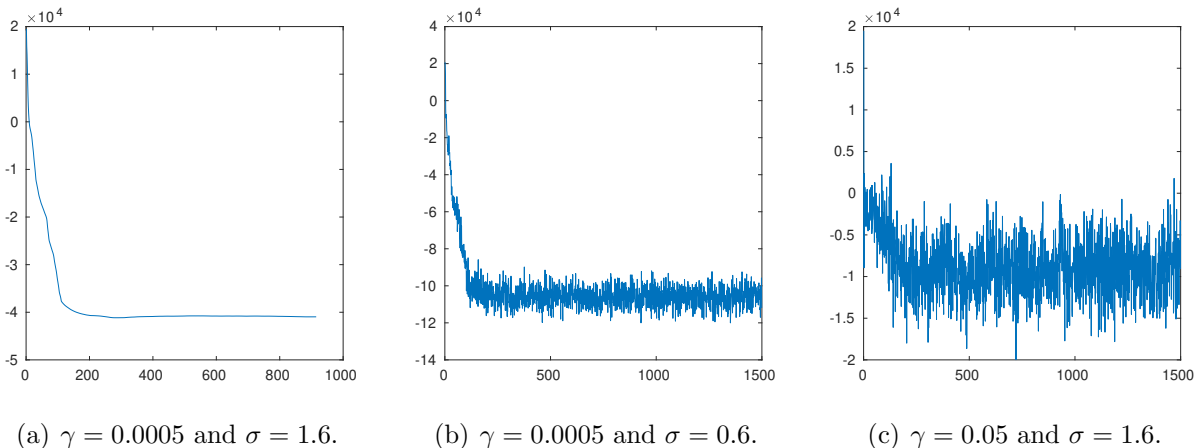


Figure 7: Value of the energy (7) with respect to the number of iterations in Algorithm 1. We can see the influence of the parameters  $\sigma$  and  $\gamma$  on the decreasing of the energy. The results are performed with  $\alpha = 750$ ,  $\beta = 1000$ ,  $\kappa_B = -30$ ,  $\sigma_B = 70$  and 1500 iterations.

Assume that  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is convex, and differentiable. If

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \text{ for any } x, y$$

then  $\nabla f$  is Lipschitz continuous with constant  $L > 0$ .

The smoothing induced by the Gaussian filtering can be seen as a decrease of the Lipschitz constant of the gradient of the energy, thus it can improve the convergence properties of the numerical scheme (see, e.g. [7], Theorem 2.2.4). Nevertheless, it modifies the data of the problem by blurring it, since the result may fit worse to the original image contours.

For the experiments of Figure 7, we have used the example of Figure 6. In Figure 7 we have drawn the value of the functional (7) during the time evolution iterations of the numerical scheme (20). Two parameters have some influence into the convergence of the numerical scheme:  $\sigma$  and  $\gamma$ . The influence of the  $\gamma$  parameter is the standard behavior for the gradient descent time-step for non-convex problems. When  $\gamma$  is small enough and the functional (7) is smooth enough ( $\sigma = 1.6$ ), the energy of the numerical scheme decreases to an asymptotic value (see, e.g., Figure 7(a)). When  $\sigma$  becomes smaller ( $=0.6$ ) the functional (7) is sharper, the numerical scheme has an energy that decreases and oscillates around a local minimum (see, e.g., Figure 7(b)). When  $\gamma$  is too large, the numerical scheme has a random behavior (see, e.g., Figure 7(c)).

**Cubic spline path refinement.** In order to refine the path, a sampling of the cubic spline interpolation can be used, as introduced in Section 3. Our strategy consists in the computation of the maximum distance between the points every 100 iterations. If it exceeds 2 pixels, the set of points is interpolated by cubic splines and the new path is sampled with 10% more points.

In Figure 8, the influence of the refinement is shown. The snake does not fit the contours in the concave parts with the original method. With the refinement, the snake is closer to the contour.

**Practical example of the lungs.** For non binary images, it is helpful to increase the contrast of the images to better fit the contours. To this aim, we have used a linear mapping that saturates 10% of the values of the pixels. With this transformation, the image range is between 0 and 255 for all examples. Since the active contour is sensitive to its parameters, this transformation reduces this sensitivity.

In the following, we propose to segment the lungs with active contours. Figure 9 shows the active contour method to segment lungs. The contour fits well a lung in its smooth part, whereas it is quite



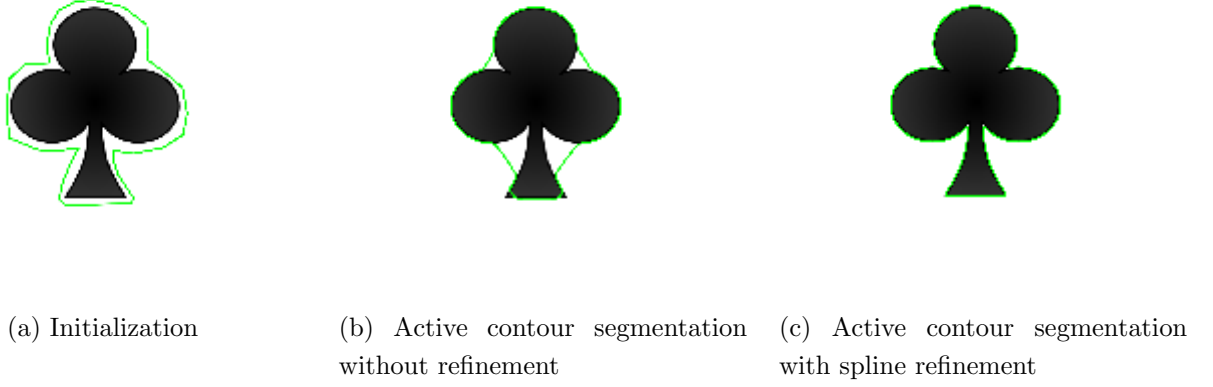


Figure 8: Effect of the discretization refinement on the segmentation result with a shape containing concavities. The results are produced with  $\alpha = 1500$ ,  $\beta = 2000$ ,  $\kappa_B = -50$ ,  $\gamma = 0.0001$ ,  $\sigma = 1$ ,  $\sigma_B = 70$  and 10000 iterations.

inaccurate in the interior part which is a complex contour. It is similar to the results existing in the literature [16] because such accuracy is enough for radiotherapy treatment planning or for training simulators.

Table 1 presents various images with sets of suitable parameters.



Figure 9: Effect of the discretization refinement on the segmentation result with a shape containing concavities. The result is produced with  $\alpha = 1500$ ,  $\beta = 2000$ ,  $\kappa_B = -50$ ,  $\gamma = 0.0001$ ,  $\sigma = 1$ ,  $\sigma_B = 70$  and 10000 iterations.

In the next section, an extension to the 3D case is proposed.






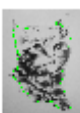




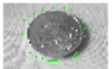

Image + initialization	size	$\alpha$	$\beta$	$\sigma$	$\gamma$	$\kappa_B$	$\sigma_B$	Iterations	Results
	$417 \times 423$	$2 \cdot 10^3$	$8 \cdot 10^2$	1.08	$6.5 \cdot 10^{-4}$	-70	70	1700	
	$886 \times 886$	$1.75 \cdot 10^3$	$7 \cdot 10^2$	1.4	$1.5 \cdot 10^{-3}$	-100	100	950	
	$450 \times 626$	$5 \cdot 10^2$	$4 \cdot 10^2$	$8 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	-20	100	1000	
	$600 \times 450$	$8 \cdot 10^2$	$5 \cdot 10^2$	$5 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	-15	100	2000	
	$464 \times 484$	$7 \cdot 10^2$	$3.7 \cdot 10^3$	1	$5 \cdot 10^{-4}$	-30	100	950	
	$600 \times 385$	$1.7 \cdot 10^3$	$3.7 \cdot 10^3$	$5 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	-10	100	2000	

Table 1: Various images with a set of valid parameters.

## 5 Extension in Three Dimensions

In this work, for the sake of simplicity, the 3D case has been considered by extending the 2D numerical scheme instead of redefining model (7) with surface integrals as done in [6]. In our framework, the source code relative of the 3D implementation of active contours is in the `release3D` folder. Similarly to the 2D case, it is composed of the main Matlab script `MAIN.EvolutionSnake3D.m` and a `functions` folder.

**3D extension of the 2D optimal conditions.** In 2D, the optimal condition, after discretization, reads as Equation (17), i.e.

$$AV - \nabla(P(V)) = 0. \quad (39)$$

In this equation,  $AV$  represents a smoothing term which ensures the regularity of the *snakes*, whereas  $\nabla(P(V))$  is a data-fidelity term. According to Equations (25) and (26), the definition of this term is only based on the pixel representation of the original image. Thus, the extension of this term to the 3D case is obvious, just by extending the definition of the gradient.

The definition of  $A$  settles on the representation on the *snake* itself. In the following, we assume that the natural 3D extension of the *snake* is a triangular mesh. This mesh is defined as a set of 3D points (vertices) in combination with an adjacency matrix defining the edges of a graph. In the following, we assume that all the faces of this mesh are triangles, but this is not a restriction for our definition of the regularity matrix  $A$ .

**Computation of the regularity matrix.** Let us define  $\{v_i \text{ with } 1 \leq i \leq N\}$  the set of the vertices of the given mesh. Precisely, we consider that  $v_i$  is a point in a three dimensional space, thus

$v_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ . In this section, we assume that the adjacency matrix is defined as follows

$$\Lambda = (\alpha_{i,j})_{1 \leq i \leq N, 1 \leq j \leq N}, \quad (40)$$

$$\text{with } \alpha_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ neighbor of } v_j, \\ 0 & \text{otherwise.} \end{cases} \quad (41)$$

Let us remark that the number of neighbors of a vertex  $i$  is given by summing the  $i$ -th column of the matrix  $\Lambda$ .

The difference finite formulas on the curve (see Equation (12), and (13)) have to be adapted to the mesh. For this purpose, let  $n_i$  denote the number of neighbors of vertex  $v_i$ . Those vertices are given by an adjacent matrix within `triangulation2adjacency.m`. The second order (Equation (12)) derivative was defined as a linear combination of the current vertex with its direct neighbors. As an extension, we propose to define the second order gradient on the mesh by the sum of the neighbors minus the vertex itself multiplied by the number of its neighbors. Thus, the matrix  $A_2$  for the second order is defined as

$$A_2 = (a_{2,i,j})_{1 \leq i \leq N, 1 \leq j \leq N}, \quad (42)$$

$$\text{with } a_{2,i,j} = \begin{cases} -n_i & \text{if } i = j, \\ 1 & \text{if } i \text{ neighbor of } j, \\ 0 & \text{otherwise.} \end{cases} \quad (43)$$

This formula is a direct extension of the second order derivative filter  $[1, -2, 1]$ . Notice that the computation using matrix  $A_2$  does not take into account the spacing between the vertices. The matrix for the fourth order is simply given by

$$A_4 = A_2^2. \quad (44)$$

This matrix operation is easy to compute with Matlab, but in the C++ implementation it can be more easily computed by the following formula

$$A_4 = (a_{4,i,j})_{1 \leq i \leq N, 1 \leq j \leq N}, \quad (45)$$

$$\text{with } a_{4,i,j} = \begin{cases} 1 & \text{if } j \text{ second order neighbor of } i, \\ -n_i - n_j & \text{if } i \text{ neighbor of } j, \\ \sum_{i=1, i \neq j}^n a_{4,i,j} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (46)$$

Denoting

$$A = -\alpha A_2 + \beta A_4 \quad (47)$$

and  $V = (x \ y \ z)^\top$ , we get the final numerical scheme, identical to the 2D case

$$V_t = (\gamma A + I_d)^{-1} (V_{t-1} + \gamma \nabla(P(V_{t-1}))). \quad (48)$$

The computation of matrix  $A$  is performed by `release3D/functions/computeA3D.m`. The implementation of Equation (48) is done in `release3D/functions/EvolutionSnake3D.m` for each of the coordinates, at lines 152 to 154, with a pre-computation of  $(\gamma A + I_d)^{-1}$  at line 113.

$\nabla(P(V_{t-1}))$  is computed from the voxel image: the image gradient is discretized for each direction. As in the 2D case, the balloon force can help the convergence of the *snake* to an image contour.

**Balloon force in 3D.** To extend the 2D computation of the balloon force, the definition of the normal force of the mesh is required. This normal vector can be naturally computed on each face of the mesh. Let us consider a face composed of the vertices  $v_1$ ,  $v_2$  and  $v_3$  and let  $\vec{N}_i$  denote the area of the  $i$ -th face, computed from the directed cross product

$$\vec{N}_i = (v_1 - v_2) \wedge (v_3 - v_1). \quad (49)$$

Next, for each vertex, the normal vectors of the adjacent faces are averaged to give the normal vector at this vertex. The balloon force is obtained for every vertex with the same computation as Equation (23). This force is computed by `release3D/functions/EvolutionSnake3D.m` for each of the coordinates.

The triangles are given by an initial mesh from a sphere where the vertices are indexed such as the normals are all in the same direction. While the mesh is evolving, the topology does not change and the normals will still be computed from the same vertex indices.

**Meshing and initializing.** To initialize the active contour, a sphere is meshed from an octahedron subdivided two times such as new faces are created from the added vertex at the middle of each edge. Every vertex of the sphere is moved such as its distance to the sphere center is equal to the sphere radius. The code to compute this initial mesh is given by `trisphere.m`. Then we expand step by step the snake following the evolution equations. An illustration of the mesh evolution is given on Figure 10. Contrary to the 2D framework, the number of contour vertices does not evolve with the iterations, therefore the number of triangles remains the same and equal to the number constituting this initial sphere.

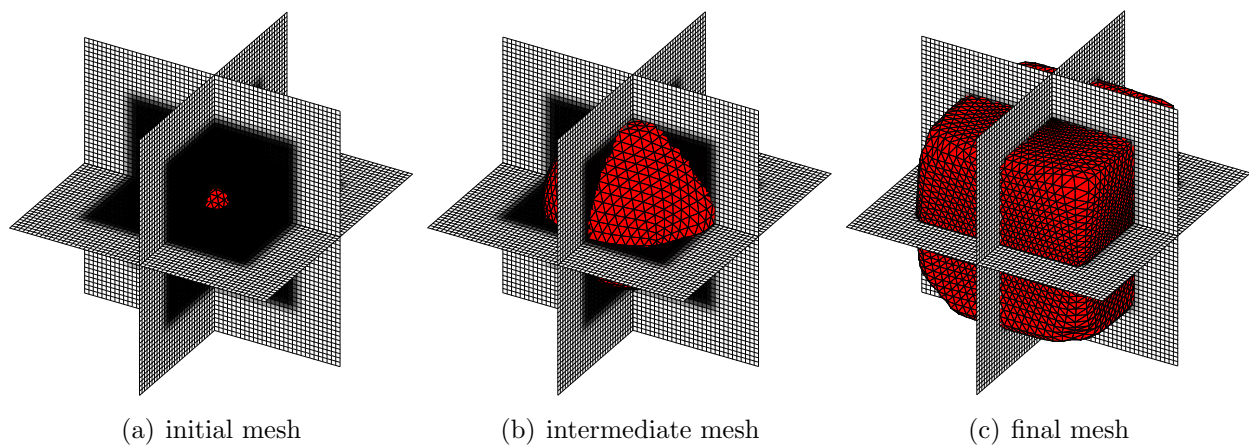


Figure 10: Evolution of the 3D contour to match a 3D image of a cube. The image voxels are illustrated on three slices cut in the middle of the image. The mesh of the contour is in red.

## 6 Numerical Results in 3D

Synthetic 3D data have been initially tested. Those data are generated by `blackCube.m`, `blackOval.m` and `blackOvals.m`. They are in the `data` folder as well as a 3D medical image of lungs.

The illustrative result of Figure 10 has been produced with  $\alpha = 500$ ,  $\beta = 170$ ,  $\kappa_B = -3000$ ,  $\gamma = 3 \times 10^{-5}$ ,  $\sigma = 1$ ,  $maxArea = 2$  and 300 iterations. The same parameters are used in the following experiments except when changes are mentioned.

The influence of the main parameters has been tested. The same behaviors as in the 2D case has been observed. Parameter  $\alpha$  acts on the stretching between points to allow the smoothness of the

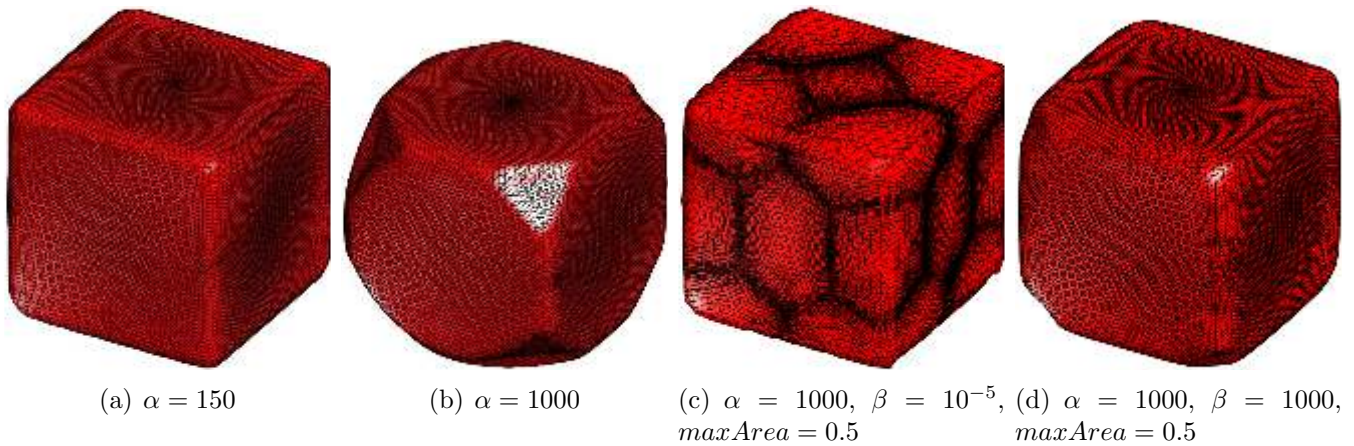


Figure 11: Influence of parameters alpha and beta.

contour. It has been checked and the results are displayed on Figure 11 (a) and (b). The value of  $\alpha$  should not be too large in order to fit the image boundaries, otherwise the contour is too smoothed and it can no longer fill the cube corners (Figure 11 (b)). For an even greater  $\alpha$ , the smoothing forces become stronger than the balloon force and the sphere narrows and converges to a point.

Parameter  $\beta$  acts on the curvature part of the smoothness of the surface. A study of its influence is illustrated on Figure 11 (c) and (d). A small value for  $\beta$  gives a more irregular and bumpy surface (Figure 11 (d)) while it becomes smoother as  $\beta$  expands (Figure 11 (c)).

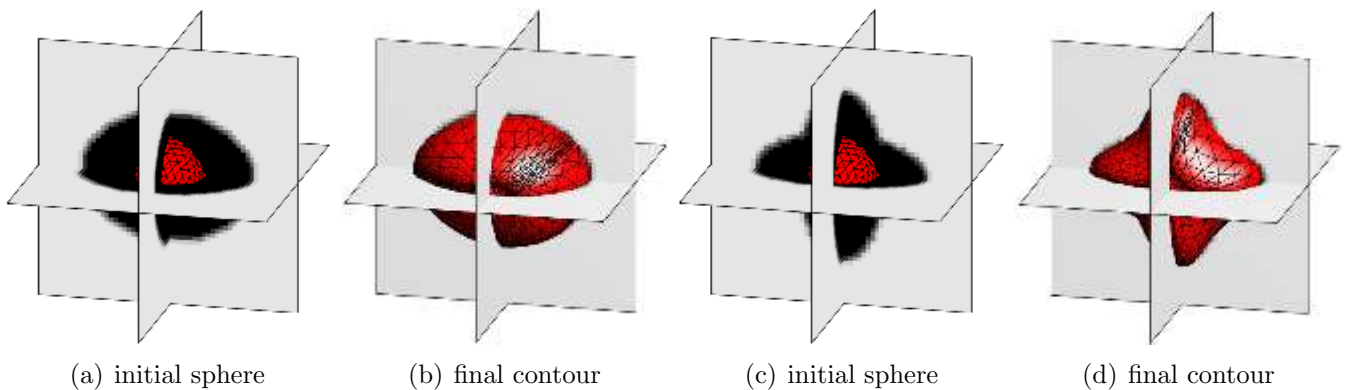


Figure 12: Results on synthesised 3D images. The image voxels are illustrated with three slices and the mesh is in red.

The method has also been tested on other synthesised 3D images. The result is illustrated on Figure 12. They have been produced with the default parameters given at the beginning of this section. The first image (Figure 12 (a) and (b)) is an ellipsoid and the second image is composed of two perpendicular ellipsoids (Figure 12 (c) and (d)). Both the initial and the final steps of the contour evolution are displayed.

Similarly to the 2D case, our method has been carried out on the practical example of the lungs. Results are illustrated on Figure 13. The parameter set has been left unchanged. Similarly to the synthesized 3D images, the active contours fit well the image. The initial contour is also a sphere located inside the lung.



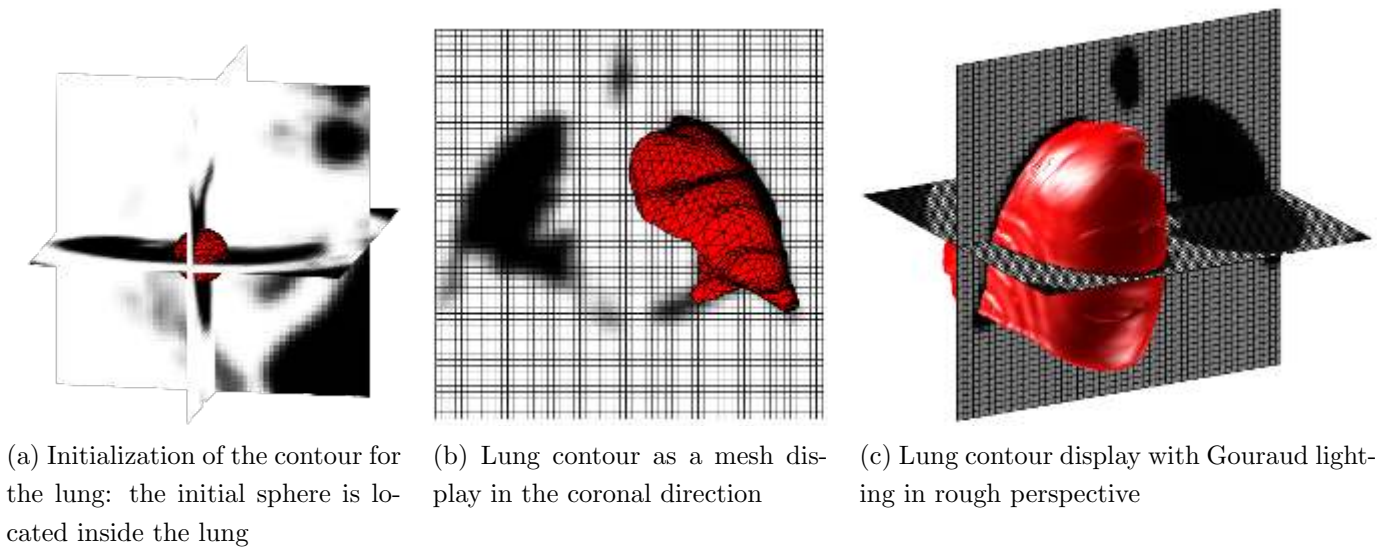


Figure 13: Results on medical images of lungs.

## 7 Conclusion

In this work, we have recalled the general principles of the active contours with meshes. The work has been performed both for 2D images as well as for 3 dimensions images. The discretization scheme and the numerical resolution are detailed. The various engineering hacks that are more or less explained in the literature are detailed in order to reproduce the method. The parameters are given and their effects on the results are discussed, still in 2D and 3D. We have also presented a concrete segmentation case with medical images and the extraction of lungs contours.

Finally, we can conclude that this approach preserves the initial topology of the contour, which can be an advantage when the shape of the object to segment is known. The drawback of this method is the sensitivity to the parameters but if the regularity of the shape does not vary too much, the tuning remains reliable.

## 8 Image Credits

The images credits of the images in Table 1 are: *shamrock*, *shield*, *cat*, *vase* and *coin*, personal work. The medical image is a slice extracted of an anonymized 3D CT scan from the dir-lab database [3] labeled as "4DCT 1" on their webpage<sup>4</sup>.

## References

- [1] X. BRESSON AND T.F. CHAN, *Fast dual minimization of the vectorial total variation norm and applications to color image processing*, Inverse Problems and Imaging, 2 (2008), pp. 455–484. <http://dx.doi.org/10.3934/ipi.2008.2.455>.
- [2] J. CANNY, *A computational approach to edge detection*, in Readings in Computer Vision, Elsevier, 1987, pp. 184–203. <https://doi.org/10.1016/B978-0-08-051581-6.50024-6>.
- [3] R. CASTILLO, E. CASTILLO, R. GUERRA, V. E. JOHNSON, T. MCPHAIL, A. K. GARG, AND T. GUERRERO, *A framework for evaluation of deformable image registration spatial accuracy*

<sup>4</sup><http://www.dir-lab.com>

- using large landmark point sets*, Physics in Medicine and Biology, 54 (2009), pp. 1849–1870. <https://doi.org/10.1088/0031-9155/54/7/001>.
- [4] T.F. CHAN AND L.A. VESE, *Active contours without edges*, IEEE Transactions on Image Processing, 10 (2001), pp. 266–277. <https://doi.org/10.1109/83.902291>.
- [5] L.D. COHEN, *On active contour models and balloons*, CVGIP: Image Understanding, 53 (1991), pp. 211–218. [https://doi.org/10.1016/1049-9660\(91\)90028-N](https://doi.org/10.1016/1049-9660(91)90028-N).
- [6] L.D. COHEN AND I. COHEN, *Finite-element methods for active contour models and balloons for 2-D and 3-D images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 1131–1147. <https://doi.org/10.1109/34.244675>.
- [7] J-B. HIRIART-URRUTY AND C. LEMARÉCHAL, *Convex analysis and minimization algorithms I: Fundamentals*, vol. 305, Springer Science & Business Media, 2013. ISBN 978-3-662-02796-7.
- [8] M. KASS, A. WITKIN, AND D. TERZOPOULOS, *Snakes: Active contour models*, International Journal of Computer Vision, 1 (1988), pp. 321–331. <https://doi.org/10.1007/BF00133570>.
- [9] E. KERRIEN, A. YUREIDINI, J. DEQUIDT, C. DURIEZ, R. ANXIONNAT, AND S. COTIN, *Blood vessel modeling for interactive simulation of interventional neuroradiology procedures*, Medical Image Analysis, 35 (2017), pp. 685–698. <https://doi.org/10.1016/j.media.2016.10.003>.
- [10] C. MAGGIA, T. MISTRAL, S. DOYLE, F. FORBES, A. KRAINIK, D. GALANAUD, E. SCHMITT, S. KREMER, I. TROPÈS, E.L. BARBIER, J-F. PAYEN, AND M. DOJAT, *Traumatic Brain Lesion Quantification based on Mean Diffusivity Changes*, in Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, BrainLes (MICCAI), ed Crimi Aea, Springer International Publishing AG, 2018.
- [11] R. MALLADI, J.A. SETHIAN, AND B.C. VEMURI, *Shape modeling with front propagation: A level set approach*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (1995), pp. 158–175. <https://doi.org/10.1109/34.368173>.
- [12] A. MASTMEYER, M. WILMS, AND H. HANDELS, *Population-based respiratory 4D motion atlas construction and its application for VR simulations of liver punctures*, in Medical Imaging 2018: Image Processing, vol. 10574, International Society for Optics and Photonics, 2018, p. 1057417. <https://doi.org/10.1117/12.2293092>.
- [13] T. MCINEMEY AND D. TERZOPOULOS, *Topology adaptive deformable surfaces for medical image volume segmentation*, IEEE Transactions on Medical Imaging, 18 (1999), pp. 840–850. <https://doi.org/10.1109/42.811261>.
- [14] S. OSHER AND J.A. SETHIAN, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49. [https://doi.org/10.1016/0021-9991\(88\)90002-2](https://doi.org/10.1016/0021-9991(88)90002-2).
- [15] N. OTSU, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man, and Cybernetics, 9 (1979), pp. 62–66. <https://doi.org/10.1109/TSMC.1979.4310076>.
- [16] R. PINHO, V. DELMON, J. VANDEMEULEBROUCKE, S. RIT, AND D. SARRUT, *Keuhkot: a method for lung segmentation*, in International Workshop on Pulmonary Image Analysis, Toronto, Canada, 2011, p. 225–232. <https://hal.archives-ouvertes.fr/hal-01967304>.



- [17] W.H. PRESS, S.A. TEUKOLSKY, W.T. VETTERLING, AND B.P. FLANNERY, *Numerical recipes in C++: the art of scientific computing*, Cambridge University Press, 2002. ISBN 978-0521750332.
- [18] M. SEZGIN AND B. SANKUR, *Survey over image thresholding techniques and quantitative performance evaluation*, Journal of Electronic Imaging, 13 (2004), pp. 146–166. <https://doi.org/10.1117/1.1631315>.
- [19] D. TERZOPOULOS, A. WITKIN, AND M. KASS, *Constraints on deformable models: Recovering 3D shape and nonrigid motion*, Artificial Intelligence, 36 (1988), pp. 91–123. [https://doi.org/10.1016/0004-3702\(88\)90080-X](https://doi.org/10.1016/0004-3702(88)90080-X).
- [20] R. TRULLO, C. PETITJEAN, S. RUAN, B. DUBRAY, D. NIE, AND D. SHEN, *Segmentation of Organs at Risk in thoracic CT images using a SharpMask architecture and Conditional Random Fields*, in IEEE International Symposium on Biomedical Imaging (ISBI 2017), April 2017, pp. 1003–1006. <https://doi.org/10.1109/ISBI.2017.7950685>.
- [21] P-F. VILLARD, F.P. VIDAL, L. AP CENYDD, R. HOLBREY, S. PISHARODY, S. JOHNSON, A. BULPITT, N.W. JOHN, F. BELLO, AND D.A. GOULD, *Interventional radiology virtual simulator for liver biopsy*, International Journal of Computer Assisted Radiology and Surgery, (2013), pp. 1–13. <https://doi.org/10.1007/s11548-013-0929-0>.