

# Level Set Methods - Overview

Aliaksei Maistrou

Computer Aided Medical Procedures, TUM

*May 17, 2008*

# 1 Introduction to the Theory of Active Contours

An important class of segmentation methods is model based methods - “Active Contours”, also known as “Evolving Fronts”. Active contour is an interface usually used to separate structures and background on the image. There are two principal approaches to build an active contour:

- explicit or Lagrangian approach, and resulting interfaces called snakes;
- implicit or Eulerian approach, and resulting interfaces called level sets.

These approaches work identically for  $n$ -dimensional images. For simplicity of explanation and definition we will use  $2D$  case only.

## 1.1 Snakes Approach

Firstly introduced in [1] Snakes method took its name due to similarity of evaluating of the contour with a moving snake (fig.1.1). This contour has a parametric representation and the method is based on the idea of energy minimization.

Suppose we have a moving object in the  $2D$   $(x, y)$  plane. At each moment of time  $t$  this object will be at some point, whose coordinates we can write as  $(x(s), y(s))$ , i.e. the  $x$  and  $y$  coordinates at each point are given as functions of the parameter  $s$ , implying the curve  $\mathbf{C}$  itself could be presented in the form:

$$\mathbf{C}(s) : \begin{cases} x = x(s) \\ y = y(s) \end{cases} \quad (1.1)$$

where  $s$  is some parameter, which need not be time. This form of the curve, or the contour representation is called a parametric representation of the curve.

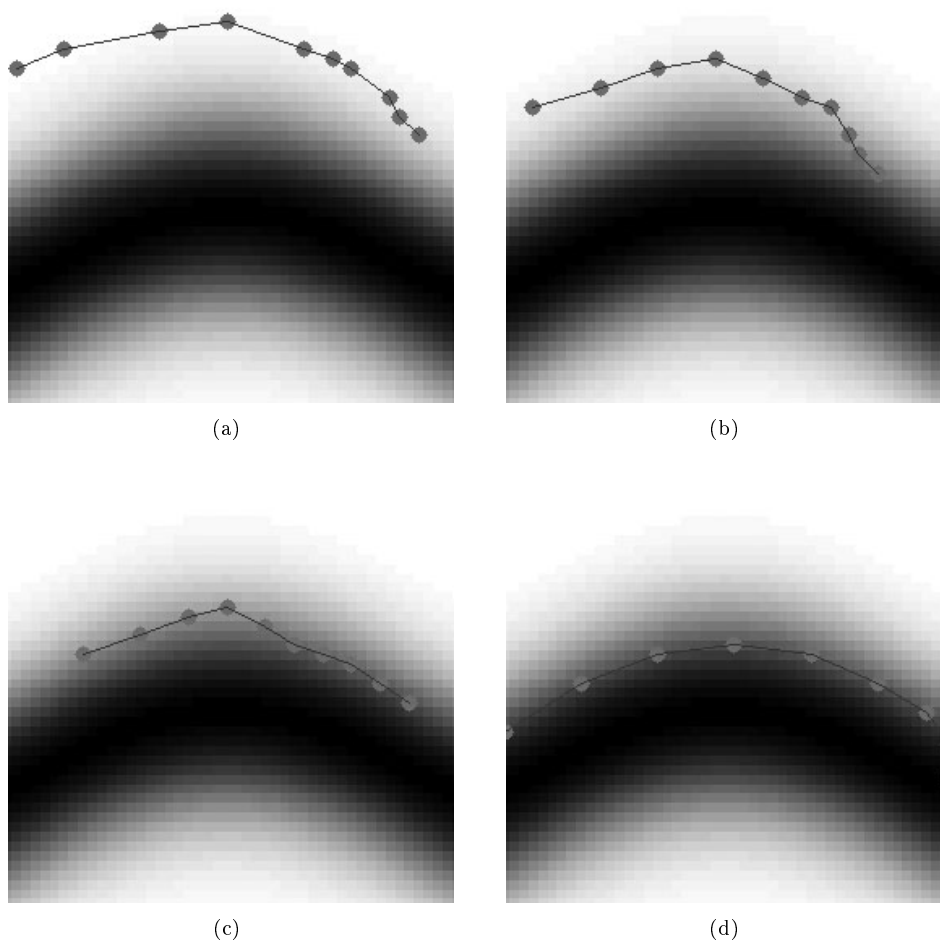
After proper rescaling of the functions  $x(s)$  and  $y(s)$ ,  $s$  will belong to interval  $[0, 1]$ . And we can rewrite (1.1) in general form as

$$\mathbf{C}(s) : [0, 1] \rightarrow \mathbf{R}^2, \quad (1.2)$$

where  $\mathbf{R}^2$  is given image domain and  $t$  is normalized parameter, which defines the current curve. Let  $I : \mathbf{R}^2 \rightarrow \mathbf{R}$  be an initial given image. We can see that parametric representation (1.2) does not depend on image itself anyhow. And it is possible to formulate the task of searching for the curve as energy minimization problem:

$$\mathbf{E}(\mathbf{C}) = \mathbf{E}_{int}(\mathbf{C}) + \mathbf{E}_{ext}(I(\mathbf{C})) \quad (1.3)$$

where  $\mathbf{E}_{int}$  represents internal energy or the smoothness of the curve, and  $\mathbf{E}_{ext}$  represents external energy, that is connected to the image intensity map.



**Figure 1.1:** Visualization of the Snake curve movement. Its position at start (a), in 5 (b), 15 (c) and 25 (d) iterative steps (<http://cobweb.ecn.purdue.edu/malcolm/interval/1995-017/>).

An example of expanded explicit formulation of the curve as argument, that gives a minimum of equation is presented below:

$$\mathbf{E}(C) = \int_0^1 \alpha |C'(s)|^2 + \beta |C''(s)| \, ds + \gamma \int_0^1 g(|\nabla I(C(s))|) \, ds \quad (1.4)$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are positive parameters,  $g(|\nabla I(C(t))|)$  is an edge-detector (like Sobel filter). The edge-detector for energy minimization task should be non-negative function  $g$  with the properties:

$$\begin{aligned} g'(z) &< 0 \\ \lim_{z \rightarrow \infty} g(z) &= 0 \end{aligned} \quad (1.5)$$

It ensures zero energy term only in case of infinitely large gradient. So, it is natural to use some hyperbolic monotonic function to satisfy (1.5):

$$g(|\nabla I(x, y)|) = \frac{1}{f(|\nabla I|)} = \frac{1}{1 + |D(G_\sigma(x, y) * I(x, y))|^p}, \quad p \geq 1 \quad (1.6)$$

where  $G_\sigma * I$  is a convolution of the discretized image  $I$  with the Gaussian kernel:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{|x^2+y^2|}{2\sigma^2}} \quad (1.7)$$

One of the simplest edge detector kernels  $D$  is Laplacian kernel. We also can provide some optimization to this sequence of two convolutions using precomputed analytical solution for the joint kernel ( $DG_\sigma$ ).

In general, searching for minimum of (1.4) is a functional minimization task, that could be solved using numerous numerical methods: steepest descent, conjugate steepest descent, simplex method, etc. Classical approaches, like steepest descent, involve calculation of energy function gradient, that corresponds to artificial forces: internal elastic force and external pressure force, that moves the curve. In reality, each step of the movement corresponds to one step in the gradient direction in the energy space. But Snakes transformation could indeed be directly associated with some kind of elastic balloon expansion or shrinking by external forces, since elastic force, caused to compensate pressure and other external forces, naturally changes the the balloon form the same way as we are trying to minimize curvature and external energy.

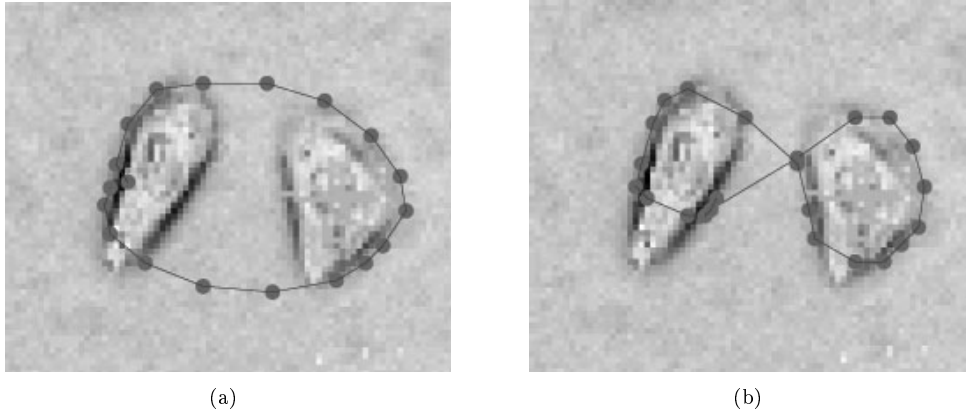
First integral in 1.4 increases when the value of curve curvature increases, the second integral increases when the gradient of an initial image intensity decreases. So, the minimum of (1.4) corresponds to the curve, that locates at the points of the maximum gradient, while the curve itself is maximally smooth.

Equation (1.4) is defined as a Lagrangian equation (that is why the Snakes also called the Lagrangian approach), and there is a straightforward way to solve it and find the curve. It is the main advantage of the snakes approach. The disadvantages of the method is numerous problems because of fixed topology of the explicit formulation, what means that contour could be only a little bit deformed circle. Mentioned problems could be better understand from the next figures:

- Fig.1.2(a) demonstrates inability to segment two objects correctly in one iteration;
- Fig.1.2(b) demonstrates self-intersection problems.

## 1.2 Level Set Approach

Another way of modeling active contours, that was initially presented by Osher and Sethian in [2], implements Level Set methods. Cumulative idea and main difference between the Snakes method and Level Set methods is that in all Level Set methods an active contour is presented implicitly through a value of scalar function.



**Figure 1.2:** Representation of the curve obtained using the Snakes method. Demonstration of the drawbacks (<http://www.mathworks.com/>)

### 1.2.1 Implicit curve representation

Similarities and differences between Level Set methods and the Snakes could be understood from simple example. Let's try to ask a PC to draw a circle on a  $2D$  surface, knowing the center of the circle and a radius. For this we need to give to our PC the array of points, which represents our circle, since there is no way for it to draw a pure analytical function. Assume for simplicity that the origin of the circle is the origin of coordinates.

One way to do it is to type explicit formulas for upper and lower arcs of the circle or write parametric representation:

$$y(x) = \pm\sqrt{R^2 - x^2}, \quad x \in [-R, R] \quad \text{or} \quad \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}, t \in [0, 2\pi) \quad (1.8)$$

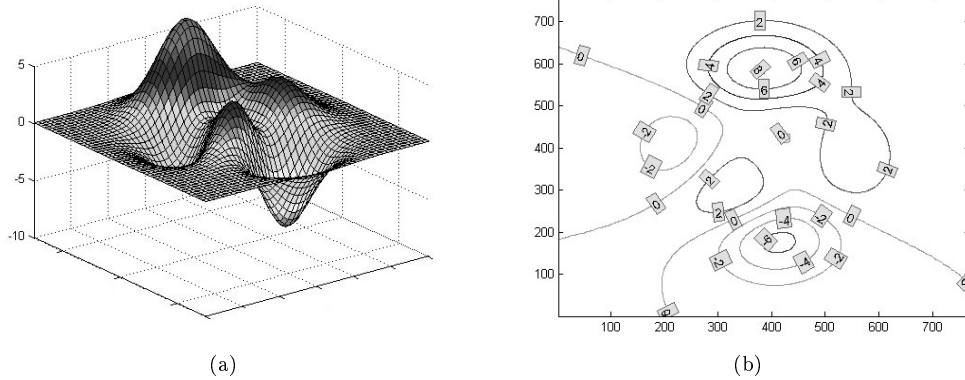
Then just ask the PC to run through the values of variable  $x$  or parameter  $t$  and to draw found points on a canvas. Other, a little bit tricky way to do it is to give to the PC another function of two variables:

$$\Phi(x, y) = x^2 + y^2 - R^2 \quad (1.9)$$

If we look more closely at the function  $\Phi$ , we will discern, that our points of the searching circle corresponds to values of variables  $x$  and  $y$ , which will make the function (1.9) equal to zero, and the function itself in each point  $(x, y)$  represents squared distance to the circle. It means, that we could implicitly find the circle when we know, even discrete set of values of the function (1.9): we could just build discrete set of values of  $\Phi$  (1.9) using two variables, then iterate through  $2D$  array of function values, choose ones, that are closest to zero and collect the array of coordinates, to which this closest to zero points corresponds, and define set of those points as the contour.

At the first look the second approach is much more sophisticated then the straightforward first one. But we could see, that the implicit approach does not need any kind of explicit representation of the curve - it could be used for arbitrary large number of circles and other geometrical figures

in arbitrary number of dimensions! Consider function  $\Phi(x, y)$  has distribution, presented on Fig.1.3(a). We could see that adding a constant factor to function  $\Phi(x, y)$  will result in changing topology (Fig.1.3(b)), that means a number of contours could vary. The only thing we should know to build figures using implicit representation is scalar values of the function, and the value, that will discrement our figures.



**Figure 1.3:** Changing in topology of the contour for  $\Phi(x, y)$  function (a) demonstrated by subtracting constant factors specified in square boxes (b) (from Matlab help)

Going back to the segmentation methods one can find direct links between the Snakes and the Level Set, and the Explicit and the Implicit representation of the circle in the last example: the Level Set method in 2D implies representation of the active contour implicitly through the two-dimensional function, what means that array of points, belonging to the active contour corresponds to the zero level set of the function  $\Phi(x, y)$  and the contour itself could be extracted from the  $\Phi$  similarly to previous example.

Strict mathematical formulation for  $C$  could be presented in the form:

$$C = \{(x, y) | \Phi(x, y) = 0\}, \tag{1.10}$$

where  $(x, y)$  - is coordinates of points, which corresponds to values  $\Phi(x, y) = 0$ ,  $\Phi$  is a Lipschitz function, usually called “Level Set Function”. Note, that  $C$  has no explicit dependencies on any variables.

We could see here that the areas inside and outside the contour  $C$  should have opposite signs of  $\Phi(x, y)$ , since there is no other way for function  $\Phi$  to get zero value on the contour (except some extreme cases of “touching” the zero level). Choosing the sign outside and inside the contour is arbitrary, but the majority of authors find more convenient to use the next properties:

$$\Phi(x, y) \begin{cases} < 0 : (x, y) \text{ is inside } C \\ = 0 : (x, y) \in C \\ > 0 : (x, y) \text{ is outside } C \end{cases} \tag{1.11}$$

### 1.2.2 Classical Derivation of the Level Set Function

Traditional approach [2] to construct and update the Level Set Function  $\Phi(x, y)$  uses artificial forces, which are moving the front in normal direction, the same way like in the Snakes method. This forces cause the contour  $C$  to move over an artificial time  $t$  with speed  $F$  in normal direction. Lets define  $(x_p(t), y_p(t))$  as the path of a little particle that belongs to the contour. Then, according to our definition of the Level Set Function we should ensure that in an arbitrary moment of time  $t$  our  $\Phi(t, x_p(t), y_p(t))$  function will be equal to zero:

$$\Phi(t, x_p(t), y_p(t)) = 0 \quad (1.12)$$

Other way we could ensure (1.12) for an arbitrary moment of time is to ensure it at the moment  $t = 0$ :  $\{\Phi(0, x_p(0), y_p(0)) = \Phi(0, \vec{x}_p(0)) = 0\}$  and ensure no further changes of  $\Phi$  in time domain, implying system:

$$\begin{aligned} \Phi(0, \vec{x}_p(0)) = 0 \\ \frac{d\Phi}{dt}(t, \vec{x}_p(t)) = 0 \end{aligned} \Leftrightarrow \begin{aligned} \Phi(0, \vec{x}_p(0)) = 0 \\ \frac{\partial\Phi}{\partial t}(t, \vec{x}_p(t)) + \nabla\Phi(t, \vec{x}_p(t)) \cdot \frac{d\vec{x}_p(t)}{dt} = 0 \end{aligned} \quad (1.13)$$

This formulation could be extended on the whole *contour* domain:

$$\begin{aligned} \Phi(0, \vec{x}(0)) = \Phi_0(x, y) \\ \Phi_t(t, \vec{x}(t)) + \nabla\Phi(t, \vec{x}(t)) \cdot \frac{d\vec{x}(t)}{dt} = 0 \quad \vec{x}(t) \in C(t) \end{aligned} \quad (1.14)$$

where  $\Phi_0(x, y)$  is the initial Level Set Function with the initial contour defined as  $(x, y)|\Phi_0(x, y) = 0$ .

Note, that we are interested in non-tangential movements of the curve, since tangential movements does not change a form of the curve. Unit vector in normal direction at the contour's point  $(\vec{x}_p)$  could be calculated as:

$$\vec{n} = \frac{\nabla\Phi}{\|\nabla\Phi\|} \quad (1.15)$$

where  $\|\nabla\Phi\|$  is the Euclidean norm. Now we can introduce a speed function  $F$ , which exactly corresponds to the movement of our contour in normal direction with respect to artificial time  $t$ :

$$F = \frac{d\vec{x}(t)}{dt} \cdot \vec{n} \Leftrightarrow F\|\nabla\Phi\| = \frac{d\vec{x}(t)}{dt} \cdot \nabla\Phi(t, \vec{x}(t)) \quad \vec{x}(t) \in C(t) \quad (1.16)$$

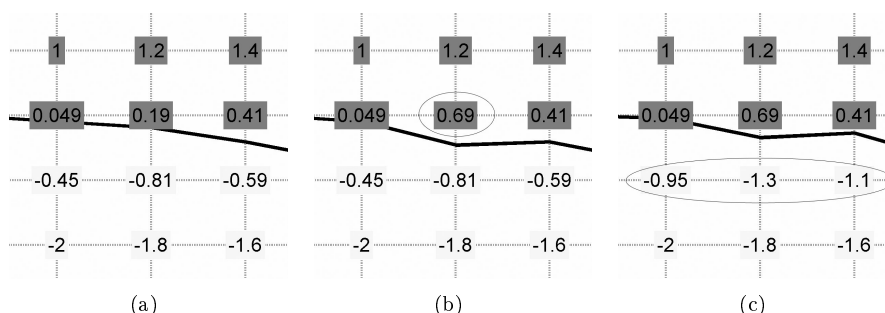
Substituting (1.16) into (1.14) yields the system:

$$\begin{aligned} \Phi(0, \vec{x}(0)) = \Phi_0(x, y) \\ \Phi_t(t, \vec{x}(t)) = -F\|\nabla\Phi\| \quad \vec{x}(t) \in C(t) \end{aligned} \quad (1.17)$$

So, in the traditional approach, evaluation of the Level Set Function  $\Phi(t, \vec{x})$  could be achieved solving initial value problem (1.17). After discretizing the time derivative in this partial differential equation we could achieve simple iterative solution scheme:

$$\begin{aligned}\Phi(0, \vec{x}(0)) &= \Phi_0(x, y) \\ \Phi(t + \Delta t, \vec{x}(t)) &\cong \Phi(t, \vec{x}(t)) - \Delta t \cdot F \|\nabla \Phi\|\end{aligned}\quad (1.18)$$

According to (1.18) after the initial setting of the function  $\Phi$  its update is done, and afterward we need to extract the contour back, as it is now presented implicitly by the zero level set of  $\Phi$ . Since we have the signed function  $\Phi$  (1.11), contour extraction technique could be improved if we use marching cubes algorithm [3]. This algorithm in  $2D$  case introduces the straightforward fast technique for extraction a boundary on discrete grid using grid points' values as weights. Fig.1.4 demonstrates how marching cubes algorithm extract the contour and how contour evaluates with respect to changes of grid point values.



**Figure 1.4:** Changing position of the contour  $C$  with respect to changing in the Level Set Function  $\Phi$

### 1.2.3 Expansions for practical implementation

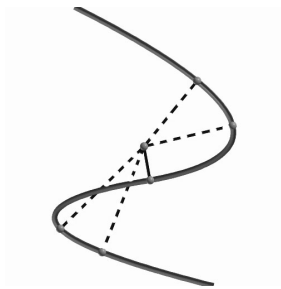
Equation (1.18) could not be practically implemented until we do not know equations for functions  $\Phi$  and  $F$ . Some directions for its representations will be presented below.

Firstly let's define the Level Set function. There are a lot of optimization algorithms, that embed special properties to the Level Set Function to not let the algorithm to diverge or to simplify its re-initializations. The most usual and the easiest to implement and explain Level Set function is defined similarly to (1.9) as the distance  $d(x, y)$  in the Euclidean metric. Taking into account (1.11) we could refine it:

$$\Phi(x, y, t) = \begin{cases} = -d(x, y, t) : (x, y) \text{ is inside } C(t) \\ = 0 : (x, y) \in C(t) \\ = d(x, y, t) : (x, y) \text{ is outside } C(t) \end{cases} \quad (1.19)$$

Here the distance, as usual for a point-to-curve configuration, means the shortest distance between the point with coordinates  $(x, y)$  and the set of curve points as shown in fig.1.5.

Calculation of the speed function  $F$  is more complicated task then choosing  $\Phi$  function. The simplest approach was firstly introduced in fluid dynamics where the speed function  $F$  was chosen to be equal to real velocity of a fluid. An example of artificial approach of constructing  $F$  if real



**Figure 1.5:** Demonstration of a point-to-curve distance: the shortest distance to curve points

equivalent is not available, is like geometric active contour model [4]. The model takes into account the mean curvature  $\kappa$  of the contour as well as gradient information. It means that the more curved contour is, the faster it moves and it stops moving where gradient of the image is high enough. Mathematically mean curvature could be represented as:

$$\kappa = \operatorname{div}\left(\frac{\nabla\Phi(x, y)}{\|\nabla\Phi(x, y)\|}\right), \quad (1.20)$$

where  $\operatorname{div}(\cdot)$  defines divergence of a vector field. For a contour in  $2D$  space  $K$  could be rewritten in discretizable form:

$$\kappa = \operatorname{div}\left(\frac{\nabla\Phi(x, y)}{\|\nabla\Phi(x, y)\|}\right) = \frac{\Phi_{xx}\Phi_y^2 + \Phi_x^2\Phi_{yy} - 2\Phi_x\Phi_y\Phi_{xy}}{\|\nabla\Phi\|^3}, \quad (1.21)$$

where  $\Phi_x$ ,  $\Phi_y$  define partial derivatives in  $x$  and  $y$  directions, and  $\Phi_{xx}$ ,  $\Phi_{yy}$  define second order partial derivatives. Detailed explanation and comparison of methods for construction of  $F$  presented in [5].

Finally, (1.17) could be rewritten in general form for a full image domain and be numerically treated as was demonstrated previously:

$$\begin{aligned} \Phi(0, \vec{x}(0)) &= \Phi_0(x, y) \\ \Phi_t(t, \vec{x}(t)) &= g(|\nabla I|)\|\nabla\Phi\|(\kappa + \nu) \end{aligned} \quad (1.22)$$

where  $g(|\nabla I|)$  defines edge-detector function, which is positive in homogeneous regions, and zero at edges,  $\nu \geq 0$  is a constant. It is convenient to use the same edge-detector (1.6) that was used for the Snakes method (see 1.6);

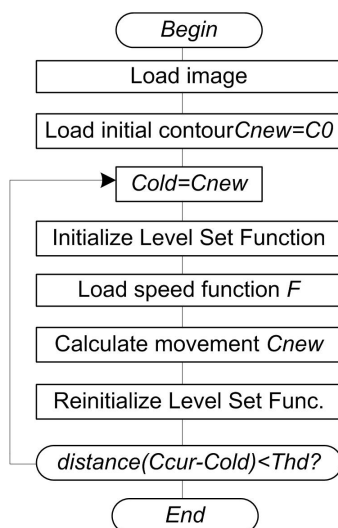
#### 1.2.4 Algorithm for classical Level Set Method implementation

Collecting all things together the straightforward way of segmentation using the Level Set method could be represented in the following algorithm stages:

1. Load discrete image  $I(x, y)$ ;
2. Interactively define the initial contour (using interpolation techniques between pointed by user image pixels);

3. Initialize  $\Phi_0(x, y)$  function on the image  $I(x, y)$  domain as the distance function (1.19);
4. Choose the speed function  $F$ .
5. Calculate movement of the contour  $C$  in one discrete artificial time step using equation (1.18);
6. Reinitialize/modify Level Set function;
7. Check the stop condition, which is usually calculated as distance between current and previous positions of the contour  $C$ . If the stop condition is not satisfied, go to item 5.

Corresponding block-scheme of the algorithm (taking into account remarks on each item) presented in fig.1.6.



**Figure 1.6:** Block-scheme of the algorithm for the classical Level Set Method implementation

### 1.3 Variational Level Set Methods

Summarize everything presented previously about level set methods we could see that we transformed equations of the Snakes motions to the implicit formulation. Next improvement is based on embedding the Level Set function itself into energy functional  $E$ . This approach allows us to define an energy components also at points not lying on the front, yielding straightforward energy minimization algorithm for implicit representation of the contour.

For further considerations we need to define the Heaviside function  $H$ :

$$H(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases} \quad (1.23)$$

and the Dirac delta distribution  $\delta$ :

$$\delta(z) = \frac{d}{dz}H(z) \quad (1.24)$$

Assume the energy density function:

$$f : \begin{cases} \mathbf{R}^n \times \mathbf{R} \times \mathbf{R}^n \rightarrow \mathbf{R} \\ (\mathbf{x}, \Phi, \nabla\Phi) \mapsto f(\mathbf{x}, \Phi, \nabla\Phi) \end{cases} \quad (1.25)$$

where  $\mathbf{x}, \Phi, \nabla\Phi$  define argument values of  $f$ , not embedded functions. Now we represent the energy functional, depending on  $f$ :

$$E(\Phi) = \int_{\Omega} f(\mathbf{x}, \Phi(x), \nabla\Phi(x)) dx, \quad (1.26)$$

We could represent the integral over the contour  $C$  separately using the Dirac distribution:

$$E_C(\Phi) = \int_C f dx = \int_{\Omega} \delta(\Phi) \cdot \|\nabla\Phi\| \cdot f dx \quad (1.27)$$

For regions outside and inside the contour, the energy integrals could be modeled using the Heaviside function:

$$E_{in}(\Phi) = \int_{\Phi < 0} f dx = \int_{\Omega} (1 - H(\Phi)) \cdot f dx, \quad (1.28)$$

$$E_{out}(\Phi) = \int_{\Phi > 0} f dx = \int_{\Omega} H(\Phi) \cdot f dx, \quad (1.29)$$

Collapsing energy terms of equations (1.27), (1.28) and (1.29) in (1.26) we complete separate energy representation:

$$E(\Phi) = E_C(\Phi) + E_{in}(\Phi) + E_{out}(\Phi), \quad (1.30)$$

Minimization of the functional using a variational approach requires calculus of variations: to compute extremum of the function we need to compute its derivative, in our case a functional derivative, called also ‘‘Gâteaux’’ or ‘‘Fréchet’’ derivative, that means calculating changes in the functional in direction of  $\Phi$  function:

$$\frac{\delta E}{\delta \Phi} = \int_{\Omega} f_{\Phi} \cdot \eta(\mathbf{x}) dx + \int_{\Omega} \langle f_{\nabla\Phi}, \nabla\eta(\mathbf{x}) \rangle dx \quad (1.31)$$

### 1.3.1 Example of Practical Implementation of Variational Level Set Method

In [6] practical implementation of variational approach for level set method for two-dimensional case is presented. The idea is to go to image domain optimization directly using (1.28) and (1.29) expansions:

$$E_{out}(\Phi) = \int_{\Omega} |I(x, y) - c_1|^2 H(\Phi) dx dy, \quad (1.32)$$

$$E_{in}(\Phi) = \int_{\Omega} |I(x, y) - c_2|^2 (1 - H(\Phi)) dx dy, \quad (1.33)$$

where  $c_1$  and  $c_2$  assumed outside and inside intensity values, respectively. Rough estimation of  $c_1$  and  $c_2$  uses average values of intensities over the domains. Energy component over the curve domain (1.27) does not change representation at all:

$$E_C(\Phi) = \int_C f dx = \int_{\Omega} \delta(\Phi) |\nabla \Phi(x, y)| dx dy \quad (1.34)$$

Summarizing everything together similar to (1.30) we come up with the functional:

$$\begin{aligned} E_s(c_1, c_2, \Phi) = & \mu \int_{\Omega} \delta(\Phi) |\nabla \Phi(x, y)| dx dy + \\ & + \lambda_1 \int_{\Omega} |I(x, y) - c_1|^2 H(\Phi) \cdot dx dy + \\ & + \lambda_2 \int_{\Omega} |I(x, y) - c_2|^2 (1 - H(\Phi)) dx dy \end{aligned} \quad (1.35)$$

If we assume  $c_1$  and  $c_2$  fixed, minimization of  $E_s$  is equivalent to associated Euler-Lagrange partial differential equation with artificial time  $t \geq 0$  for  $\Phi$ :

$$\begin{cases} \Phi(0, x, y) = \Phi_0(x, y) \\ \frac{\partial \Phi}{\partial t} = \delta(\Phi) (\mu \operatorname{div}(\frac{\nabla \Phi}{|\nabla \Phi|} - \lambda_1 (I - c_1)^2 - \lambda_2 (I - c_2)^2) = 0 \\ \frac{\delta(\Phi)}{|\nabla \Phi|} \frac{\partial \Phi}{\partial \bar{n}} = 0 \end{cases} \quad (1.36)$$

where  $\mu$ ,  $\lambda_1$  and  $\lambda_2$  are weight coefficients, in practice  $\lambda_1 = \lambda_2 = 1$  are usually set, but  $\mu$  has a scaling role and varies in experiments.

Algorithm for this variational Level Set method can be summarize as follows:

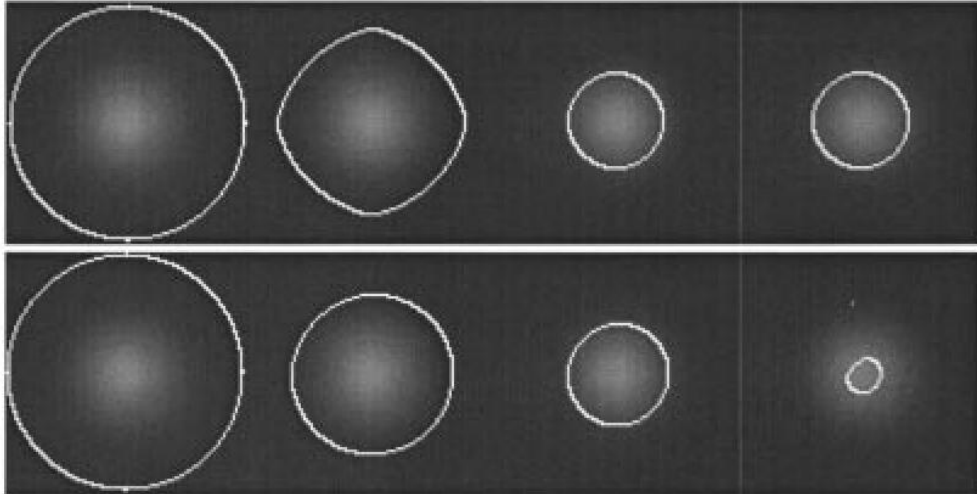
1. Load discrete image  $I(x, y)$ ;
2. Initialize  $\Phi_0(x, y)$  function on the image  $I(x, y)$  domain as distance equation (1.19);
3. Solve the PDE (1.36) and get new  $\Phi$ ;
4. Check stop condition. Check if the solution is stationary. If not satisfied, go back to item 3.

Outstanding feature of presented algorithm is its intensity based functional minimization core. It results in a list of advantages:

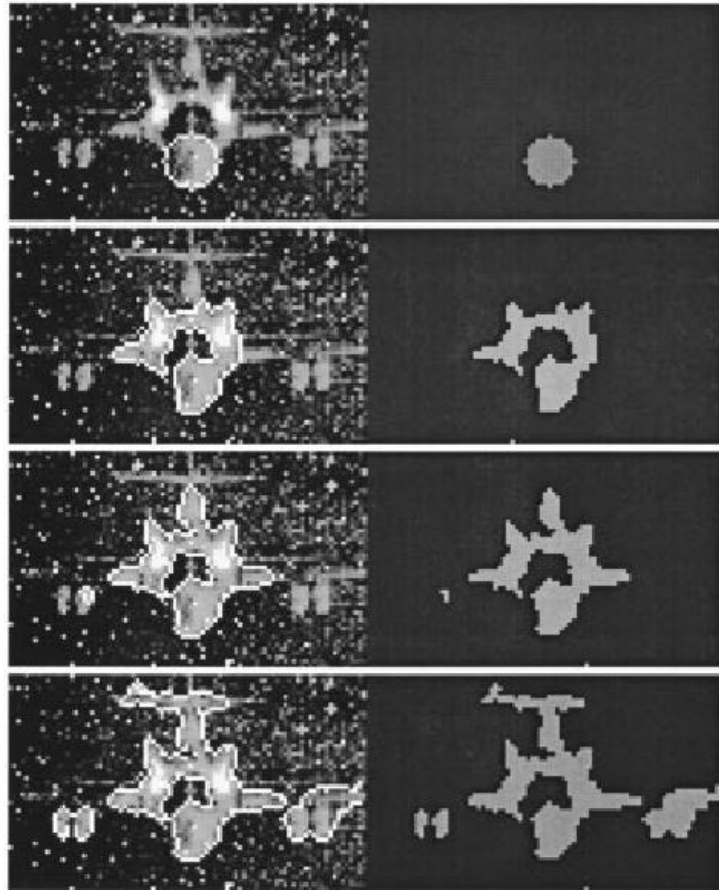
## 1 Introduction to the Theory of Active Contours

1. The model is not based on an edge function;
2. No need in smoothing of an initial image;
3. Works good with very noisy images;
4. Works good with very smooth boundaries;
5. Position of the initial contour has no influence on the resulting segmentation.

Verification of the method was extensively powered by authors. The key mentioned advantages is presented on fig.1.7 and fig.1.8.



**Figure 1.7:** Detecting object with the very smooth contour. Top: using variational approach. Bottom: classical model. Reprinted from [6].



**Figure 1.8:** Detecting of the contours of a plane from real noisy image. Reprinted from [6].

## Bibliography

- [1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *Int'l J. Comp. Vis.*, 1(4):321–331, 1988.
- [2] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Computational Physics*, 79(1):12–49, 1988.
- [3] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface reconstruction algorithm. In *Proc. SIGGRAPH*, volume 21, pages 163–169. ACM, 1987.
- [4] Vicent Caselles, Francine Catté, Tomeu Coll, and Françoise Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1):1–31, 1993.
- [5] David Adalsteinsson and James A. Sethian. The fast construction of extension velocities in level set methods. *J. Computational Physics*, 148(1):2–22, 1999.
- [6] Tony F. Chan and Luminita A. Vese. Active contours without edges. *IEEE Trans. Image Process.*, 10(2):266–277, 2001.