Introduction to Medical Imaging 046831, Spring 2010

Sobolev Active Contours Without Edges

Lior Deutsch, 200700987. Marina Kokotov, 305891350.

1. Gradient Based Energies

The active contours framework is a method used for detecting an object in a given image. The idea is to evolve a curve until it reaches the boundary of the object. The curve evolves so as to minimize an energy which depends on the image and on the curve, and which attains its minimal value when the curve is located on the boundary of the object.

A common way of defining an energy which holds the latter property is by making it depend on the gradient of the image, which usually has a high absolute value on boundaries and edges in the image. An example is given by the energy

$$E_1(C) = \alpha \int_0^1 |C'(s)|^2 \,\mathrm{d}s + \beta \int_0^1 |C''(s)|^2 \,\mathrm{d}s + \lambda \int_0^1 g\left(\nabla u_0(C(s))\right)\mathrm{d}s\,, \tag{1}$$

where $C:[0,1]\to\mathbb{R}^2$ is a parameterized curve; $C(s)$, $C'(s)$ and $C''(s)$ are respectively the curve, its first derivative and its second derivative evaluated at the parameter s; u_0 is the image; $\nabla u_0(C(s))$ is the gradient of the image evaluated at the point $C(s)$; α, β and λ are positive parameters; and g is a positive function of ∇u_0 which decreases as a function of $|\nabla u_0|$ and satisfies:

$$\lim_{|z|\to\infty} g(z) = 0\,. \tag{2}$$

g acts as an edge-detector, causing the energy to be small when the curve C is located on boundaries. For instance,

$$g\left(\nabla u_0\right) = \frac{1}{1+\left|G_\sigma * \nabla u_0\right|^p}, \quad p \ge 1 \tag{3}$$

where $G_\sigma * \nabla u_0$, a smoother version of ∇u_0, is the convolution of the image $\nabla u_0(x,y)$ with the Gaussian $G_\sigma(x,y)$.

The first two terms in the energy $E_1(C)$ control the smoothness of the contour and do not depend on the image u_0. Therefore they are referred to as the *internal energy*. The third term, as mentioned above, attracts the contour towards the object in the image and is referred to as the *external energy*.

The evolution of the active contour starts from an initial curve which iteratively changes its position and shape so as to minimize the energy. The evolution can be determined by a greedy search, dynamic programming, gradient descent or other energy minimization schemes.

An image-gradient dependant energy can give rise to object detection only if the object's boundary is defined by a gradient, i.e. there is a sharp change in the image's value when crossing the

boundary. However, due to the discrete representation of images, in practice the gradients are bounded, therefore the edge detector is never zero on edges, and the curve might pass through the boundary. Furthermore, some objects do not have a boundary – their image values vary smoothly between the interior and the exterior of the object. Another limitation that a gradient based energy has is when segmenting a very noisy image. Noisy pixels might introduce a high gradient, and smoothing the image (isotropically) will smooth the edges too.

2. The Chan-Vese Energy

The limitations mentioned in the last paragraph call for a different definition of the energy which does not depend on the gradient of the image. The Chan-Vese energy is supposed to cope also with objects that do not have edges in a noisy image. It is defined as follows:

$$E_{\mathrm{CV}}(C) = \int_{\mathrm{inside}(C)} \left|u_0(x,y) - c_{\mathrm{in}}(C,u_0)\right|^2 \mathrm{d}x\mathrm{d}y + \int_{\mathrm{outside}(C)} \left|u_0(x,y) - c_{\mathrm{out}}(C,u_0)\right|^2 \mathrm{d}x\mathrm{d}y \,, \tag{4}$$

where $\mathrm{inside}(C)$ is the region in $\mathbb{R}^2$ enclosed by the contour C; $\mathrm{outside}(C)$ is the exterior of the region $\mathrm{inside}(C)$ (if the domain of the image is the bounded subset Ω of $\mathbb{R}^2$, then $\mathrm{outside}(C)$ is $\Omega \setminus \mathrm{inside}(C)$); c_{in} and c_{out} are the averages of u_0 inside C and respectively outside C.

Fig. 1(a) displays a simple case of an image which is formed by two regions, each with a constant intensity. The object's intensity is u_0^{in} and the background's intensity is u_0^{out}. When C lies exactly on the boundary of the object, as in fig. 1(b), we have $c_{\mathrm{in}} = u_0^{\mathrm{in}}$ and $c_{\mathrm{out}} = u_0^{\mathrm{out}}$, thus $E_{\mathrm{CV}}(C) = 0$. It is easy to see that in any other configuration, such as those in fig. 1(c) and fig. 1(d), we have $E_{\mathrm{CV}} > 0$. This example illustrates the basic idea of the Chan-Vese energy.

(a) (b) (c) (d)

Fig. 1

Additional regularizing terms, such as length of the curve C and the area of the region inside C, can be added to the energy $E_{\mathrm{CV}}(C)$. However, in our implementation we will just use the energy defined in (4). The regularization of the curve's evolution will be achieved via the Sobolev inner product, as will be explained later.

3. The Gradient of the Chan-Vese Energy

In this project, we will use the gradient descent optimization algorithm for minimizing the energy $E_{\mathrm{CV}}(C)$. The algorithm requires setting an initial contour, which then evolves in each step in

the opposite direction of the gradient of the energy. By "direction" we mean a vector in the tangent space of the manifold of smooth curves, at the curve C. The evolution in that direction may involve any translation and deformation of the curve.

The form of the gradient of $E_{\mathrm{CV}}(C)$ depends on the inner product defined on the tangent space of the manifold of curves. Let $\langle ,\rangle_C$ be the inner product of tangent vectors at curve C. Then the gradient $\nabla E(C)$ of the real valued function $E(C)$ is the unique vector in the tangent space of curve C that satisfies

$$\langle h, \nabla E(C)\rangle_C = \frac{\mathrm{d}}{\mathrm{d}t} E(C+th)\Big|_{t=0} \tag{5}$$

for all h that belong to the tangent space of curve C. In other words, the inner product of h and $\nabla E(C)$ equals the directional derivative at C in the direction h of the function E.

Usually, the L^2 inner product is used:

$$\langle h, k\rangle_{L^2} = \frac{1}{L}\int_C h(s)\cdot k(s)\,\mathrm{d}s\,, \tag{6}$$

where h and k are tangent to C; L is the length of C; and the parameterization is by the arclength. Alternatively, the Sobolev inner product can be used:

$$\langle h, k\rangle_{\mathrm{Sobolev}} = \left(\frac{1}{L}\int_C h(s)\,\mathrm{d}s\right)\left(\frac{1}{L}\int_C k(s)\,\mathrm{d}s\right) + \lambda L^2 \langle h', k'\rangle_{L^2}\,, \tag{7}$$

where λ is a positive number; h' denotes the derivative of h with respect to arclength; and the contour integration $\int_C$ is equivalent to $\int_0^L$. (This is just a simple case out of the family of all Sobolev inner products). Later we will explain the advantages of the Sobolev inner product over the L^2 inner product. Now we shall find an expression for the gradient of the Chan-Vese energy using each of the inner products (6) and (7). Let $E(C) = \int_{\mathrm{inside}(C)} \varphi(x,y)\,\mathrm{d}x\mathrm{d}y$ be a region-based energy. Then, using differentiation under the integral sign (Reynolds transport theorem):

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}E(C+th) &= \frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathrm{inside}(C+th)} \varphi(x,y)\,\mathrm{d}x\mathrm{d}y = \\ &= \int_{\mathrm{inside}(C+th)} \frac{\partial\varphi}{\partial t}\,\mathrm{d}x\mathrm{d}y - \int_{C+th} \varphi\frac{\partial(C+th)}{\partial t}\cdot\mathcal{N}\,\mathrm{d}s = \\ &= -\int_{C+th} \varphi\,\mathcal{N}\cdot h\,\mathrm{d}s\,,\end{aligned} \tag{8}$$

where $\mathcal{N}$ is the inwards normal. It follows that

$$\frac{\mathrm{d}}{\mathrm{d}t}E(C+th)\Big|_{t=0} = -\int_C \varphi\mathcal{N}\cdot h\,\mathrm{d}s\,. \tag{9}$$

According to (5), the gradient $\nabla_{L^2}E(C)$ induced by the L^2 inner product (6) satisfies for all h

$$\frac{1}{L}\int_C \nabla_{L^2} E(C)\cdot h\,\mathrm{d}s = \frac{\mathrm{d}}{\mathrm{d}t}E(C+th)\bigg|_{t=0} = -\int_C \varphi\mathcal{N}\cdot h\,\mathrm{d}s\,. \tag{10}$$

Since the equality should hold for all h, it follows that

$$\nabla_{L^2} E(C) = -L\varphi\mathcal{N}\,. \tag{11}$$

It is easy to see that for Chan-Vese

$$\varphi = \left(u_0 - c_{\text{in}}\right)^2 - \left(u_0 - c_{\text{out}}\right)^2 = 2\left(c_{\text{in}} - c_{\text{out}}\right)\left(\frac{c_{\text{in}} + c_{\text{out}}}{2} - u_0\right), \tag{12}$$

therefore the L^2 gradient of the Chan-Vese energy is

$$\nabla_{L^2} E(s) = -2L\left(c_{\text{in}} - c_{\text{out}}\right)\left(\frac{c_{\text{in}} + c_{\text{out}}}{2} - u_0(s)\right)\mathcal{N}\,. \tag{13}$$

From the definition of the gradient (5), it follows that for all h in the tangent space at curve C

$$\left\langle h, \nabla_{L^2} E\right\rangle_{L^2} = \left\langle h, \nabla_{\text{Sobolev}} E\right\rangle_{\text{Sobolev}}\,. \tag{14}$$

Using the definitions (6) and (7), integration by parts and periodic boundary conditions (as we are dealing with closed smooth curves) it is easy to see that

$$\left\langle h, \nabla_{\text{Sobolev}} E\right\rangle_{\text{Sobolev}} = \left\langle h, \frac{1}{L}\int_C \nabla_{\text{Sobolev}} E\mathrm{d}s - \lambda L^2 \frac{\mathrm{d}^2}{\mathrm{d}s^2}\nabla_{\text{Sobolev}} E\right\rangle_{L^2}, \tag{15}$$

therefore, from (14), the gradients satisfy the equation

$$\nabla_{L^2} E = \frac{1}{L}\int_C \nabla_{\text{Sobolev}} E\mathrm{d}s - \lambda L^2 \frac{\mathrm{d}^2}{\mathrm{d}s^2}\nabla_{\text{Sobolev}} E\,. \tag{16}$$

It can be shown that the solution to the equation, with periodic boundary conditions, is

$$\nabla_{\text{Sobolev}} E(s) = \int_C K_\lambda(s-s)\nabla_{L^2} E(s')ds' = \left(K_\lambda * \nabla_{L^2} E\right)(s)\,, \tag{17}$$

where the convolution is done using the periodic extension of the kernel

$$K_\lambda(s) = \frac{1}{L}\left(1 + \frac{(s/L)^2 - (s/L) + 1/6}{2\lambda}\right), \quad s\in[0,L]\,. \tag{18}$$

In the gradient descent algorithm, the "force" experienced by the curve in each iteration is (up to a positive normalizing constant) the negative of the gradient:

$$F_{L^2}(s) = \left(c_{\text{in}} - c_{\text{out}}\right)\left(\frac{c_{\text{in}} + c_{\text{out}}}{2} - u_0(s)\right)\mathcal{N} \tag{19}$$

$$F_{\text{Sobolev}}(s) = \left(K_\lambda * F_{L^2}\right)(s) \tag{20}$$

4. Advantages of the Sobolev Force over the L^2 Force

From (19) it can be seen that the force F_{L^2} displaces each point on the contour in the normal direction either inwards or outwards. The exact orientation – inwards or outwards – depends on whether the value of the image u_0 at the point is higher or lower than the average of c_{in} and c_{out}. Although this seems like a desirable property, it has the problem of sensitivity to noise: If the curve lies

on a noisy pixel, the force might move the corresponding point on the snake in the wrong direction. As a result, the curve could become non-smooth. A possible solution to this problem is to add regularizing terms to the energy (4), but then the local minima of the energy will not have the exact desired property as a boundary detector.

The force F_{Sobolev}, given in (20), is less sensitive to noise, as it is a result of smoothing the force F_{L^2} by the smoothing low-pass filter K_λ. The convolution with the kernel K_λ makes the force F_{Sobolev} at each point of the curve not depend only on the value of u_0 at that point (given c_{in} and c_{out}), thus the effect of a noisy pixel is less significant. This non-local behavior of F_{Sobolev} as compared to F_{L^2} is a result of the dependency of the inner product (7) on the derivatives of h. This dependency acts as a regularizing factor on the evolution of the curve, preventing it from becoming non-smooth. Therefore there is no need for regularity terms in the energy.

The local minimizers of the energy when using the L^2 inner product are also local minimizers when using the Sobolev inner product. However, the definitions of locality are different in each case. An initial contour might have in the L^2 space a close local minimum that is not close to the same initial contour in the Sobolev space. In the Sobolev space there is a much larger distance between a smooth curve and a noisy non-smooth deformation of it. Thus, the evolution of the Sobolev active contour is less likely to be attracted to undesirable local minima. K_λ, being a low-pass filter, attenuates in (20) higher frequencies of F_{L^2}, thus causing the evolution of the curve to prefer global over fine deformations. By "global" deformations we mean motions that do not greatly change the shape of the curve, such as translations. The DC frequency of the force, for example, represents a force component which is constant as a function of the parameter s, thus representing a translation. The higher frequencies represent force components which are less smooth, thus less global. The fact that the Sobolev force prefers global motions, as compared to the L^2 force, causes the Sobolev active contour first to be attracted to the neighborhood of the object being detected, and then to deform to the object's shape. In this form of evolution, the active contour bypasses undesirable minimizers which might be in the locality when using the L^2 active contour.

5. Implementation of the Chan-Vese Sobolev Active Contour

(Remark: In this section, underlined words refer to parameters in the Matlab code which are user-controlled by changing their values at the beginning of the .m file).

The implementation of the object detection algorithm was done in the Matlab environment. The algorithm uses the method of steepest descent for minimizing the Chan-Vese energy, thus it requires an initial contour to be specified. The specification of the initial contour is done manually by the user, using the function `roipoly`. The function returns the vertices of a polygon, into the variable `snake`. The vertices are typically very sparse, and if we want to approximate a continuous curve we must add vertices. The sub-routine `addpoint` does that by iteratively adding vertices to `snake` (to the centers

of the polygon's longest sides) until the distance between each successive pair of vertices is smaller than `max dist` and larger than `min dist`. Next, the initial contour is smoothed by a moving average filter whose width is `smooth length` times the number of vertices. If `smooth_length` is taken to be 0, no smoothing is done. A problem with the moving average filter is that the curve also undergoes shrinkage – a better solution would be to use a smoothing algorithm without shrinkage.

Now that the initial contour has been specified, the gradient descent iterations start. A total of `steps` iterations are done. In each iteration, the region `BW` bounded by `snake` is calculated using, again, `roipoly`. The averages of the image in `BW` and outside of `BW`, c_{in} and c_{out}, are calculated. The calculation of the average c_{out} takes into account only the pixels which are closer than `out size` to `BW`. If `out_size` is set to the string `'all'`, than c_{out} is calculated based on the whole image (except `BW`).

Next, the arclength parameterization is calculated.

The normal to the contour at each vertex is taken as the unit normal to straight line connecting the previous vertex with the next vertex. The result of this calculation is a continuous normal that points either outwards or inwards. A disruption which occurs during some evolutions is the self-intersection of the contour. To fix self-intersections, we first make the assumption that most of the contour is not part of a "loop". When we did the first step of calculating the normal the result was a continuous normal. This means that at some vertices the normal pointed outwards and at others it pointed inwards. Under the assumption we are making, the smaller of the two sets {vertices with outward pointing normal} and {vertices with inward pointing normal} consists of the vertices which are part of a "loop". Thus, these vertices are deleted and instead new vertices are added using `addpoints`. To make the normal point inwards we check the value of `BW` at the neighboring pixel pointed by the normal at each vertex. If `BW` is 1, then the direction is correct and if `BW` is 0 then we negate the direction.

The force F_{L^2} depends on the image u_0, as seen in (19). We have a sampled image, therefore the values should be determined by interpolation. We use nearest-neighbor interpolation: Each point on the curve is rounded to the nearest pixel.

The parameter λ of the kernel function K_λ is `lambda`. After the kernel has been computed, the force F_{Sobolev} is found by a cyclic convolution of K_λ with F_{L^2}. The resulting force should be normalized, so that the size of the step of each gradient descent iteration, `dt`, will be constant. There are at least two ways this can be done. The first is to multiply the force by a normalization factor $\alpha > 0$ such that the maximum displacement that a vertex would undergo would be `dt`, i.e.

$$\max_s \alpha \cdot F_{\text{Sobolev}}(s) = 1 . \tag{21}$$

The normalization that we chose is to multiply the force by a factor $\alpha > 0$ such that

$$\alpha \cdot \frac{1}{L} \int_0^L F_{\text{Sobolev}}(s)\mathrm{d}s = 1 . \tag{22}$$

The normalization done, we update the vertices of the contour by adding to them the force times `dt`.

Here is a summary of the parameters of the Matlab code:

Parameter	Possible Values	Description
`file_name`	string	Name of image file.
`load_initial_contour`	0, 1	Determines whether to let the user interactively specify an initial contour or to load an existing initial contour. In the former case, the new initial contour will be saved to the disk with the file name specified in `initial_contour_file_name`. In the latter case, the contour will be loaded from the file `initial_contour_file_name`.
`initial_contour_file_name`	string	See above.
`record_movie`	0, 1	Determines whether to save the evolution of the contour as a video in .avi format. If yes, the filename will be `movie_file_name`.
`movie_file_name`	string	See above.
`save_frames`	0, 1	Determines whether to save the evolution of the contour as a series of images, each in .bmp format. If yes, an image will be saved after each iteration of the algorithm, and the filename will have the prefix `frames_file_name` followed by '_frame' and the step number.
`frames_file_names`	string	See above.
`use_Sobolev`	0, 1	Determines whether to use the Sobolev or the L^2 inner product.
`lambda`	positive number	Smallest allowed distance between consecutive vertices of the initial contour.
`steps`	positive integer	Number of gradient descent iterations.
`dt`	positive number	Step size of gradient descent.
`out_size`	positive integer, 'all'	Determines how large an area will be used for computing c_{out}. If value is 'all', the whole image outside the contour will be used.
`min_dist`	positive number < `max_dist`	Smallest allowed distance between consecutive vertices of the initial contour.
`max_dist`	positive number > `min_dist`	Largest allowed distance between consecutive vertices of the initial contour.

Parameter	Possible Values	Description
`smooth_length`	[0,1)	The fraction of the number of vertices of the curve used for the moving average smoothing of the initial contour. If value is 0 then no smoothing will be done.

6. Experimental Results

All of the experiments described in this section were conducted using `lambda`=0.042, `dt`=0.045, `out_size`=30, `min_dist`=1, `max_dist`=2, `smooth_length`=0.095. The time flow in the image sequences is always from left to right. The first row of images always shows, from left to right, the original image, the initial contour and the smoothed initial contour. The experiments can all be reconstructed by running the algorithm on the images with the corresponding `initial_contour_file_name`.

Fig. 2 demonstrates the advantage of Sobolev active contours over L^2 active contours. The second and third rows of images show, respectively, the evolution using L^2 and Sobolev. The noise in the image prevents the curve from evolving smoothly when using L^2, and the evolution is badly disrupted. The Sobolev active contour, on the other hand, stays smooth and copes with the noise. It can also be seen that in the Sobolev evolution a self-intersection was caused due to a bad choice of an initial contour, yet the algorithm eliminated it.

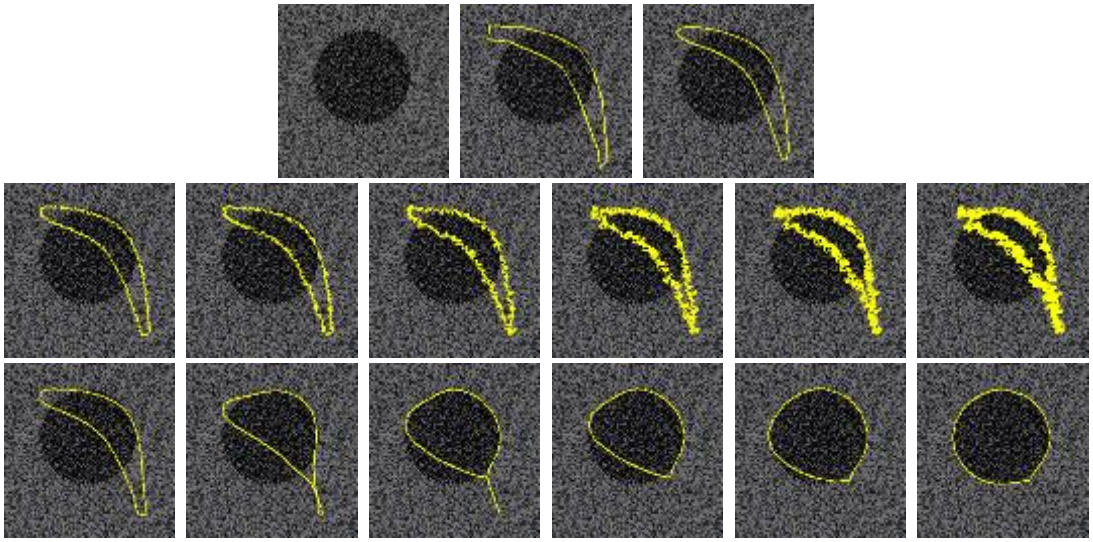

Fig. 2

More examples of the Sobolev evolution are shown in fig. 3 – fig. 5.

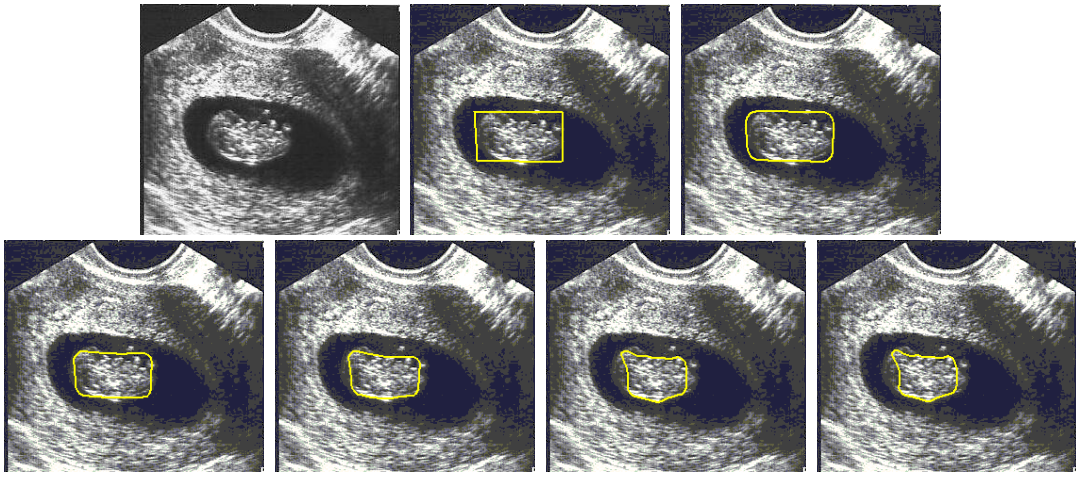

Fig. 3

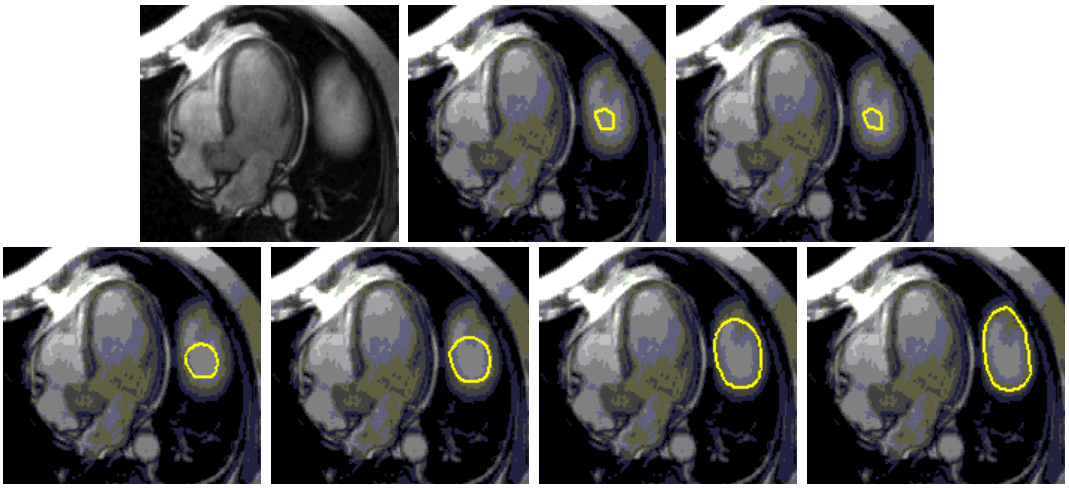

Fig. 4

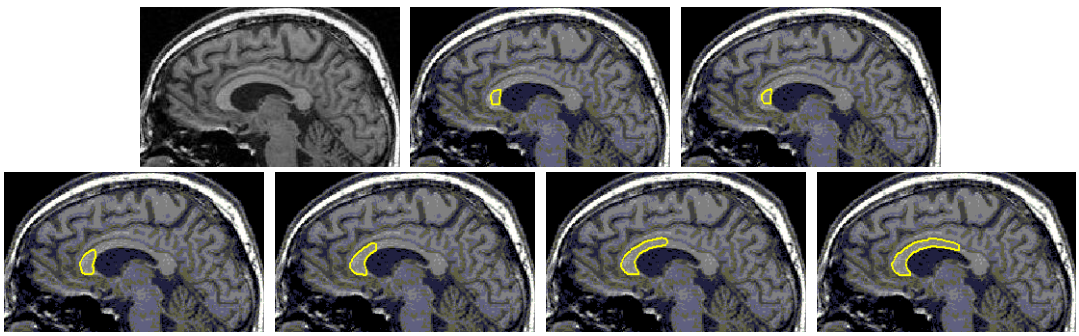

Fig. 5

References

T. F. Chan, L. A. Vese, “Active Contours Without Edges”, *IEEE Trans. Image Processing*, vol. 10, no. 2, pp. 266-277, Feb. 2001.

G. Sundaramoorthi, A. Yezzi, A. C. Mennucci, “Sobolev Active Contours”, *Int’l. J. Computer Vision*, vol. 73, no. 3, pp. 345-366, Jul. 2007.