Fluid vector flow and applications in infant brain MRI analysis

Tao Wang

Abstract

A parametric active contour model based on fluid vector flow is presented in this paper. The contribution of this model is two-fold. First, it has the largest capture range. Second, it is able to extract concave shape. We apply this method to infant brain MRI analysis and the results are very impressive.

1. Introduction

Active contour models (snakes) [1-4] have been proven to be very useful for segmentation and object tracking. In the literature, there are two types of active contour models: parametric active contour models [1-3] and level set active contour models [4]. Level set active contour models have a number of advantages such as insensitive to initializations and capturing multiple objects. However, they also have many shortcomings. First, continuity of contour is not considered in level set active contour models (see Figure 1). Second, they tend to extract many "false" objects when noise exists (see Figure 2) because of the ability of capturing multiple objects. For many applications, users want to extract single object with continuous contour. Since continuity and uniqueness of contour is guaranteed in parametric active contour models, we focus on parametric active contour models in this paper.

Figure 1: discontinuity of level set snake.

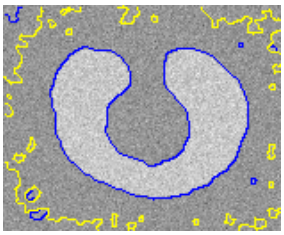
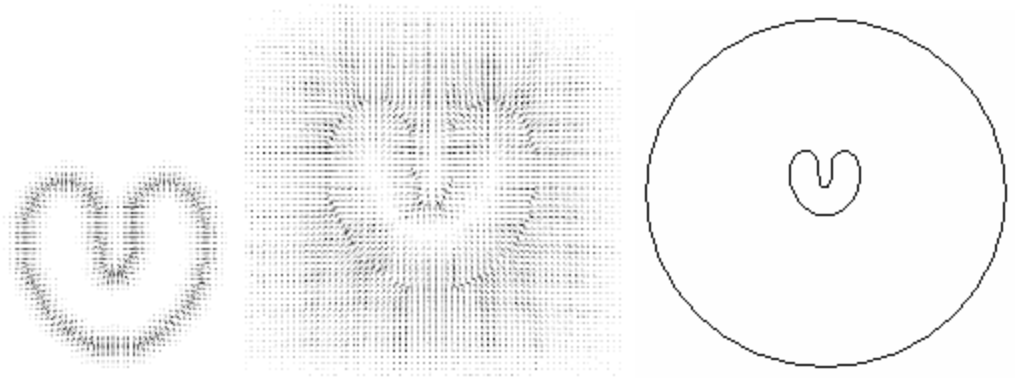


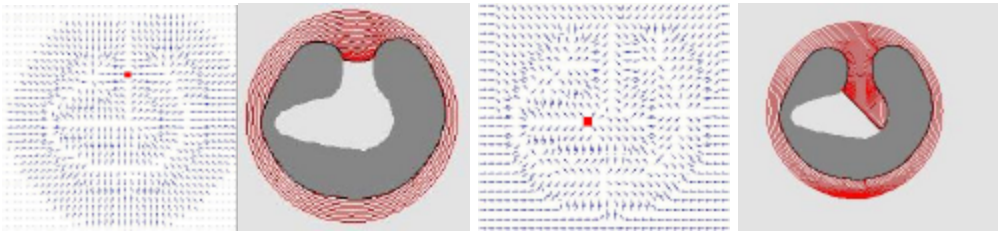
Figure 2: "false" objects extracted by level set snake.

The internal spline energy of parametric active contour model can guarantee continuity. In addition to that, uniqueness is also guaranteed because one parametric active contour will not evolve into multiple contours. However, parametric active contour models have two major shortcomings. First, the capture range is limited (see Figure 3 (a) and (b)). If the initial contour given by a user is out of the capture range, the active contour will not evolve (see Figure 3 (c)). Second, they are unable to extract acute concave shapes (see Figure 4).



(a) (b) (c)

Figure 3: capture range of (a) traditional snake (b) GVF snake (c) initial contour is out of the capture ranges of traditional snake and GVF snake.



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

Figure 4: (a) a saddle point in GVF (b) GVF snake (c) a stationary point in BVF (d) BVF snake

In this paper, a parametric active contour model based on fluid vector flow is proposed. It has two major advantages. First, it has the largest capture range (the entire image) by using a polar interpolation method. Second, it is able to extract acute concave shape. We notice other parametric active contour models are unable to extract acute concavities because the external force fields are stationary [4] so that the active contours could be stuck at saddle points and/or stationary points (see Figure 4). In our method, the external force field will change dynamically with the evolution of the active contour. Therefore, the active contour will be stuck and acute concave shapes can be extracted.

2. Background

2.1. Traditional snake

A traditional snake is a parametric active contour:

$c(s) = (x(s), y(s)), s \in [0, 1]$ (1)

With a given initial contour, it evolves within an image $I(x, y)$ to minimize the energy functional:

$$E_{snake} = \int_0^1 [E_i(c(s)) + E_e(c(s))]ds \quad (2)$$

where E_i is the internal (spline) energy and E_e is the external energy.

The internal energy $E_i = \dfrac{a(s)|c'(s)|^2 + b(s)|c''(s)|^2}{2}$ (3)

In many implementations, the coefficient of the first-order term is a constant, $a(s) = a$; and the $ß(s)$ is set to zero to allow the snake to be second-order discontinuous to develop a corner. Many parametric active contour models share the same internal energy. They are mostly different in the expression of external energy. A snake should evolve to minimize the energy functional E_{snake}. This problem can be formulated with the Euler-Lagrange equation. In calculus of variations, the Euler-Lagrange equation of

$$J[c(s)] = \int_{s_0}^{s_1} F(s, c(s), c'(s), c''(s))ds \quad (4)$$

is

$$F_c - \frac{d}{ds} F_{c'} + \frac{d^2}{ds^2} F_{c''} = 0 \quad (5)$$

Therefore, the Euler-Lagrange equation of (2) is

$$ac''(s) - ßc''''(s) + \nabla E_e = 0 \quad (6)$$

To find the numeric solution of (6), the snake is treated as function of time t as well as s:

$$ac''(s,t) - ßc''''(s,t) + \nabla E_e = 0 \quad (7)$$

When the contour stabilizes, the time term vanishes and the solution is achieved.

2.2. GVF snake

GVF snake diffuses the edge information from the object contour to its neighborhood therefore it has larger capture range than the traditional snake. The external force of GVF snake [2] differs from traditional snake in that it cannot be written as the negative gradient of a potential function. In addition to that, the GVF snake is formulated directly from a force balance condition rather than a variational formulation. The gradient vector flow is defined to be the vector field

$$G_{gvf}(x, y) = (u(x, y), v(x, y)), x \in [x_0, x_1] \text{ and } y \in [y_0, y_1] \quad (8)$$

that minimizes the energy functional

$$E_{gvf} = \int_{y_0}^{y_1}\int_{x_0}^{x_1} (k(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |G_{gvf} - \nabla f|^2) dxdy \quad (9)$$

Where k is a blending parameter, u_x, u_y, v_x, *and* v_y are the derivatives of the vector field, ∇f is the gradient of the edge map. The GVF snake is computed by solving the following Euler-Lagrange equations:

$$k\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \quad (10)$$

$$k\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0 \quad (11)$$

2.3. BVF snake

The basic idea of BVF [3] is interpolation. In this way, it extends the capture range to the entire image. It applies a threshold to generate a binary boundary map of the input image. Then, four potential functions $?_x$, $?_y$, $?_{xy}$, and $?_{yx}$ are computed by line-by-line interpolations in horizontal, vertical and two diagonal directions (see Figure 5). The boundary vector flows are defined based on the gradients of the potential functions as:

$$?_1 = (\nabla ?_x, \nabla ?_y) \quad (12)$$

$$? _2 = (\frac{\sqrt{2}}{2}(\nabla ? _{xy} + \nabla ? _{yx}),\ \frac{\sqrt{2}}{2}(\nabla ? _{xy} - \nabla ? _{yx})) \quad (13)$$

The external force is defined as:

$$E_e(x, y) = ? (x, y) \quad (14)$$

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

Figure 5: Interpolations of 4 directions: (a) horizontal; (b) vertical; (c) and (d) diagonals.

3. FVF: fluid vector flow

Given an input image

$$I(x, y) = f(x_j, y_k) \in R, j \in [0, 1, \ldots, M-1] \text{ and } k \in [0, 1, \ldots, N-1]. \quad (15)$$

and a closed parametric contour

$$c(x, y) = (x_i, y_i) \in R^2, i \in [0, 1, \ldots, P-1] \quad (16)$$

we assume the user wants to evolve the contour to extract a target object $O(x, y)$. Our approach has three main stages: binary boundary map generation, vector flow initialization and fluid vector flow computation. The internal energy is equation (3).

3.1. Binary boundary map generation

The boundary map is defined as:

$$M_B(x, y) = |\nabla(- G_s(x, y) * I(x, y))| \quad (17)$$

where $G_s(x, y)$ is a Gaussian smooth filter with deviation s and ∇ is the gradient operator. Then, we compute the normalized boundary map:

$$M_{NB}(x, y) = \frac{M_B(x, y) - \min(M_B(x, y))}{\max(M_B(x, y)) - \min(M_B(x, y))} \quad (18)$$

Inspired by BVF [3], we apply a threshold $T \in [0, 1]$ to generate the binary boundary map:

$$M_{BB}(x, y) = \begin{cases} 1, & \text{if } M_{NB}(x, y) > T \\ 0, & \text{otherwise} \end{cases} \quad (19)$$

Figure 6 shows an input image and its binary boundary map (threshold $T = 0.8$).

Figure 6: (Left) an image and (Right) its binary boundary map.

3.2. Vector flow initialization

The parametric contour $c(x, y)$ may be outside, inside, or overlap the target object $O(x, y)$ as shown in Figure 7. To demonstrate the basic idea of our algorithm, we suppose the contour is a circle and outside the object. Since any parametric contour is represented in discrete format by (16), the method described below can be generalized to other

parametric contours with ease. In addition to that, our method can be easily modified when the contour is inside or overlaps the object.

Figure 7: Contour is (Left) outside (Mid) inside (Left) overlap the object.

For the circle $c(x, y)$, its center is:

$$c_c(x_c, y_c) = \left(\frac{\sum_{i=0}^{P-1} x_i}{P}, \frac{\sum_{i=0}^{P-1} y_i}{P}\right) \tag{20}$$

Then we define an external energy functional as:

$$E_e(x, y) = \begin{cases} c\,(f_x - \cos f,\ f_y - \sin f), & when\ M_{BB}(x, y) == 0 \\ 0, & otherwise \end{cases} \tag{21}$$

where c is normalization operator, $(f_x, f_y) = c\,(\nabla I(x, y))$, and

$$f = \begin{cases} \arctan\left(\frac{y - y_c}{x - x_c}\right), & when\ x \neq x_c \\ \frac{p}{2}, & when\ x == x_c\ and\ y > y_c \\ \frac{3p}{2}, & when\ x == x_c\ and\ y < y_c \end{cases}, \quad f \in [0, 2p] \tag{22}$$

The energy E_e has two terms: the gradient term and the directional term. When the contour is far from the object, the directional force makes the contour shrink. When the contour is close to the object, the gradient force pushes the contour (see Figure 8 (b)) to the object. However, if concavity exists, convergence will not be achieved. The contour will evolve to the yellow line show in Figure 8 (c).

The vector flows of (21) spread around in the entire image $I(x, y)$. The capture range is the whole image so that it is the largest. Therefore, even if the initial contour given by user is far from the object (see Figure 3(c)), the snake can still evolve to capture the object. Moreover, if the initial contour is not given by user, we can use the contour of the image as the initial contour and snake can also evolve to capture the object. This feature is not available neither in traditional snake [1] or GVF snake [2].

Although the capture range of BVF [3] is also largest, our method is different. The difference is two-fold. First, the performance of our method is better because interpolation is avoided. Second, the BVF interpolation is done in four directions but our method is direction invariant ($f \in [0, 2p]$).

3.3. Fluid vector flow computation

In this step, a boundary trace method is applied to the binary boundary to get an array of control points:

$B(x_q, y_q) = ?(M_{BB}(x, y)),\ q \in [0, 1, \ldots, Q-1]$ (23)

where $?$ is boundary trace operator and Q is the number of the control points.

Then, we define fluid vector flow energy functional as:

$$E_{FVF}(x, y) = \begin{cases} c(c(\nabla \mathrm{I}(x, y)) + c(x - x_q, y - y_q)), & when\ M_{BB}(x, y) == 0 \\ & and\ (x_q, y_q) \in B \\ 0, & otherwise \end{cases} \quad (24)$$

The energy E_{FVF} has two terms: the gradient term and the directional term. The directional force attracts contour towards itself. When a control point is in a concave region, the contour can evolve into the concave region because of this force. When the contour is close to the object, the gradient force will push the contour (see Figure 8 (b)) to the object. Convergence will be achieved when the contour stops evolving.

In Figure 8 (c) – (k), the fluid vector flows are shown as blue arrows, the active contours are yellow lines, the binary boundaries are black lines, and $B(x_q, y_q)$ are green points. The control points selected by the boundary trace operator moves along the binary boundary, like water flows along the boundary. Therefore, we name it "fluid vector flow".

The pseudo code of FVF is as follows:

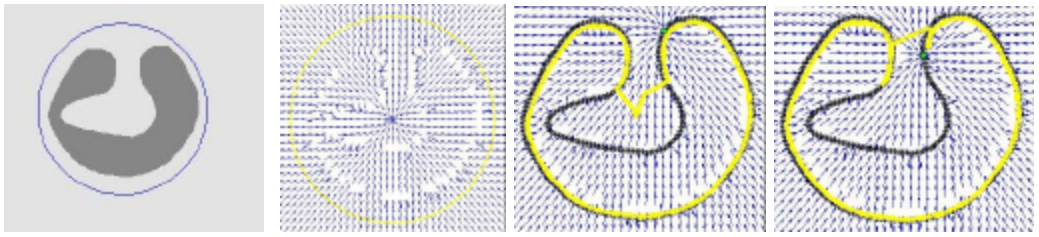
Algorithm

Input: an image I and an initial snake S_i

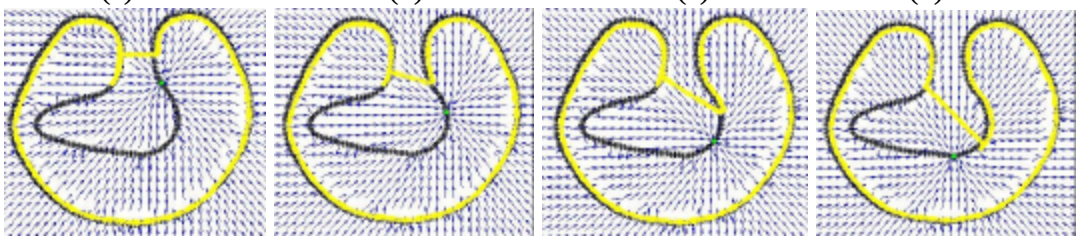
1. Compute binary boundary map.
2. Compute initial vector flow and evolve the snake.
3. While convergence is not reached
    a. Find a new control point on the binary boundary map
    b. Create new vector flow based on the new control point and evolve the snake

Output: FVF snake S_D

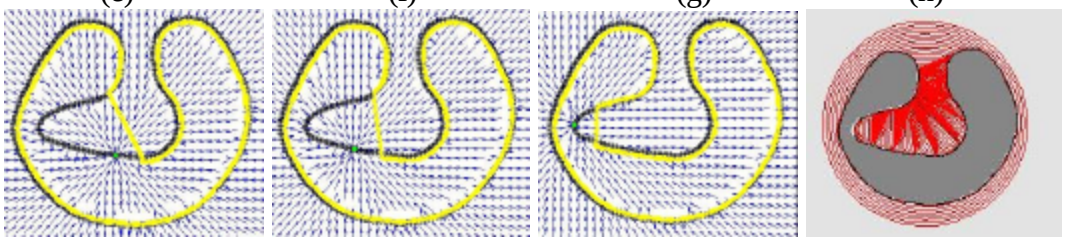
End algorithm



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



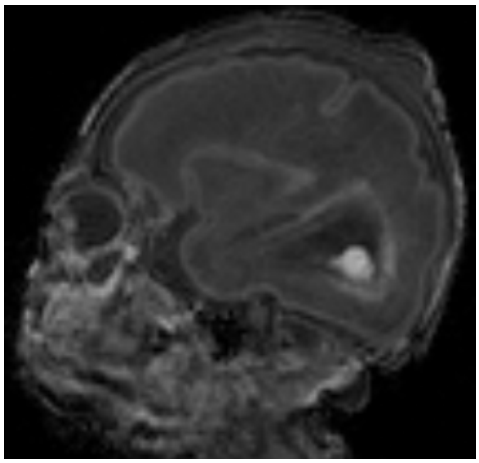
(e) (f) (g) (h)



(i) (j) (k) (l)

Figure 8: (a) image and contour (b) initial vector flow (c)-(k) fluid vector flow (l) result.

4. Experiments



References

[1] Snakes: Active contour models.
[2] Snakes, Shapes, and Gradient Vector Flow.
[3] Boundary vector field for parametric active contours.
[4] MAC: Magnetostatic Active Contour Model.