A FAST LEVEL SET METHOD WITHOUT SOLVING PDES

Yonggang Shi, William Clem Karl

Electrical and Computer Engineering Department
Boston University
Boston, MA 02215
{yshi, wckarl}@bu.edu

ABSTRACT

In this paper, we propose a novel and fast level set method without the need of solving PDEs while preserving the advantages of level set methods, such as the automatic handling of topological changes. The foundation of our method is the direct use of an optimality condition for the final curve location based on the speed field. By testing this condition, only simple operations like insertion and deletion on two lists of boundary points are needed to evolve the curve. Our method is suitable for a set of general evolution speeds that are composed of two parts: an external speed derived from the image data and a speed term imposing boundary smoothness or regularization. In our experiments, we demonstrate that our algorithm is approximately two orders of magnitude faster than previous optimized narrow band algorithms for image segmentation tasks.

1. INTRODUCTION

The localization of object boundaries is an important and challenging task in many imaging problems, such as segmentation and tracking. In recent years there has been intensive interest in the use of the level set method[1, 2, 3] for boundary localization. The level set method is attractive for its ability to handle topological changes automatically. Its numerical implementation is also straightforward for any dimension. While the level set method has many advantages, its implementation, based on the solution of certain partial differential equations (PDEs), results in a significant computational burden, which limits its use in real time applications. In this paper, we propose a novel and fast level set algorithm which sidesteps solution of PDEs, allowing a significant reduction in computation time, and the promise of real time implementation.

Other research has aimed at reducing the computational load of level set techniques. When only the zero level set is of interest, narrow band techniques have been proposed to accelerate the evolution process. In [4], a tube is constructed in the neighborhood of the zero level set with the fast marching method[5, 6] and the PDE is solved only within this tube. Improvements to the above narrow band algorithm were proposed in [7] and [8]. Both algorithms are similar in that they reinitialize the level set function to be a signed distance function at every iteration and the tube is only constructed once at the beginning and updated dynamically thereafter. In [7], the reinitialization of the level set function is achieved by solving a Hamilton-Jacobi PDE for a fixed number of steps in every iteration evolving the level set function. In [8], the reinitialization is achieved by computing the distance function approximately. For both methods, a bandwidth of at least five has to be selected for reasonable evaluation of all the required gradients.

Despite their differences, previous narrow band algorithms are common in that they all attempt to track the evolution of the zero level set accurately by solving the associated PDE locally. However, this accuracy is not necessary for many imaging problems, such as segmentation, where the goal is to extract the final object boundary. In such cases, the evolution process of the level set function itself is of less interest. In this paper, we propose a new algorithm to reduce the computation requirements and convergence time of the level set method dramatically for this type of problem. In our method, we first define an optimality condition for the final location of the boundary curve (i.e. the level set) in terms of the behavior of the evolution speed in the neighborhood of the boundary. We then use this condition to evolve the level set function by updating only two boundary point lists. The resulting algorithm can be applied to problems with a very general set of underlying speed fields that are composed of two parts: an external evolution speed derived from the image data and a speed term imposing boundary smoothness or regularization. Compared with the algorithm in [8], our method is approximately two orders of magnitude faster for image segmentation.

We introduce the optimality condition for the final curve in terms of the evolution speed field in Section II. A fast algorithm based on this condition is then proposed in Section III. We demonstrate the application of our algorithm to image segmentation in Section IV. Experimental results for both edge-based and region-based models are presented.

2. OPTIMALITY CONDITION

In the curve evolution method, an initial curve C is evolved based on a speed field F using the following curve evolution differential equation:

$$\frac{dC}{dt} = F\vec{N} \tag{1}$$

where $\vec{N}$ is the normal of the curve pointing outward. The speed field F is generally composed of two parts: an external speed derived from the image data and an internal or intrinsic speed that depends on the geometrical properties of C. In solving problems based on a variational energy minimization [9], this speed and the corresponding curve evolution, is associated to a gradient descent

This work was partially supported by the Engineering Research Centers Program of the National Science Foundation under award number EEC-9986821, National Institutes of Health under Grant NINDS 1 R01 NS34189, Air Force under Award number F49620-03-1-0257.

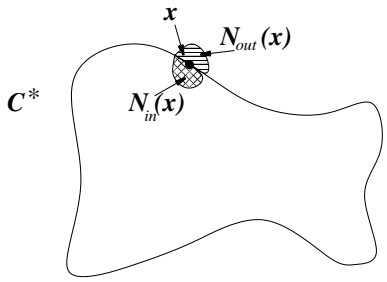


Fig. 1. The neighborhood of a point on the optimal curve.

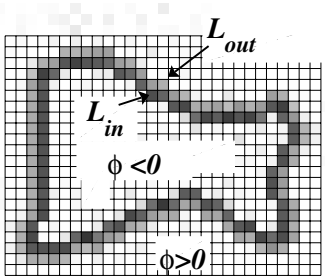


Fig. 2. Implicit representation of a curve in 2D based on two lists of boundary points L_{in} and L_{out}.

solution. The aim is then to evolve the curve C until it stops at a (in general, local) minima of the energy C^*, corresponding to a stationary point of the dynamic equation (1). The curve C^* at such stationary points must satisfy the following optimality condition in terms of the speed field F.

The Continuous Optimality Condition : *$\forall \mathbf{x} \in C^*$, there is an outside neighborhood $N_{out}(\mathbf{x})$ and an inside neighborhood $N_{in}(\mathbf{x})$ (see Fig.1) such that the speed field F satisfies:*

$$F(\mathbf{y}) < 0 \ \ \forall \mathbf{y} \in N_{out}(\mathbf{x}) \text{ and } F(\mathbf{y}) > 0 \ \ \forall \mathbf{y} \in N_{in}(\mathbf{x}).$$

In level set methods, the object boundary is represented implicitly as the zero level set of a function ϕ. Here we choose ϕ to be negative inside the curve C and positive outside C. We assume that ϕ is defined over a domain D in $\mathbf{R}^K$ and it is discretized onto a grid of size $M_1 \times M_2 \times \cdots \times M_K$. Without loss of generality, we assume the grid is sampled uniformly and the sampling interval is one. For a point $\mathbf{x}$ in the grid, we denote its coordinate as $\mathbf{x} = (x_1, x_2, \cdots, x_K)$. We may represent a given object boundary uniquely through two lists of points: the list of outside boundary points L_{out} and the list of inside boundary points L_{in}, as shown in Fig. 2. Formally they are defined as:

$$L_{out} = \{\mathbf{x} \mid \phi(\mathbf{x}) > 0 \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) < 0\}$$
$$L_{in} = \{\mathbf{x} \mid \phi(\mathbf{x}) < 0 \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) > 0\}$$

where $N(\mathbf{x})$ is a discrete neighborhood of $\mathbf{x}$ defined as follows:

$$N(\mathbf{x}) = \left\{ \mathbf{y} \in D \;\middle|\; \sum_{k=1}^{K} |y_k - x_k| = 1 \right\} \quad \forall \mathbf{x} \in D.$$

Using these two boundary point lists, we can translate the continuous optimality condition into the discrete domain as:

The Discrete Optimality Condition : *For the curve C^* with boundary points L_{in}, L_{out} the speed field F satisfies:*

$$F(\mathbf{x}) < 0 \ \ \forall \mathbf{x} \in L_{out} \text{ and } F(\mathbf{x}) > 0 \ \ \forall \mathbf{x} \in L_{in}. \qquad (2)$$

We can see that only the sign of the speed is necessary in testing this optimality condition. Since the object boundary is uniquely determined by L_{out} and L_{in}, and they are defined using only the sign of ϕ, we can propose a novel strategy to evolve the boundary. At every boundary point, we test the following non-convergence condition:

$$Con(\mathbf{x}) = \begin{cases} 1, & \text{if } \exists \mathbf{y} \in N(\mathbf{x}), \text{s.t. } \phi(\mathbf{x})\phi(\mathbf{y}) < 0 \\ & \text{and } F(\mathbf{x})F(\mathbf{y}) > 0; \\ 0, & otherwise. \end{cases}$$

If $Con(\mathbf{x}) = 0$, it means the optimality condition is satisfied at that point, and we can stop evolving it. If $Con(\mathbf{x}) = 1$, it means the optimality condition is not satisfied and we can move the boundary point outward or inward according to the sign of the speed. To accomplish this motion, we need only update the two lists L_{out} and L_{in} using simple operations, such as insertion and deletion. The value of ϕ only needs to be updated to be consistent with the definition of L_{out} and L_{in}. The basic idea is to directly evolve the boundary in the discrete domain in the direction indicated by the sign of the speed function. Compared with previous narrow band algorithms, our strategy removes the numerical restrictions that follow from accurate solution of the level set PDE. As a result the algorithm is much faster. In the next section, we will present the details of our algorithm.

3. FAST CURVE EVOLUTION ALGORITHM

In this section, we first develop the fast algorithm assuming the speed field F consists only of an external speed. Later we will incorporate modifications to include the effects of an intrinsic term imposing boundary smoothness or regularization. To this end, assume we have only an external speed function F.

We begin by specifying our representation of the curve C through the a level set function ϕ. For faster computation, we choose ϕ to be integer valued as follows:

$$\phi(\mathbf{x}) = \begin{cases} 3, & \text{if } \mathbf{x} \text{ is outside } C \text{ and } \mathbf{x} \notin L_{out}; \\ 1, & \text{if } \mathbf{x} \in L_{out}; \\ -3, & \text{if } \mathbf{x} \text{ is inside } C \text{ and } \mathbf{x} \notin L_{in}; \\ -1, & \text{if } \mathbf{x} \in L_{in}. \end{cases} \qquad (3)$$

With this definition, we can easily tell the relative position of a point in the scene with respect to the curve C from its value ϕ. Near the curve this function is similar to a distance function.

Next, let us define two procedures on L_{out} and L_{in}. The first procedure $check\_in(\mathbf{x})$ switches a point $\mathbf{x}$ from the set L_{out} to the set L_{in}, and is defined as follows:

$check\_in(\mathbf{x})$:

- Step 1: Delete $\mathbf{x}$ from L_{out} and add it to L_{in}. Set $\phi(\mathbf{x}) = -1$ and compute its speed $F(\mathbf{x})$.
- Step 2: $\forall \mathbf{y} \in N(\mathbf{x})$ satisfying $\phi(\mathbf{y}) = 3$, add $\mathbf{y}$ to L_{out}, set $\phi(\mathbf{y}) = 1$, and compute its speed $F(\mathbf{y})$.

The second procedure $check\_out(\mathbf{x})$ switches a point $\mathbf{x}$ from L_{in} to L_{out} and is defined as follows:

$check\_out(\mathbf{x})$:

- Step 1: Delete $\mathbf{x}$ from L_{in} and add it to L_{out}. Set $\phi(\mathbf{x}) = -1$ and compute its speed $F(\mathbf{x})$.
- Step 2: $\forall \mathbf{y} \in N(\mathbf{x})$ satisfying $\phi(\mathbf{y}) = -3$, add $\mathbf{y}$ to L_{in}, set $\phi(\mathbf{y}) = -1$, and compute its speed $F(\mathbf{y})$.

The basic steps of our algorithm to evolve the curve C according to the external speed function F can now be described as follows. At each iteration, the non-convergence condition is sequentially tested at each point in L_{out} and L_{in}. If it is true, we then move the boundary inward or outward according to the sign of F. If $F > 0$ at a point in L_{out}, we then apply a $check\_in(\mathbf{x})$ procedure to this point to move the boundary outward. If $F < 0$ at a point in L_{in}, then we apply a $check\_out(\mathbf{x})$ procedure to this point to move the boundary inward. During this process, idle points that are no longer formal boundary points of C can be generated in both lists. Thus we delete those points after each iteration to make sure L_{out} and L_{in} satisfy their definition.

We next discuss how to incorporate boundary smoothness and regularization into our algorithm with low computational cost. Such smoothness is usually imposed through the addition of an intrinsic speed, chosen proportional to boundary curvature. The curvature-based speed is typically calculated from the Laplacian of the level set function ϕ, which is taken as the signed distance function of the curve. Such calculations are costly and also pose a challenge for our discrete-valued level set function and sign-based evolution strategy. However, we know that evolving a function according to its Laplacian is equivalent to Gaussian smoothing, based on solutions to the heat equation. Motivated by this observation, we instead impose boundary smoothness by performing a separate stage of curve evolution based on Gaussian filtering of our discrete valued level set function $\phi(\mathbf{x})$.

In particular, what we do in this smoothing step is to Gaussian filter the level-set function $\phi(\mathbf{x})$ and then to update the lists L_{out} and L_{in} based on the sign of the result. Let G denote a K dimensional Gaussian filter of size $N_g \times N_g \times \cdots \times N_g$. One cycle of our smoothing evolution through all the boundary points is as follows:

Smoothing Cycle:

- For every point $\mathbf{x}$ in L_{out}, if $G \otimes \phi(\mathbf{x}) < 0$, $check\_in(\mathbf{x})$;
- For every point $\mathbf{x}$ in L_{in}, if $G \otimes \phi(\mathbf{x}) > 0$, $check\_out(\mathbf{x})$.

Similar to the evolution driven by the external speed, the smoothing evolution will also generate idle points. As before, we eliminate such points at the end of each evolution step.

This smoothing evolution is combined with the evolution corresponding to the external speed to form our overall fast level set algorithm, which is summarized in Table 1. We alternate between these two types of evolution in our algorithm. For each overall cycle we first run N_a evolution iterations corresponding to the external speed F, then we apply N_g evolution steps of the smoothing cycle. The relative strength of these two effects is controlled by the values (N_g, N_a). The parameter N_g is the size of the Gaussian filter and controls the elimination of small holes in the final result. For example, to eliminate holes with radius smaller than r, we can choose $N_g = 2r$. Typically we choose $N_a \geq N_g$.

Table 1. Summary of fast level set algorithm.

- Step 1: Initialize arrays ϕ, F, and lists L_{out} and L_{in}.
- Step 2: For i=1:N_a do
  - Step 2.1: Scan through the lists and update ϕ, F, L_{out} and L_{in}.
    - For each point $\mathbf{x} \in L_{out}$ with $F(\mathbf{x}) > 0$, check_in($\mathbf{x}$) if $Con(\mathbf{x})$;
    - For each point $\mathbf{x} \in L_{in}$, if $\forall \mathbf{y} \in N(\mathbf{x}), \phi(\mathbf{y}) < 0$, delete $\mathbf{x}$ from L_{in}, and set $\phi(\mathbf{x}) = -3$.
    - For each point $\mathbf{x} \in L_{in}$ with $F(\mathbf{x}) < 0$, check_out($\mathbf{x}$) if $Con(\mathbf{x})$;
    - For each point $\mathbf{x} \in L_{out}$, if $\forall \mathbf{y} \in N(\mathbf{x}), \phi(\mathbf{y}) > 0$, delete $\mathbf{x}$ from L_{out}, and set $\phi(\mathbf{x}) = 3$.
  - Step 2.2: Check the discrete non-convergence condition. If it is satisfied, go to Step 2.1; otherwise, go to Step 4.
- Step 3: For i=1:N_g
  - For every point $\mathbf{x}$ in L_{out}, compute $G \otimes \phi(\mathbf{x})$. If $G \otimes \phi(\mathbf{x}) < 0$, $check\_in(\mathbf{x})$;
  - For each point $\mathbf{x} \in L_{in}$, if $\forall \mathbf{y} \in N(\mathbf{x}), \phi(\mathbf{y}) < 0$, delete $\mathbf{x}$ from L_{in}, and set $\phi(\mathbf{x}) = -3$.
  - For every point $\mathbf{x}$ in L_{in}, compute $G \otimes \phi(\mathbf{x})$. If $G \otimes \phi(\mathbf{x}) > 0$, $check\_out(\mathbf{x})$.
  - For each point $\mathbf{x} \in L_{out}$, if $\forall \mathbf{y} \in N(\mathbf{x}), \phi(\mathbf{y}) > 0$, delete $\mathbf{x}$ from L_{out}, and set $\phi(\mathbf{x}) = 3$.
- Step 4: Stop the algorithm.

4. EXPERIMENTAL RESULTS

In this section, we demonstrate our algorithm for the problem of image segmentation. We apply it to both edge-based and region-based models. We compare our computation time with corresponding algorithms implemented in the Insight Toolkit (ITK)[10], which has both edge-based and region-based image segmentation methods implemented with the "highly optimized" [11] sparse-field method [8]. All experiments are run on a 3.2GHz Intel Xeon CPU with 3.5GB Memory.

For edge-based segmentation, we follow the geodesic model [12, 13] and two images are tested. The first is an MRI heart image as shown in Fig. 3(a). It size is 512×512. In the second image, shown in Fig. 4(a), a plane is segmented. The size of the image is 481×321. As we can see from Fig.3 and 4, similar results are obtained by our algorithm and ITK.

For region-based segmentation, we apply our algorithm to a simple threshold-based model implemented in ITK, where the external speed F is 1 when the image intensity is in a specified range and -1 otherwise. For both ITK and our algorithm, the same range is chosen in the following experiments. In the first experiment, we segment the white matter from a MRI brain image as shown in Fig. 5(a).The size of the image is 181×217. In our second experiment, we segment a spiral galaxy image. The original image is shown in Fig. 6(a). The size of the image is 512×512. From the segmentation results shown in Fig. 5 and Fig. 6, we can see that the results of ITK and our algorithm are comparable.

The computation times of ITK and our algorithm for the four segmentation experiments are listed in Table 2. For all the experi-

(a) (b) (c)

Fig. 3. MRI heart image segmentation results. (a) The original image and the initial curve (the white circle). (b) The result of ITK. (c) The result of our algorithm.

(a) (b) (c)

Fig. 4. Plane image segmentation results. (a) The original image and the initial curve (the black circle). (b) The result of ITK. (c) The result of our algorithm.

ments, we can see that our algorithm is approximately two orders of magnitude faster than the sparse-field algorithm implemented in ITK. With our algorithm, all the images are segmented in less than $1/24$ second, thus it holds the potential of real-time video-rate processing.

Table 2. Comparison of image segmentation time with ITK.

Image	ITK	New Fast Algorithm	Speedup Factor
MRI heart	0.877s	0.00816s	107
Plane	1.837s	0.0261s	70
MRI brain	1.608s	0.0153s	105
Spiral galaxy	2.178s	0.0237s	92

5. REFERENCES

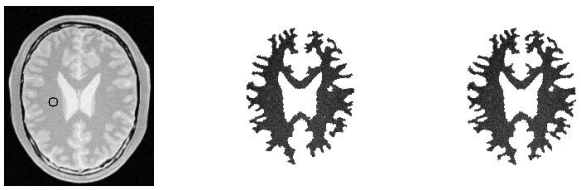
[1] S. Osher and J.A. Sethian, "Fronts propagation with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of computational physics*, vol. 79, pp. 12–49, 1988.

[2] J. Sethian, *Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, Cambridge University Press, 1999.

[3] S. Osher and R.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer Verlag, 2002.

[4] D. Adalsteinsson and J.A. Sethian, "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, pp. 269–277, 1995.

[5] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Trans. on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, Sep 1995.



(a) (b) (c)

Fig. 5. MRI brain image segmentation results.(a) The original image and the initial curve (the black circle). (b) The result of ITK. (c) The result of our algorithm.

(a) (b) (c)

Fig. 6. Spiral galaxy image segmentation results. (a) The original image and the initial curve (the white circle). (b) The result of ITK. (c) The result of our algorithm.

[6] J.Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci*, vol. 93, no. 4, pp. 1591–1595, 1996.

[7] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A pde-based fast local level set method," *Journal of Computational Physics*, vol. 155, pp. 410–438, 1999.

[8] R.T. Whitaker, "A level-set approach to 3d reconstruction from range data," *Int'l Journal of Computer Vision*, vol. 29, no. 3, pp. 203–231, OCT 1998.

[9] A. Tsai, A. Yezzi, and A. Willsky, "Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification," *IEEE Trans. on Image Processing*, vol. 10, no. 8, pp. 1169–1186, AUG 2001.

[10] "The insight toolkit," http://www.itk.org, 2003.

[11] A. Lefohn, J. Kniss, C. Hansen, and R. Whitaker, "A streaming narrow-band algorithm: interactive computation and visualization of level sets," *IEEE Trans. on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 422–433, Jul/Aug 2004.

[12] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[13] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contour models," in *Proceedings of ICCV*, Boston,USA, 1995, pp. 810–815.