Geometric Active Deformable Models in Shape Modeling

Haiyan Wang and Bijoy Ghosh

***Abstract*—This paper analyzes the problem of *shape modeling* using the principle of *active geometric deformable models.* While the basic modeling technique already exists in the literature, we highlight many of its drawbacks and discuss their source and steps to overcome them. We propose a new stopping criterion to address the stopping problem. We also propose to apply *level set algorithm* to implement the active geometric deformable models, thereby handling topology changes automatically. To alleviate the numerical problems associated with the implementation of the level set algorithm, we propose a new *adaptive multigrid narrow band algorithm.* All the proposed new changes have been illustrated with experiments with synthetic images and medical images.**

***Index Terms*—Curvature, curve evolution, geometric active contour model, level set, narrow band algorithm, topology free boundary detection.**

I. Introduction

In computer vision and image processing, an important goal is to recover shapes of objects in either two or three dimensions from various types of visual data. One way to achieve this goal is to detect the associated boundaries in the data using model-based techniques. Such a technique has been originally introduced by Kass *et al.* [1] using *active contour models* or *snakes.* More recently, new methods using *curve evolution theory* in differential geometry has been proposed simultaneously by Caselles *et al.* [2] and by Malladi *et al.* [3]. In recent years, there has been a rising interest in the use of curvature driven flows and partial differential equations (PDE's) in computer vision and image analysis [10]–[22]. Curve or surface evolution, driven by a PDE, has been studied from a theoretical standpoint by Gage and Hamilton [4], Grayson [5], [6], and from the point of view of numerical implementation by Sethian [7], [9] and Osher and Sethian [8]. These PDE based studies have employed a new numerical method called the *level set method,* in which moving curves and surfaces are represented by the level set of a higher dimensional hypersurface and flow of the hypersurface in time is described by a differential equation. The advantage of using a level set representation is that the algorithm can handle changes in the topology of the shape as the curve/surface evolves in time.

In this paper, we have extended the earlier known algorithms on the geometric active curve evolution model described by [2], [3], and [16]–[19]. The proposed algorithm provides an improved stopping criterion for the active curve evolution, and is implemented using the level set method introduced earlier in [7]–[9]. We propose a new scheme for applying the level set method called *adaptive multigrid narrow band algorithm.* The efficiency of the proposed scheme has also been tested on synthetic images and on medical images.

II. Background and Related Work

Based on the theory of curve evolution, Caselles *et al.* [2] and Malladi *et al.* [3], [16] have proposed independently, what is now called the *geometric active contour models* for image segmentation in two or three dimensional data sets. These models are constructed on the basis of the fact that the associated contours/surfaces propagate in time with a velocity profile which is a function of the curvature. Such a curvature based flow ensures that in time the curves are increasingly regular (smooth) among all other possibilities. The other, and possibly the most important, goal of the flow is to ensure that the curve expands or shrinks toward the boundary of objects and stops propagating sufficiently close to the boundary. From the above discussion, it is clear that the velocity of the flow usually contains two terms: one related to the regularity of the curve and the other related to the propagation and stopping at the appropriate true boundary. In order to describe examples of such differential equations, let us consider image $I(X)$ where X is a point in $\mathcal{R}^2$ or $\mathcal{R}^3$. The flow of the contours can be typically described by a partial differential equation as [2], [3], [16]

$$\frac{\partial C(p,t)}{\partial t} = g(\|\nabla G_\sigma * I\|)F(\kappa)\vec{\mathcal{N}} \tag{2.1}$$

where

$C(p,\,t)$	closed evolving contour;
κ	curvature of the curve $C(p,\,t)$ at $(p,\,t)$;
$\vec{\mathcal{N}}$	unit normal vector(inward or outward) along $C(p,\,t)$;
G_σ	Gaussian filter (any other choice of smooth filters suffice as well);
σ	scale-space parameter;
∇	gradient operator;
$F(\kappa)$	evolution speed function, typically;

$$F(\kappa) = 1 - \epsilon\kappa \text{ or } \nu + \kappa \tag{2.2}$$

where $\nu \geq 0$ and $\epsilon > 0$ are constants.

If $g(\|\nabla G_\sigma * I\|) = 1$, (2.1) describes that the evolving contour $C(p,\,t)$ expands or shrinks with the speed $F(\kappa)$ along the normal direction of $C(p,\,t)$. One needs to address, however, the question of how the evolving contour stop at the boundary. This is achieved by forcing the speed of the contour to be 0 at the boundary. Of course, since we do not know the boundary, this has to be estimated from the image data and a typical choice is given by $\|\nabla G_\sigma * I\|$. Since $\|\nabla G_\sigma * I\|$ is large near a boundary, the function $g(\|\nabla G_\sigma * I\|)$ is very small when X is on the boundary and otherwise is 1 if we take $g(\|\nabla G_\sigma * I\|)$ as

$$g(\|\nabla G_\sigma * I\|) = \frac{1}{1 + \|\nabla G_\sigma * I\|^m}, \quad m = 1 \text{ or } 2. \tag{2.3}$$

There are other models proposed in the literature as well. Kichenassamy *et al.* [17], [18] derived another geometric active contour model, called the *geodesic active contour model,* from the curve shortening theory and the conformal Eulidean metric on $\mathcal{R}^2$ as follows:

$$\frac{\partial C(p,\,t)}{\partial t} = g(\|\nabla G_\sigma * I\|)(\nu + \kappa)\vec{\mathcal{N}} - (\nabla g \cdot \vec{\mathcal{N}})\vec{\mathcal{N}}. \tag{2.4}$$

Likewise, Caselles *et al.* [19], [20] also obtained the above model by unifying the curve evolution approaches with the classical *energy minimization methods.*

One of the good points of using the flow model proposed in (2.4) is the existence of the additive term $(\nabla g \cdot \vec{\mathcal{N}})$ to the overall speed function. This additive term has been called a *doublet* [17], [18]. It is claimed that this term attracts the evolving contour to the boundary, as it approaches the boundary. The problem however is in choosing the function $g(\|\nabla G_\sigma * I\|)$ such that $g(\nu + \kappa) - (\nabla g \cdot \vec{\mathcal{N}}) = 0$ on the boundary, particularly since the values of $(\nu + \kappa)$ and $\nabla g \cdot \vec{\mathcal{N}}$ depend on the shape of the evolving contour. A positive error for $g(\nu + \kappa) -$

Manuscript received April 22, 1998; revised June 16, 1999. This work was supported in part by the National Science Foundation under Grant 9720357. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Robert J. Schalkoff.

The authors are with the Department of Systems Science and Mathematics, Washington University, St. Louis, MO 63130 USA. (e-mail: wang@zach.wustl.edu).

Publisher Item Identifier S 1057-7149(00)01156-8.

$(\nabla g \cdot \vec{\mathcal{N}})$, would indicate that the contour passes over the true edge. A negative error, on the other hand, would indicate that the contour would move in the reverse direction away from the true edge along the normal vector. This would force the function $g(\|\nabla G_\sigma * I\|)$ to be large and $\|\nabla g\|$ to be small so that eventually $g(\nu + \kappa) - (\nabla g \cdot \vec{\mathcal{N}})$ would be positive forcing the contour to move toward the edge once again. Thus in practice, the evolving contour would oscillate near the boundary and reach a dynamic balance where points on the contour would move back and forth since the speed function changes the sign. On the other hand, model (2.4) cannot detect the ramp edges well, since a spurious edge may cause $(g(\|\nabla G_\sigma * I\|))(\nu+\kappa) - \nabla g \cdot \vec{\mathcal{N}}) \leq 0$, therefore the contour will stop away from the boundary as illustrated in Fig. 1. Obviously, the value of $(g(\|\nabla G_\sigma * I\|))(\nu + \kappa) - \nabla g \cdot \vec{\mathcal{N}})$ depends on the choice of $g(\|\nabla G_\sigma * I\|)$ and ν, accordingly, ν may have influence on edge detection.

In Fig. 1, we illustrate the effect due to a spurious edge. We show in Fig. 1(a) a synthetic image consisting of a disk of intensity 100 on a background of intensity zero, with a blur filter applied to it, and this filter is a normalized $k \times k$ box filter with unity entries. In Fig. 1(b) we show the segmentation result applying model (2.4) for $k = 7$, $\nu = 1$. After 16 iterations, the evolving contour does not show any perceptible changes in its position. We therefore conclude that the propagation of the contour has stabilized somewhat away from the true edge. The segmentation result shown in Fig. 1(c) is the result of applying proposed algorithm in this paper and the details will be given in following sections.

It is therefore clear that the choice of a stopping criterion is quite critical to the successful execution of a *geometric model* based algorithm. The goal is to move the contour with a speed F and to stop the contour on the boundary. One natural way to choose speed function is to let $F = F_1 + F_2$, and choose $F_2 = 0$ to move the contour with speed F_1 and choose $F_2 = -F_1$ to stop the contour on the boundary. This idea has been explored by Malladi, Sethian and Vemuri [16], where the speed function F has been chosen to be

$$F = F_A + \frac{-F_A}{M_1 - M_2}\{\|\nabla G_\sigma * I\| - M_2\} \tag{2.5}$$

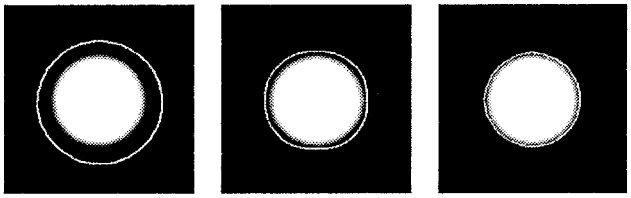
where

- F_A constant;
- M_1 maximum values of the magnitude of the smoothed image gradient $\|\nabla G_\sigma * I\|$;
- M_2 minimum values of the magnitude of the smoothed image gradient $\|\nabla G_\sigma * I\|$.

In principle, when X is close to the boundary, the value of $\|\nabla G_\sigma * I(X)\|$ is close to M_1, thereby forcing the speed function F to be zero. Away from the boundary, the speed function takes a constant value F_A, as has been originally intended. In practice, however, $\|\nabla G_\sigma * I(X)\|$ does not take the same value M_1 over the entire boundary. This leads to a nonzero positive speed function even on the boundary resulting in an overshoot. If the speed function F is a function of curvature, Malladi *et al.* [16] didn't find a speed function that will cause the speed of the moving contour to be zero on edges. Instead, the speed function $F(\kappa)$ is multiplied by a stopping function $g(\|\nabla G_\sigma * I\|)$ given by

$$g(\|\nabla G_\sigma * I\|) = \frac{1}{1 + \|\nabla G_\sigma * I\|}, \text{ or}$$
$$g(\|\nabla G_\sigma * I\|) = e^{-\|\nabla G_\sigma * I\|}. \tag{2.6}$$

However, as has been pointed out in [21], if the initial contour is much closer to one portion of the boundary than the another, the evolving contour will cross over the boundary closest to the initial contour. This is because the *stop function* is small, but not zero, near and at the boundary as detailed and illustrated via examples in [21]. In addition, model (2.1) and (2.4) are kind of multiplication models, the product of speed and stopping function under small speed or small values of stopping function may be same, but they have very different physical meanings.



(a) *initial* (b) *by model*(2.4) (c) *by model*(3.6)

Fig. 1. Ramp edges detection by model (2.4) and model (3.6), $l_{\max} = 5$.

In this paper, a new choice of the stopping function g has been addressed, that would in particular, reduce or eliminate the problem of oscillation as described above and stop the evolving contour on or near the boundary.

III. A New Geometric Active Contour/Surface Model

In this section, a new *geometric active contour/surface model* has been introduced. In this model, a new stopping criterion has been introduced to address the oscillating problem in previous geometric models. The details of the model are now described as follows. Let $C(p, t) = (x(p, t), y(p, t))^T$ be a family of parameterized planar curves in $\mathcal{R}^2$ which is generated by moving an initial contour, where p parameterizes the curves and t parameterizes the family (t usually time). The evolution equation of this family of curves describing the differential change of the curves in t can be written as [23]

$$\begin{cases} \dfrac{\partial C(p, t)}{\partial t} = \langle \vec{\boldsymbol{v}}, \vec{\mathcal{N}} \rangle \vec{\mathcal{N}} + \langle \vec{\boldsymbol{v}}, \vec{\mathcal{T}} \rangle \vec{\mathcal{T}} \\ C(p, 0) = C_0(p) \end{cases} \tag{3.1}$$

where

- $\vec{\boldsymbol{v}}(p, t)$ velocity vector of the curves evolving at point (p, t);
- $\vec{\mathcal{N}}$ unit normal vector;
- $\vec{\mathcal{T}}$ unit tangent vector;
- $\langle \cdot, \cdot \rangle$ Euclidean inner product;
- $\langle \vec{\boldsymbol{v}}, \vec{\mathcal{N}} \rangle$ normal components of the evolution velocity $\vec{\boldsymbol{v}}(p, t)$;
- $\langle \vec{\boldsymbol{v}}, \vec{\mathcal{T}} \rangle$ tangent components of the evolution velocity $\vec{\boldsymbol{v}}(p, t)$.

A basic result from the theory of the curve evolution is that the geometric shape of the curve is only affected by the normal component of the velocity, the tangential component of the velocity affects only the parameterization of the curves, not the geometric shape of the curves [24]. Therefore, (3.1) can be written as

$$\begin{cases} \dfrac{\partial C(p, t)}{\partial t} = \langle \vec{\boldsymbol{v}}, \vec{\mathcal{N}} \rangle \vec{\mathcal{N}} \\ C(p, 0) = C_0(p). \end{cases} \tag{3.2}$$

This means that curve evolution is determined by its normal component of the velocity. The question, of course, how to choose the velocity function so that the curve evolves and stops at the boundaries of an object of interest.

In order to describe the process of selecting the velocity function, we decompose the normal component of the velocity into two components as follows:

$$\langle \vec{\boldsymbol{v}}, \vec{\mathcal{N}} \rangle = v_{int} + v_{ext}$$

where $v_{int}\vec{\mathcal{N}}$ is the velocity component along the normal direction that keeps the curve moving outward or inward and keep the curves smooth. $v_{ext}\vec{\mathcal{N}}$ is the velocity component along the normal direction computed from image data which is in the reversed direction as $v_{int}\vec{\mathcal{N}}$ to stop the evolving curves at the boundaries.

There are many possible choices for the speed function v_{int}, as has been shown in Fig. 2 [9]. Usually, a constant speed would often lead to singularities in shape, while curvature dependent speed would smooth the shape of the evolving contour [4]–[6]. In particular, if one chooses $v_{int} = -\kappa$, then a number of interesting properties of the curve evolution (3.2) can be described. For example, shapes of the moving contours would smooth to round points (i.e., points that dilate to circles) without developing any singularities or self-intersections [4]–[6]. In view of the above discussion, we propose to consider

$$v_{int} = f(\kappa)$$

a smooth function of the curvature.

Remark 1: The choice of $f(\kappa) = \alpha_0 + \alpha_1\kappa$, where α_0, α_1 are constants, refers to a tradeoff between reaction and diffusion (see [10] and [11] for details). For this choice of $f(\kappa)$ [Fig. 2(c)], the contour is not only getting smooth because of the speed component $\alpha_1\kappa$ which has the same regularization effect on the evolving contour as the internal deformation energy term in snake models [1], but also moving to the boundaries because of the speed component α_0 which is analogous to the inflation force defined in [29].

In order to make the evolving curves stop at the boundaries, we can take

$$v_{ext} = -\gamma f(\kappa)P(I) \tag{3.4}$$

where

$$P(I) = \begin{cases} 1, & \text{if } \|\nabla G_\sigma * I\| > T_0 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

and where $\gamma \geq 1$ is a constant chosen usually close to 1. The constant T_0 is a threshold value. Thus, The differential equation (3.2) describing the evolution of the contour can now be written as

$$\frac{\partial C(p,\, t)}{\partial t} = (1 - \gamma P(I))f(\kappa)\vec{\mathcal{N}}. \tag{3.6}$$

Remark 2: In comparison to the models (2.1), we have defined a new stopping function $g(\|\nabla G_\sigma * I\|) = 1 - \gamma P(I)$ in model (3.6). This new stopping function will cause the net speed of the moving contour to be precisely zero on and around the boundary based on the choice of $P(I)$. The term $v_{ext} = -\gamma f(\kappa)P(I)$ is analogous to the external force defined in [1].

Remark 3: The threshold T_0 can be used to determine what kind of edges would be detected and what kind of weak edges or spurious edges would be skipped. In some cases, we have to make a trade-off between weak edges and noises. The simplest way to choose T_0 is to fix them by hand at some value, but it may be necessary to try several times to get to the proper value. Another way is based on the histogram of the edge image or directly checking the intensity values of the edge image at some typical pixels.

In order to compute the evolution of (3.6) by using level set method, this equation can be converted to evolve the embedding function $u(x,\, y,\, t)$ according to [7]–[9]

$$\frac{\partial u}{\partial t} = (1 - \gamma P(I))f(\kappa)\|\nabla u\|. \tag{3.7}$$

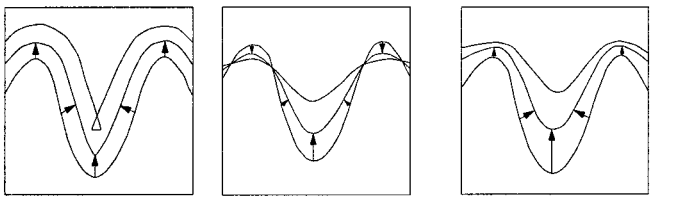


Fig. 2. Speed dependent on curvature. Here $\kappa > 0$ for convex shape, and $\kappa < 0$ for concave shape.

Since the level set algorithm is an accurate and stable algorithm, it can handle the topology changes due to the splitting and merging of multiple contours in a natural way [2], [3], [16], [19].

Remark 4: Although not explicitly detailed in this paper, the proposed models (3.6) and (3.7) can be generalized to three dimensional images as well. In this case, the function u in (3.7) is chosen as $u = u(x,\, y,\, z,\, t)$ and κ is chosen to be the mean curvature of the evolving surface.

Next, we will discuss the relationship between (3.6) and the previous models (2.1) and (2.4). In comparison to model (2.1), let us take $\gamma = 1$, $f(\kappa)$ is separated into a constant term F_A and a remainder $F_G(\kappa)$, i.e., $f(\kappa) = F_A + F_G(\kappa)$ in (3.6), then model (3.6) is similar to model (2.1) where $g(\cdot)$ is replaced by $(1 - P)$ and $F(\kappa)$ is replaced by $f(\kappa)$. But in (3.6) the speed function considered would cause the net speed of the moving contour to approach exact zero on the boundary. Additionally, model (3.6) can also detect boundaries whose gradients suffer from large variations, since we have normalized the gradient image such that it is a binary image by considering a threshold value in (3.5). In order to compare the model (3.6) with (2.4), let us take $v_{int} = f(\kappa) = \nu + \kappa$, $v_{ext} = -(\nu + \eta + \kappa)P(I)$, where $0 \leq \eta \leq \sqrt{2}$. Since on a step edge we have $g(\|\nabla G_\sigma * I\|) = 1/(1 + \|\nabla G_\sigma * I\|^m) \approx 0$, it follows that $(\partial C/\partial t) \approx -(\nabla g \cdot \vec{\mathcal{N}})\vec{\mathcal{N}}$ in model (2.4). However in model (3.6), $\partial C/\partial t = (v_{int} + v_{ext})\vec{\mathcal{N}} = -\eta\vec{\mathcal{N}}$, because $1 - P(I) = 0$ on the edge. Thus $\nabla g \cdot \vec{\mathcal{N}}$ is to be replaced by a constant η in our model, which in view of the inequality $0 \leq \|\nabla g \cdot \vec{\mathcal{N}}\| \leq \sqrt{2}$ on step edges is a reasonable replacement. The problem is that in the model (2.4), the term $\nabla g \cdot \vec{\mathcal{N}}$ is dependent on the normal vector and is affected by the shape of the contour. Thus different shape of the initial contour and subsequently the evolving contour would affect the convergence of the algorithm which is not the case in (3.6) in view of the fact that η is chosen to be an *a priori* constant.

Furthermore, the choice of the constant ν in the speed term of the model (2.1) and (2.4) is not a trivial issue. In fact since in either of the models, the term $g(\cdot)$ is not exactly zero on the edge, the magnitude of ν determines the overall speed of the evolving contour. In practice, this may lead to the risk of overshooting some edges. On the other hand in model (3.6), the choice of the constant term α_0 [in the expression $f(\kappa) = \alpha_0 + \alpha_1\kappa$] is arbitrary and independent of edges and doesn't lead to overshooting provided α_0 is not too large.

IV. Adaptive Multigrid Narrow Band Algorithm

In this section, we would present an algorithm to solve partial differential equation (3.7). One numerical approach to this problem is to discretize the parameterization p of the evolving contour $C(p,\, t)$ by a set of markers. These discrete markers are updated in time by approximating the spatial derivatives in (3.7), and advancing their positions. There are several difficulties with this approach as discussed in [7], for example, it cannot handle topological changes in the moving curve. An alternative to choose markers for the moving contour is provided by the level set technology introduced in [8]. Using this approach, a complex

curve can be studied. Sharp corners and cusps are handled naturally, and changes in the topology of the moving contour require no additional effort. Furthermore, these methods work in any number of space dimensions.

The central idea in the level set approach is to represent a moving curve/surface by the level set of a higher dimensional hypersurface, for instance, for the evolving curves $C(p, t)$, one would construct a smooth and Lipschitz continuous function $\phi(\boldsymbol{x}, t)$: $\mathcal{R}^2 \times [0, T] \to \mathcal{R}$, $\phi(\boldsymbol{x}, 0) = C(p, 0) = C_0(p)$, such that $C(p, t) = \{\boldsymbol{x} | \phi(\boldsymbol{x}, t) = 0\}$. $C(p, t)$ is called the zero level set of the time varying function $\phi(\boldsymbol{x}, t)$. Usually, $\phi(\boldsymbol{x}, t)$ is defined as a distance function of point $\boldsymbol{x}$ to the zero level set with negative in the interior and positive at the exterior of the zero level set or vice versa.

Let us now consider an evolution equation

$$\phi_t + F(\kappa)\|\nabla\phi\| = 0 \tag{4.1}$$

$$\text{with } \phi(\boldsymbol{x}, t = 0) \text{ given} \tag{4.2}$$

where κ is the curvature of the moving curve. In general, the function F can be split into two components: $F = F_A + F_G$. The term F_A is a constant term and F_G is the remaining term which depend on the curvature of the moving curve. We can rewrite (4.1) as

$$\phi_t + F_A\|\nabla\phi\| + F_G\|\nabla\phi\| = 0. \tag{4.3}$$

The numerical scheme for (4.3) using level set method is described as follows [8], [9]:

$$\begin{aligned}\phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t F_A((\min(D_-^x\phi_{ij}^n, 0))^2 \\ + (\max(D_+^x\phi_{ij}^n, 0))^2 + (\min(D_-^y\phi_{ij}^n, 0))^2 \\ + (\max(D_+^y\phi_{ij}^n, 0))^2)^{1/2} - \Delta t F_G\|\nabla\phi\|\end{aligned} \tag{4.4}$$

where

$$D_+^x\phi_{ij}^n = \frac{\phi_{i+1j}^n - \phi_{ij}^n}{\Delta x} \quad D_-^x\phi_{ij}^n = \frac{\phi_{ij}^n - \phi_{i-1j}^n}{\Delta x}$$
$$D_+^y\phi_{ij}^n = \frac{\phi_{ij+1}^n - \phi_{ij}^n}{\Delta y} \quad D_-^y\phi_{ij}^n = \frac{\phi_{ij}^n - \phi_{ij-1}^n}{\Delta y}.$$

The final term $F_G\|\nabla\phi\|$ can be approximated using a straightforward central difference approximation, Δt is the time step, Δx and Δy are the space discretization steps.

Since the speed function is only given on the moving curve, i.e., zero level set, the construction of an appropriate speed function for the entire domain that identifies with the speed function of the zero level set is known as extension problem. In fact, the most difficult part of the level set method is this extension problem. One possible way to extend the speed function from the moving curve to the grid points is that at each grid point $(i\Delta x, j\Delta y)$ on the image plane, one needs to find the closest point on the zero level set, and define the speed function at $(i\Delta x, j\Delta y)$ to be equal to the speed of this closest point. One efficient scheme is to use the "narrow band method," wherein, the curve can be moved by updating the level set function at a small set of points in the neighborhood of the zero level set instead of updating it at all the points on the image grid [16], [25], [26]. This scheme can save the computational labor significantly [26].

Since the speed function extension needs to find the closest point on the zero level set for each image grid point, this step usually takes most of the computing time of level set algorithm. In order to make the "closest point" search easier, we rebuild a new grid on the image based on the zero level set (curve) using interpolation instead of original image grid. In this way, we can find the "closest point" without doing any search. The configuration of our adaptive multigrid narrow band algorithm is shown in Fig. 3.

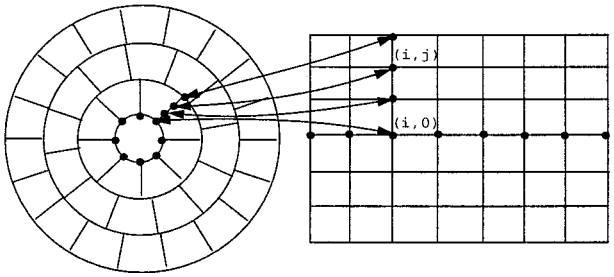


Fig. 3. Configuration of the algorithm.

Algorithm

Step 0: Draw an initial curve which is a closed, nonintersecting curve (polygon) on the image.

Step 1: For the initial curve and for each adjacent pair of nodes on the initial curve, interpolate the points between these two nodes so that the nodes on the initial curve are connected one by one and no gaps exist between any pairs of the nodes on the initial curve.

Step 2: Generate an $N \times N$ array (N is the size of the image) with all entries are N except that the entries on the interpolated curve are zeros. Now, use the algorithm for the distance transformations [27] to this generated array to get the distances for each entry on the array to the initial curve (only need to find the distances to the initial curve less than a specified amount "*Dist*").

Step 3: Since the interpolated curve is a polygon, use the polygon filling algorithm [28] to find the coordinates inside the initial curve and then change the sign of the distances inside or outside of the initial curve. This way, we can get a signed distance array $\Psi(x, y)$.

Step 4: Use this signed distance array $\Psi(x, y)$ to calculate the normal unit vector along the initial curve

$$\vec{n} = \frac{\nabla\Psi}{\|\nabla\Psi\|}$$

and the curvature

$$\kappa = \operatorname{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right) = \frac{\Psi_{\mathrm{xx}}\Psi_{\mathrm{y}}^2 - 2\Psi_{\mathrm{x}}\Psi_{\mathrm{y}}\Psi_{\mathrm{xy}} + \Psi_{\mathrm{yy}}\Psi_{\mathrm{x}}^2}{(\Psi_{\mathrm{x}}^2 + \Psi_{\mathrm{y}}^2)^{3/2}}.$$

Step 5: In order to decrease the computational labor, for a given $l_{\max}$, choose one node every $l_{\max}$ nodes along the interpolated curve. Let us suppose that we have chosen L nodes from the interpolated curve. For each of these L nodes, use simple interpolation

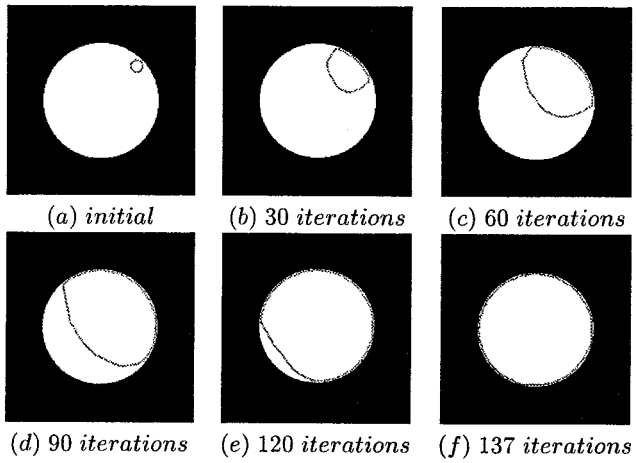


Fig. 4. Disk image and the segmentation results using proposed method, $l_{\max} = 5$.

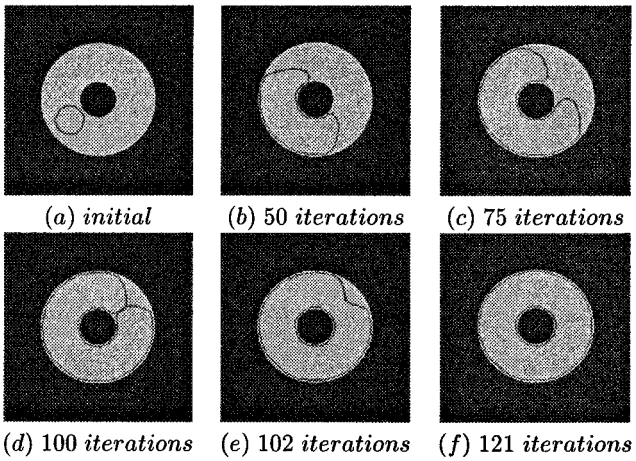


Fig. 5. Annulus image degraded by Gaussian noise with SNR = 12.7 dB and the segmentation results using proposed method, $l_{\max} = 5$.

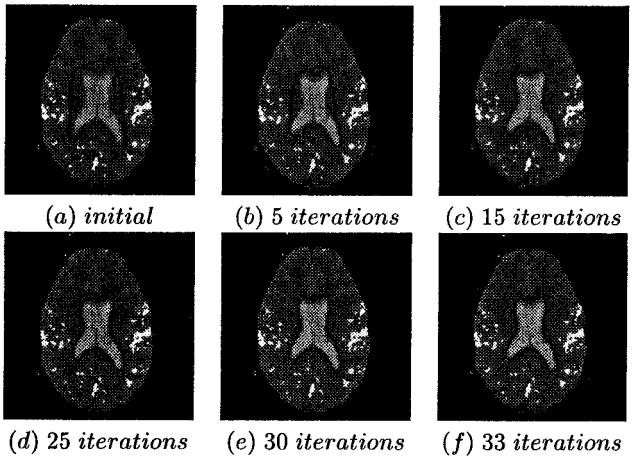


Fig. 6. Brain image and the segmentation results using proposed method, $l_{\max} = 5$.

to get a line segment along the normal direction based on the calculated unit normal vector in *Step 4.* The node point on the interpolated curve is the center point of this segment, and the length of the half of the segment is less than or equal to "*Dist*" in *Step 2.* Thus, we have got a new array ϕ_{ij} whose columns are composed of the values of $\Psi(x, y)$ on the normal segments using interpolation.

Step 6: Apply the level set iterating equation to the array ϕ_{ij} from n to n+q as follows: Taking $f(\kappa) = \alpha + \beta\kappa$ in (3.7) and using the algorithm in (4.4), we can obtain following algorithm:

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t\alpha(\hat{P}_I)_{ij}\{(\max(D_x^- \phi_{ij}^n, 0))^2 + (\min(D_x^+ \phi_{ij}^n, 0))^2 + (\max(D_y^- \phi_{ij}^n, 0))^2 + (\min(D_y^+ \phi_{ij}^n, 0))^2\}^{1/2} - \Delta t\beta\kappa(\hat{P}_I)_{ij}\|\nabla\phi_{ij}^n\|$$

where $1 \le q \le$ *Dist*, $\hat{P}_I$ is the extension of the $(1 - \gamma P(I))$ off the zero level set.

Step 7: Find the new zero level set curve which is the points on each column of ϕ_{ij}^n where the entry is zero, or adjacent entries change the sign on the column. If no zero entry and no sign changes on some columns, it may correspond to a splitting or merging of the zero level set.

After we have obtained the new zero level set, compare this new zero level set with previous one, if the distance between them are less than a threshold value, then stop. Otherwise, take this new zero level set curve as the new initial curve, then go to *Step 1.*

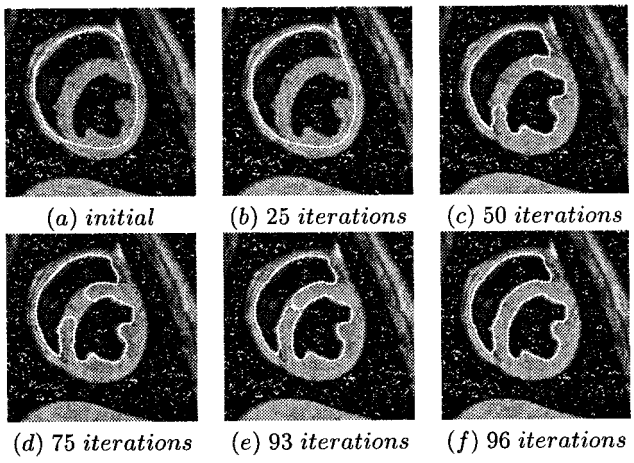


Fig. 7. Heart ventricles and the segmentation results using proposed method, $l_{\max} = 5$.

Remark 5: In comparison to the algorithm in [16] and [26], the proposed adaptive multigrid narrow band algorithm has following new features.

- Since we have built the grids in such a way that the transverse grid lines are the line segments on the normal direction, it follows that for each point on the grid, the closest point on the zero level set is the point of intersection of the transverse grid line with the moving contour (i.e., the zero level set). Therefore, we do not need any additional search in order to solve the extension

problem. For example, the closest point to the point (i, j) on the zero level set is $(i, 0)$ (as shown in Fig. 3).

- The grids for moving the contour has been built adaptively using interpolation, thus we don't need to search the distance from each grid point to zero level set since the distance is precisely the value of the array ϕ_{ij} at the corresponding coordinates of this grid point. For example, at point (i, j) in Fig. 3, $\phi_{ij} = j$ is the distance from (i, j) to zero level set.

To summarize, the narrow band algorithm proposed in this section, has a distinct computational advantage compared to the existing implementations of the algorithm in the literature. This stems from the fact that we have built a different grid from the original image grid plane so that the computation to find the closest point and its distance from a specific grid point to the zero level set has become trivial.

V. Experimental Results

In this section, we propose to apply the adaptive multigrid narrow band algorithm in Section IV to image data for shape segmentation. For all experiments in this section, we take $\Delta t = 0.5$, $q = 2$ in *Step 5* of Section IV.

First, let us go back to Fig. 1 in Section II. Fig. 1(c) shows the segmentation result for $f(\kappa) = 1 - 0.05\kappa$, $\gamma = 1$ in model (3.6), $k = 7$, $T_0 = 100$. After 18 iterations, the evolving contour stop moving. Here, we found the right boundaries were detected by the new geometric active contour model in Section III and the algorithm in Section IV.

Fig. 4(a) shows the image and initial contour; Fig. 4(b)–(e) show the evolution process of the moving contour; and Fig. 4(f) shows the final result. $f(\kappa) = 1 - 0.05\kappa$, $\gamma = 1$. After iterating 137 times, the evolving contour stops moving, and is stable on the boundaries. We didn't find the evolving contour crossing over object boundaries. However, the experimental results reported in [21] show that if applying model (2.1) to the image and initial contour shown in Fig. 4, the evolving contour would pass over the boundary.

Fig. 5 shows an annulus image with intensity 255 on annulus and zero on the background. This image is degraded by Gaussian noise with SNR = 12.7 dB. Fig. 5(a) shows the initial contour; Fig. 5(b)–(e) show the evolving contour at 50, 75, 100, 102, respectively; and Fig. 5(f) shows the final evolution result. $f(\kappa) = 1-0.05\kappa$, $\gamma = 1+2\times 10^{-5}$. The evolving contour splits automatically at the 101th iteration.

Figs. 6 and 7 depict the segmentation of the MR brain and heart ventricle images respectively. In Fig. 6(a) shows the initial contour; Fig. 6(b)–(e) show evolution of the evolving contour at 5, 15, 25, 30, respectively; and Fig. 6(f) shows the final result. $f(\kappa) = 1-0.1\kappa$, $\gamma = 1+2\times 10^{-5}$. In Fig. 7, $f(\kappa) = 1-0.05\kappa$, $\gamma = 1+2\times 10^{-5}$, Fig. 7(a) shows the initial contour which envelops the left and right ventricles; Fig. 7(b)–(e) show evolution of the evolving contour at 25, 50, 75, 93, respectively; and Fig. 7(f) shows the final result. The evolving contour splits automatically at the 94th iteration.

In all experiments above, we took $\gamma = 1 + \delta$, $\delta = 0$ or $\delta = 2 \times 10^{-5}$, a very small number, this can relieve or eliminate the oscillations when the moving contour approaches the edges, since very small speed $-\delta f(\kappa)$ cannot move the points on the evolving contour back to the interior of the interest region. We have applied the model (2.1) and (2.4) to the images shown in Figs. 5–7. The experimental results show that model (2.1) suffers from overshooting problem, model (2.4) can be used to detect the boundaries of objects in Figs. 5–7, however, the speed of convergence of model (3.7) are better than that of model (2.4) as discussed in Sections II and III.

VI. Conclusion

To conclude, in this paper we have implemented a new stopping criterion for the evolution of an active contour and a narrow band adaptive multigrid algorithm to apply the level set method. Together, we have proposed a new approach to shape recognition with improved speed of convergence, ability to remain stable near the boundary and ability to handle changes in the topology of the contour. The proposed approach has been demonstrated experimentally with synthetic and real images and comparisons with two existing *geometric active deformable models* have been made.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, pp. 321–331, 1988.

[2] V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numer. Math.*, vol. 66, pp. 1–31, 1993.

[3] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Evolutionary fronts for topology-independent shape modeling and recovery," in *Proc. 3rd ECCV, Lecture Notes Computer Science*, vol. 800, Stockholm, Sweden, May 1994, pp. 3–13.

[4] M. Gage and R. S. Hamilton, "The heat equation shrinking convex plane curves," *J. Diff. Geom.*, vol. 23, pp. 69–96, 1986.

[5] M. A. Grayson, "The heat equation shrinks embedded plane curves to round points," *J. Diff. Geom.*, vol. 26, pp. 285–314, 1987.

[6] ——, "Shortening embedded curves," *Ann. Math.*, vol. 129, pp. 71–111, 1989.

[7] J. A. Sethian, "Curvature and the evolution front," *Commun. Math. Phys.*, vol. 101, pp. 487–499, 1985.

[8] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations," *J. Comput. Phys.*, vol. 79, pp. 12–49, 1988.

[9] J. A. Sethian, "Numerical algorithms for propagating interfaces: Hamilton–Jacobi, equations and conservation laws," *J. Diff. Geom.*, vol. 31, pp. 131–161, 1990.

[10] B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "Toward a computational theory of shape: An overview," in *Lecture Notes in Computer Science*. New York, NY: Springer, 1990, vol. 427, pp. 402–407.

[11] ——, "Shapes, shocks, and deformations—I: The components of two-dimensional shape and the reaction-diffusion space," *Int. J. Comput. Vis.*, vol. 15, pp. 189–224, 1995.

[12] G. Sapiro, R. Kimmel, D. Shared, B. B. Kimia, and A. M. Bruckstein, "Implementing continuous-scale morphology via curve evolution," *Pattern Recognit.*, vol. 26, pp. 1363–1372, 1993.

[13] G. Sapiro and A. Tannenbaum, "Affine invariant scale-space," *Int. J. Comput. Vision*, vol. 11, pp. 25–44, 1993.

[14] ——, "On affine plane curve evolution," *J. Func. Anal.*, vol. 119, pp. 79–120, 1994.

[15] ——, "Area and length preserving geometric invariant scale-space," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 67–72, 1995.

[16] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 158–175, 1995.

[17] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contours," in *Proc. ICCV'95*, Cambridge, MA, June 1995.

[18] ——, "Conformal curvature flows: From phase transitions to active vision," *Arch. Ration. Mech. Anal.*, vol. 134, pp. 275–301, 1996.

[19] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.*, vol. 22, pp. 61–79, 1997.

[20] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, "Minimal surfaces based object segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 394–398, 1997.

[21] H. Tek and B. B. Kimia, "Volumetric segmentation of medical images by three-dimensional bubbles," *Comput. Vis. Image Understand.*, vol. 65, pp. 246–258, 1997.

[22] R. Kimmel, A. Amir, and A. M. Bruckstein, "Finding shortest paths on surfaces using level sets propagation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 635–640, 1995.
[23] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ: Prentice-Hall, 1976.
[24] C. L. Epstein and M. Gage, *The Curve Shortening Flow, Wave Motion: Theory, Modeling, and Computation*, A. Chorin and A. Majda, Eds. New York: Springer, 1987.
[25] D. Chopp, "Computing minimal surfaces via level set curvature flow," *J. Comput. Phys.*, vol. 106, pp. 77–91, 1993.
[26] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *J. Comput. Phys.*, vol. 118, pp. 269–277, 1995.
[27] G. Borgefors, "Distance transformations in arbitrary dimensions," *Comput. Vis., Graph., Image Process.*, vol. 27, pp. 321–345, 1984.
[28] J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide*. Reading, MA: Addison-Wesley, 1993.
[29] L. D. Cohen, "On active contour models and balloons," *Comput., Vis., Graph., Image Process.: Image Understand.*, vol. 53, pp. 211–218, 1991.

Estimation of Generalized Mixture in the Case of Correlated Sensors

Wojciech Pieczynski, Julien Bouvrais, and Christophe Michel

***Abstract*—This paper deals with unsupervised Bayesian classification of multidimensional data. We propose an extension of a recent method of generalized mixture estimation to correlated sensors case. The method proposed is valid in the independent data case, as well as in the hidden Markov chain or field model case, with known applications in signal processing, particularly speech or image processing. The efficiency of the method proposed is shown via some simulations concerning hidden Markov fields, with application to unsupervised image segmentation.**

***Index Terms*—Bayesian classification, image segmentation, Markov fields, mixture estimation, multisensor data.**

I. Introduction

The aim of this paper is to deal with the following problem. We are faced with m series of real data produced by m sensors. For each sensor $1 \leq j \leq m$ the data are denoted by $y_1^j, \cdots, y_n^j$. We assume that for each point $1 \leq s \leq n$ the data $y_s^1, \cdots, y_s^m$ correspond to a certain class, among k classes $\omega_1, \cdots, \omega_k$, and the problem is to find which class it is. In other words, the problem is to classify each point $1 \leq s \leq n$ from the data available. The probabilistic approach, which will be our approach in this paper, consists in assuming that the class of the point $1 \leq s \leq n$ is a realization of a random variable X_s, and the data $y_s^1, \cdots, y_s^m$ produced by the m sensors are a realization of a random vector $Y_s = (Y_s^1, \cdots, Y_s^m)$. Thus the problem is to estimate the unobserved realizations of a random process $X = (X_1, \cdots, X_n)$ from the observed realization of a random process $Y = (Y_1, \cdots, Y_n)$. Different methods of such a statistical classification exist once the distribution $P_{(X,Y)}$ of (X, Y) is known. When $P_{(X,Y)}$ is not known, one has to identify it from $Y = y$, the only data available. The aim of our paper is to generalize to correlated sensors the method proposed in [8].

Manuscript received June 23, 1998; revised February 16, 1999. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Robert J. Schalkoff.

The authors are with the Département Signal et Image, Institut National des Télécommunications, 91000 Evry, France (e-mail: wojciech.pieczynski@int-evry.fr).

Publisher Item Identifier S 1057-7149(00)01257-4.

Let us first consider the case of one sensor. When $P_{(X,Y)}$ depends on an unknown parameter θ, the problem is to estimate θ from Y. This problem, which is known as the mixture estimation problem, is a very general and important one [17]. The pioneering method of mixture estimation is the expectation-maximization (EM) algorithm [6], [16], which admits theoretical justifications and gives very good results in classical cases, such as independent Gaussian mixtures. Other methods like stochastic gradient [18] or iterative conditional estimation (ICE, [12]) can also be used. In particular, their use in the Markov random field model (MFR) context leads to the unsupervised image segmentation [2], [9], [18], among others. In fact, once the parameters are estimated, the segmentation can be performed with simulated annealing [7], maximum posterior mode (MPM [11]), or iterated conditional mode (ICM [1]).

All these methods are easily generalizable to the multi-sensor case when the noise is Gaussian. When the noise is not Gaussian and the sensors are independent, one may use the ICE-general mixture (ICE-GEMI) algorithm, valid in the following context [8]. We have k classes, and so we have to find the k probability densities $f_1, \cdots, f_k$ on R^m. Because of the independence, each of these densities f_i is written

$$f_i(y_1^1, y_1^2, \cdots, y_1^m) = f_i^1(y_1^1) f_i^2(y_1^2) \cdots f_i^m(y_1^m). \tag{1.1}$$

ICE-GEMI allows one to find the form of the km functions f_i^j, and estimate their parameters, once we know that each f_i^j belongs to a given set of forms. For instance, in the case of three classes and two sensors, in which each component can be exponential or Gaussian, there are sixty-four possibilities and ICE-GEMI makes possible to search what case the data lie in. In this paper, we propose the following generalization of ICE-GEMI: each f_i of the densities $f_1, \cdots, f_k$ is searched in the set of possible densities of the distribution of a random vector $Y_i = A_i Z_i$, where A_i is $m \times m$ matrix and Z_i a random vector with independent components. Roughly speaking, we add matrices $A_1, \cdots, A_k$ making one possible to deal with correlated sensors. The distribution of Z_i is thus given by

$$g_i(z_1^1, z_1^2, \cdots, z_1^m) = g_i^1(z_1^1) g_i^2(z_1^2) \cdots g_i^m(z_1^m) \tag{1.2}$$

where each g_i^j belongs to a given set of forms. The density of the distribution of Y_i is then written

$$\begin{aligned} f_i(y_1^1, y_1^2, \cdots, y_1^m) = (\det A_i) g_i^1(a_{11}^i y_1^1 + \cdots + a_{1m}^i y_1^m) \\ \cdots g_i^m(a_{m1}^i y_1^1 + \cdots + a_{mm}^i y_1^m). \end{aligned} \tag{1.3}$$

We can see how (1.3) generalizes (1.1). Finally, the method we propose allows one to find the form of each g_i^j, estimate its parameters, and estimate the k matrices $A_1, \cdots, A_k$. It is valid in the independent data case, as well as in the hidden Markov chain or field model case. The efficiency of the method proposed is shown via some simulations concerning hidden Markov fields, with application to unsupervised image segmentation.

The organization of the paper is as follows. In the next section we specify the assumptions needed and describe the run of the proposed method. Section III is devoted to simulation results. The final section contains some concluding remarks and perspectives.

II. Generalized Correlated Sensors Mixture Estimation

Let $X = (X_s)_{s \in S}, Y = (Y_s)_{s \in S}$, be two random processes, where each X_s takes its values in the set of classes $\Omega = \{\omega_1, \cdots, \omega_k\}$ and each Y_s takes its values in the set of observations R^m. The distribution