Fast Implicit Active Contour Models

Gerald Kühne[1], Joachim Weickert[2], Markus Beier[1], and Wolfgang Effelsberg[1]

[1] Praktische Informatik IV, Universität Mannheim,
L 15, 16, D-68131 Mannheim, Germany
{kuehne,beier,effelsberg}@informatik.uni-mannheim.de
[2] Fakultät für Mathematik und Informatik, Universität des Saarlandes,
Geb. 27.1, Postfach 15 11 50
D-66041 Saarbrücken, Germany
weickert@mia.uni-saarland.de

Abstract. Implicit active contour models are widely used in image processing and computer vision tasks. Most implementations, however, are based on explicit updating schemes and are therefore of limited computational efficiency. In this paper, we present fast algorithms based on the semi-implicit additive operator splitting (AOS) scheme for both the geometric and the geodesic active contour model. Our experimental results with synthetic and real-world images demonstrate that one can gain a speed up by one order of magnitude compared to the widely used explicit time discretization.

1 Introduction

Level-set-based or so-called implicit active contour models have been used in a variety of image processing and computer vision tasks [17]. Their main advantages over classical explicit snakes [6] are implicit handling of topological changes, numerical stability and independence of parametrization. However, their main drawback is the additional computational complexity. In their simplest implementation, most approaches are based on an explicit or forward Euler scheme which demands very small time steps.

To solve this problem, we provide a fast algorithm using an semi-implicit additive operator splitting (AOS) technique [9,20]. Our approach is suitable both for the geometric [3,10] and the geodesic active contour model [8,4].

The remainder of this paper is organized as follows: Section 2 introduces the geometric and the geodesic active contour model. Section 3 describes our numerical implementation of both models based on the AOS scheme. Section 4 presents results and computation times for the different implementations. Finally, Section 5 concludes the paper.

Related work. A number of fast implementations for implicit snakes have been proposed. However, most of them concentrate on narrow-band techniques and multi-scale computations [1,14,13]. In [5], Goldenberg et al. present for geodesic active contours an AOS strategy in combination with a narrow-band technique. However, in contrast to our implementation, each iteration in their approach requires a reinitialization step.

L. Van Gool (Ed.): DAGM 2002, LNCS 2449, pp. 133–140, 2002.

2 Implicit Active Contour Models

Active contour models (deformable models) are used in a variety of image processing tasks, e. g., image segmentation or object tracking. The basic idea is that the user specifies an initial guess of an interesting contour (e. g. an organ, a tumour, or a person to be tracked). Then, this contour evolves under smoothness control (internal energy) and image-driven forces (external energy) to the boundaries of the desired object.

In the classical *explicit* snake model [6] the contour represented by a closed planar parametric curve $\mathcal{C}_0(s) = (x(s), y(s)), s \in [0, 1]$, is embedded into an energy minimization framework. Apart from energy minimization the parametric curve can also evolve directly under motion equations derived from geometric considerations [16].

However, the parametrization of the curve causes difficulties with respect to topological changes and numerical implementations. Thus, to prevent these difficulties *implicit* active contour models have been developed. Here, the basic idea is to represent the inital curve $\mathcal{C}_0(s)$ *implicitly* within a higher dimensional function, and to evolve this function under a partial differential equation. Usually, $\mathcal{C}_0$ is embedded as a zero level set into a function $u_0 : \mathbb{R}^2 \to \mathbb{R}$ by using the *signed distance function*:

$$u_0(x) = \begin{cases} d(x, \mathcal{C}_0), & \text{if } x \text{ is inside } \mathcal{C}_0 \\ 0, & \text{if } x \text{ is on } \mathcal{C}_0 \\ -d(x, \mathcal{C}_0), & \text{if } x \text{ is outside } \mathcal{C}_0, \end{cases} \tag{1}$$

where $d(x, \mathcal{C}_0)$ denotes the distance from an arbitrary position to the curve.

The *implicit geometric active contour model* discovered by Caselles et al. [3] and later on by Malladi et al. [10] includes geometrical considerations similar to [16]. Let $\Omega := (0, a_x) \times (0, a_y)$ be our image domain in $\mathbb{R}^2$. We consider a scalar image on Ω. Then, using the level set technique [12] the model reads

$$\begin{aligned} \frac{\partial u}{\partial t} &= g(x)|\nabla u| \left(\operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + k \right) && \text{on} \quad \Omega \times (0, \infty) \\ u(x, 0) &= u_0(x) && \text{on} \quad \Omega. \end{aligned} \tag{2}$$

Here, k is a constant force term comparable to the balloon force known from explicit models and $g : \mathbb{R}^2 \to (0, 1]$ denotes a stopping function that slows down the snake as it approaches selected image features, e. g., edges. Note that normal and curvature to a level set are given by

$$n = -\frac{\nabla u}{|\nabla u|}, \quad \kappa = \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) = \frac{u_{xx} u_y^2 - 2 u_x u_y u_{xy} + u_{yy} u_x^2}{(u_x^2 + u_y^2)^{3/2}}. \tag{3}$$

In the *implicit geodesic active contour model* proposed simultaneously by Caselles et al. [4] and Kichenassamy et al. [8] the function u is embedded into

an energy functional inspired by the explicit snake model:

$$\begin{aligned} \frac{\partial u}{\partial t} &= |\nabla u| \left(\operatorname{div} \left(g(x) \frac{\nabla u}{|\nabla u|} \right) + k g(x) \right) \quad \text{on} \quad \Omega \times (0, \infty) \\ u(x, 0) &= u_0(x) \qquad\qquad\qquad\qquad\quad \text{on} \quad \Omega. \end{aligned} \tag{4}$$

3 Numerical Implementation

While implicit active contour models avoid several of the difficulties known from explicit models, their main disadvantage is additional computational complexity. First, in their simplest implementation, the partial differential equation must be evaluated on the complete image domain. Second, most approaches are based on explicit updating schemes which demand very small time steps. While the first limitation can be addressed by narrow-band and/or multi-scale techniques [1,17,14], the latter requires different discretizations. In the following we focus on the second problem and develop semi-implicit schemes for both the geometric and the geodesic active contour model based on an additive operator splitting (AOS) technique [9,20]. Note that narrow-band and multi-scale techniques can be easily combined with our implementation.

Let us consider the following equation, which unifies the geometric and the geodesic model by introducing two additional functions a and b:

$$\frac{\partial u}{\partial t} = a(x) |\nabla u| \operatorname{div} \left(\frac{b(x)}{|\nabla u|} \nabla u \right) + |\nabla u| k g(x). \tag{5}$$

Setting $a := g$, $b := 1$ yields the geometric model, while $a := 1$, $b := g$ results in the geodesic model. For the sake of clarity we assume a constant force $k = 0$ in the following and discuss the complete model later on. Interpreting the term $\frac{b(x)}{|\nabla u|}$ as "diffusivity" we can employ techniques similar to those as described in [20] in the context of nonlinear diffusion filtering.

To provide a numerical algorithm one has to consider discretizations of space and time. We employ discrete times $t_n := n\tau$, where $n \in \mathbb{N}_0$ and τ denotes the time step size. Additionally, an image is divided by a uniform mesh of spacing $h = 1$ into grid nodes x_i. To simplify the notation a discrete image is represented in the following by a vector $f \in \mathbb{R}^N$, whose components $f_i, i \in \{1, \ldots, N\}$, contain the pixel values. Consequently, pixel i corresponds to some grid node x_i. Thus, using standard notation, u_i^n denotes the approximation of $u(x_i, t_n)$. Hence, following [20], (5) with $k = 0$ reads in its semi-implicit formulation as

$$u_i^{n+1} = u_i^n + \tau \left(a_i |\nabla u|_i^n \sum_{j \in \mathcal{N}(i)} \frac{\left(\frac{b}{|\nabla u|} \right)_i^n + \left(\frac{b}{|\nabla u|} \right)_j^n}{2} \frac{u_j^{n+1} - u_i^{n+1}}{h^2} \right), \tag{6}$$

where $\mathcal{N}(i)$ denotes the 4-neighborhood of the pixel at position x_i. Here, straightforward finite difference implementations would give rise to problems when $|\nabla u|$ vanishes in the 4-neighborhood. These problems do not appear if one

uses a finite difference scheme with *harmonic* averaging [19], thus replacing $\frac{1}{2}\left(\left(\frac{b}{|\nabla u|}\right)_i^n + \left(\frac{b}{|\nabla u|}\right)_j^n\right)$ in (6) by its harmonic counterpart:

$$u_i^{n+1} = u_i^n + \tau\left(a_i|\nabla u|_i^n \sum_{j\in\mathcal{N}(i)} \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_j^n} \frac{u_j^{n+1} - u_i^{n+1}}{h^2}\right). \tag{7}$$

Note that by evaluating only image positions with $|\nabla u|_i \neq 0$, the denominator in this scheme cannot vanish. In matrix-vector notation this becomes

$$u^{n+1} = u^n + \tau\left(\sum_{l\in\{x,y\}} A_l(u^n)\right) u^{n+1}, \tag{8}$$

where A_l describes the interaction in l direction. In detail, the matrix $A_l(u^n) = (\hat{a}_{ijl}(u^n))$ is given by

$$\hat{a}_{ijl}(u^n) := \begin{cases} a_i|\nabla u|_i^n \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_j^n}, & j \in \mathcal{N}_l(i) \\ -a_i|\nabla u|_i^n \sum\limits_{m\in\mathcal{N}_l(i)} \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_m^n}, & j = i \\ 0, \text{ else}, \end{cases} \tag{9}$$

where $\mathcal{N}_l(i)$ represents the neighboring pixels with respect to direction $l \in \{x, y\}$. However, the solution u^{n+1} cannot be directly determined from this scheme. Instead, it requires to solve a linear system of equations. Its solution is formally given by

$$u^{n+1} = \left(I - \tau \sum_{l\in\{x,y\}} A_l(u^n)\right)^{-1} u^n \tag{10}$$

where I denotes the unit matrix. Reformulating (10) using an AOS approximation yields

$$u^{n+1} = \frac{1}{2} \sum_{l\in\{x,y\}} \left(I - 2\tau A_l(u^n)\right)^{-1} u^n. \tag{11}$$

The operators $B_l(u^k) := I - 2\tau A_l(u^n)$ come down to strictly diagonally dominant tridiagonal linear systems which can be solved very efficiently [11]. Moreover, this scheme is unconditionally stable, thus, we can apply arbitrarily large time steps.

However, we have so far neglected the constant force term $|\nabla u|kg$ (cf. (5)). This term stems from the hyperbolic dilation/erosion equation $\partial_t u = \pm|\nabla u|$. Consequently, in a numerical implementation the gradient has to be approxi-

mated by an upwind scheme [12]:

$$|\nabla u|_i^n \approx \begin{cases} |\nabla^- u|_i^n = \big(\max(D^{-x}u_i^n,0)^2 + \min(D^{+x}u_i^n,0)^2 + \\ \qquad\qquad \max(D^{-y}u_i^n,0)^2 + \min(D^{+y}u_i^n,0)^2\big)^{1/2}, \text{ if } k \le 0 \\ |\nabla^+ u|_i^n = \big(\min(D^{-x}u_i^n,0)^2 + \max(D^{+x}u_i^n,0)^2 + \\ \qquad\qquad \min(D^{-y}u_i^n,0)^2 + \max(D^{+y}u_i^n,0)^2\big)^{1/2}, \text{ if } k > 0 \end{cases}, \tag{12}$$

where D^{+x}, D^{+y}, D^{-x}, and D^{+y} denote forward and backward approximations of the spatial derivatives (see e. g. [17]). Integrating the constant force term into (11) is straightforward and yields for $k < 0$:

$$u^{n+1} = \frac{1}{2} \sum_{l \in \{x,y\}} \left(I - 2\tau A_l(u^n)\right)^{-1} \left(u^n + \tau |\nabla^- u|^n k g\right). \tag{13}$$

Since the dilation/erosion equation approximated on a grid with $h = 1$ is stable only for $\tau \le 0.5$ [12], the constant force term limits the applicable time step. Consequently, (13) is stable only for $|\tau k g| \le 0.5$. However, since g is bounded by one, k is usually a small fraction of 1.0, and very large time steps ($\tau > 5.0$) degrade accuracy significantly [20,19], this constraint is not severe.

4 Experimental Results

With the AOS-based implementation it is possible to choose time steps much larger than in explicit updating schemes. Consequently, the evolution of the contour to its final location requires only a small number of iterations compared to explicit algorithms. However, a semi-implicit AOS iteration is more expensive than its explicit counterpart. In order to compare both approaches, we implemented the AOS-based models according to (13). For the explicit scheme we employed standard techniques [17,2]. In addition, we used a stopping criterion to indicate that the curve has reached a stable steady state. Every time a certain period Δt_k has elapsed, the average gray value of the evolving image u is calculated. E. g., when setting $\Delta t_k = 50$ and $\tau = 0.25$, the average gray value is computed every 200 iterations. The process stops if two consecutive measurements differ by less than an accuracy parameter α. In all experiments the parameters for the stopping criterion were set to $\Delta t_k = 50$ and $\alpha \in \{0.01, 0.1\}$.

To assess the final placement of the contour with regard to the underlying algorithm, a simple distance measure was developed. Given a result contour and a reference contour, we calculated for each pixel on the result contour the distance to the nearest pixel on the reference contour. Averaging these values yields the distance between the two contours. As reference contour we used in all cases the explicit implementation with a small time step $\tau = 0.1$.

We applied both algorithms to sample images (cf. Figures 1–3). A stopping function according to the Perona-Malik diffusivity [15] was used:

$$g(x) = \frac{1}{1 + |\nabla f_\sigma(x)|^2/\lambda^2}, \tag{14}$$

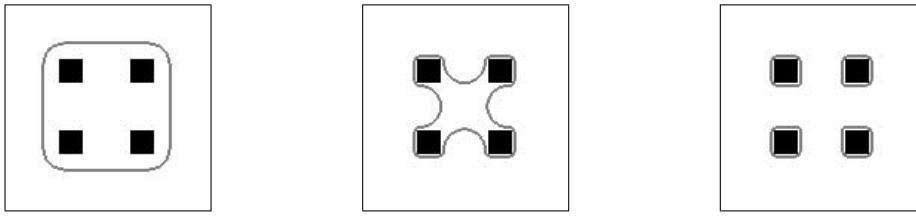

Fig. 1. AOS-based geometric active contour model on a synthetic image (size 128×128, $\tau = 5.0$, $k = -0.1$, $\sigma = 0.5$, $\lambda = 1$). From left to right: 10, 150, 250 iterations.

Fig. 2. AOS-based geodesic active contour model on hall-and-monitor image (size 352×240, $\tau = 5.0$, $k = -0.02$, $\sigma = 0.5$, $\lambda = 1$). From left to right: 100, 500, 1000 iterations.

Fig. 3. AOS-based geometric active contour model on medical image (size 284×284, $\tau = 5.0$, $k = -0.1$, $\sigma = 1$, $\lambda = 1.5$). From left to right: 50, 150, 300 iterations.

where f_σ denotes the convolution of image f with a Gaussian kernel of standard deviation σ and λ is a contrast factor. While close to edges (high gradient magnitudes) of the image f, the stopping function approaches zero, it reaches one in flat image areas (low gradient magnitudes). To extract the person in the hall-and-monitor sequence we replaced the gradient term in the above equation by the results of a motion detector [7]. In each case the image u_0 was intialized to a signed distance function [18,14,17] from a mask that covered nearly the complete image domain. Table 1 summarizes the results calculated on a standard personal computer with 1.4 GHz. As expected, the AOS-based implementation reduced the number of iterations on the average by a factor of 20. Due to the coarse stopping criterion the reduction varies from 18 to 22. Furthermore, we observe that an AOS-based iteration is about twice as expensive, and in some cases three times as expensive as an explicit iteration. Combining those results, we observe that using AOS-based implementations of implicit active contour models yields

Table 1. Comparison of explicit and AOS-based schemes

geometric model (explicit scheme)					
image	τ	k	iterations	CPU time	distance (pixels)
synthetic image	0.25	-0.1	20200	49.0 s	0
hall-and-monitor image	0.25	-0.1	20000	324.5 s	0
medical image	0.25	-0.1	6600	126.3 s	0.01
geometric model (AOS scheme)					
image	τ	k	iterations	CPU time	distance (pixels)
synthetic image	5.0	-0.1	950	7.4 s	0.75
hall-and-monitor image	5.0	-0.1	1040	54.0 s	0.87
medical image	5.0	-0.1	370	25.0 s	0.48
geodesic model (explicit scheme)					
image	τ	k	iterations	CPU time	distance (pixels)
synthetic image	0.25	-0.02	10400	36.9 s	0
hall-and-monitor image	0.25	-0.02	30800	634.9 s	0
medical image	0.25	-0.05	12200	306.1 s	0.01
geodesic model (AOS scheme)					
image	τ	k	iterations	CPU time	distance (pixels)
synthetic image	5.0	-0.02	480	4.2 s	1
hall-and-monitor image	5.0	-0.02	1390	70.2 s	1.79
medical image	5.0	-0.05	640	36.8 s	1.32

a significant speedup. In our examples the speedup ranges from a factor of 5 to a factor of 9. Additionally, we applied the simple distance measure to the final contours of the AOS-based and the explicit algorithms. The distance column in Table 1 shows the average distance (in pixels) of the contours to the reference contour obtained by an explicit algorithm with $\tau = 0.1$. In all cases the results indicate that the accuracy of the final placement is sufficient with respect to the underlying segmentation task. We should note that the accuracy might be further improved by refining the simple stopping criterion.

5 Conclusions

We have presented fast algorithms for both the geometric and the geodesic active contour model. Our implementation based on the additive operator splitting scheme outperforms other widely used explicit updating schemes clearly. Future work will comprise the integration of narrow-band and multi-scale techniques and should further improve the computational efficiency.

References

1. D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.

2. G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, volume 147 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2002.
3. V. Caselles, F. Catt , T. Coll, and F. Dibois. A geometric model for active contours. *Numerische Mathematik*, 66:1–31, 1993.
4. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.
5. R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *IEEE Transactions on Image Processing*, 10(10):1467–1475, October 2001.
6. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.
7. G. K hne, J. Weickert, O. Schuster, and S. Richter. A tensor-driven active contour model for moving object segmentation. In *Proc. IEEE International Conference on Image Processing (ICIP)*, volume II, pages 73–76, October 2001.
8. S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Conformal curvature flows: from phase transitions to active vision. *Archive of Rational Mechanics and Analysis*, 134:275–301, 1996.
9. T. Lu, P. Neittaanm ki, and X.-C. Tai. A parallel splitting up method and its application to Navier-Stokes equations. *Applied Mathematics Letters*, 4(2):25–29, 1991.
10. R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.
11. K. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations : An Introduction*. Cambridge University Press, Cambridge, UK, 1994.
12. S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
13. N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, March 2000.
14. D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.
15. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.
16. J. A. Sethian. Curvature and the evolution of fronts. *Communications in Mathematical Physics*, 101:487–499, 1985.
17. J. A. Sethian. *Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK, 1999.
18. M. Sussman, P. Smereka, and S. Osher. A level-set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114:146–159, 1994.
19. J. Weickert. Application of nonlinear diffusion in image processing and computer vision. *Acta Mathematica Universitatis Comenianae*, 70(1):33–50, 2001.
20. J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, March 1998.