Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint

Fast Methods for Implicit Active Contour Models

Joachim Weickert and Gerald Kühne

Preprint No. 61

Saarbrücken 2002

Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Fast Methods for Implicit Active Contour Models

Joachim Weickert

Universität des Saarlandes
Fakultät für Mathematik und
Informatik, Geb. 27.1
Postfach 15 11 50
66041 Saarbrücken
Germany
E-Mail:
weickert@mia.uni-saarland.de

Gerald Kühne

Universität Mannheim
Praktische Informatik IV
L 15, 16
68131 Mannheim
Germany
E-Mail:
kuehne@informatik.uni-mannheim.de

submitted: June 11, 2002

Preprint No. 61

Saarbrücken 2002

Edited by
FR 6.1 – Mathematik
Im Stadtwald
D–66041 Saarbrücken
Germany

Fax: + 49 681 302 4443
e-mail: preprint@math.uni-sb.de
WWW: http://www.math.uni-sb.de/

Abstract

Implicit active contour models belong to the most popular level set methods in computer vision. Typical implementations, however, suffer from poor efficiency. In this paper we survey an efficient algorithm that is based on an additive operator splitting (AOS). It is suitable for geometric and geodesic active contour models as well as for mean curvature motion. It uses harmonic averaging and does not require to compute the distance function in each iteration step. We prove that the scheme satisfies a discrete maximum-minimum principle which implies unconditional stability if no balloon forces are present. Moreover, it possesses all typical advantages of AOS schemes: simple implementation, equal treatment of all axes, suitability for parallel computing, and straightforward generalization to higher dimensions. Experiments show that one can gain a speed up by one order of magnitude compared to the widely used explicit time discretization.

AMS Subject Classification: 68T45, 68T10, 35K55, 35K65
Key words: computer vision, active contour models, mean curvature motion, splitting methods

Contents

1 Introduction

Active contour models (also called deformable models or snakes) [15] have been used in a variety of different image processing and computer vision tasks, ranging from interactive image segmentation to object tracking in image sequences. 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 is moved by image-driven forces to the boundaries of the desired object.

Implicit active contour models [7] constitute a very interesting applications of level set ideas within the active contour framework. They embed the active contour as a level set in a suitable image evolution that is determined by a partial differential equation (PDE). Then the final contour is extracted when the evolution is stopped. The main advantages of implicit active contours over classical explicit snakes are the automatic handling of topological changes, high 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 requires very small time steps. This severely limits their efficiency.

A number of fast implementations for implicit snakes have been proposed to circumvent this problem. Often they concentrate on narrow-band techniques and multi-scale computations [1, 29, 27], but more recently also methods based on additive operator splittings (AOS) have become popular [11, 12, 19, 28, 39]. It is the goal of this paper to give an introduction to AOS schemes for implicit active contour models and related PDEs.

AOS schemes has been introduced to image analysis in the context of nonlinear diffusion filtering [43]. They have been used for medical imaging problems [31], for regularization methods [40], image registration [10] and for optic flow computations [42]. The basic idea behind AOS schemes is to decompose a multi-dimensional problem into one-dimensional ones that can be solved very efficiently. Then the final multi-dimensional solution is approximated by averaging the one-dimensional solutions. AOS schemes perform well not only on sequential architectures: experiments have demonstrated that they are also well-suited for parallel computing on systems with shared [41] as well as distributed memory [6].

The usefulness of AOS ideas has also been shown in a number of other applications ranging from Navier–Stokes equations [37, 21] to sandpile growth simulations [14]. It seems that Navier–Stokes equations have constituted one of their historically first application domains.

The description in this paper follows our previous publications [19, 39, 43].

Our approach is suitable both for the geometric [7] and the geodesic active contour model [8, 16], and it may also be used for mean curvature motion [3, 18]. It differs from recent work by Goldenberg et al. [12, 11] by the fact that it does not require to recompute a distance transformation in each iteration step. This may lead to significant savings in computation time. Indeed, we shall see that – under realistic accuracy requirements – AOS schemes allow to speed up implicit active contour models by one order of magnitude.

The present 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 an evaluation of the accuracy and efficiency of the AOS scheme. Finally, Section 5 concludes the paper by giving a summary and mentioning extensions that can further increase the computational efficiency.

2 Implicit Active Contour Models

In active contour models one places a closed planar parametric curve $\mathcal{C}_0(s) = (x(s), y(s))\,, s \in [0,1]$, around image parts of interest. Then this curve evolves under smoothness control (internal energy) and the influence of an image force (external energy).

In the classical *explicit* snake model [15] the parametric curve 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 [34].

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*:

$$u_0(x) = \begin{cases} d(x, C_0), & \text{if } x \text{ is inside } C_0 \\ 0, & \text{if } x \text{ is on } C_0 \\ -d(x, C_0), & \text{if } x \text{ is outside } C_0, \end{cases} \tag{1}$$

where $d(x, C_0)$ denotes the distance between some point x and the curve C_0. The *implicit geometric active contour model* discovered by Caselles et al. [7] includes geometrical considerations similar to [34]. Let $\Omega := (0, a_x) \times (0, a_y)$ be our image domain in $\mathbb{R}^2$. We consider a scalar image $u_0(x)$ on Ω. Then, the geometric active contour model investigates the evolution of u_0 under the PDE

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

Here, k is a constant force term comparable to the balloon force [9] 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 such as edges. Note that normal and curvature to a level set are given by

$$n = -\frac{\nabla u}{|\nabla u|}, \qquad (3)$$

$$\kappa = \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) = \frac{u_{xx}u_y^2 - 2u_xu_yu_{xy} + u_{yy}u_x^2}{(u_x^2+u_y^2)^{3/2}}. \qquad (4)$$

In the *implicit geodesic active contour model* proposed simultaneously by Caselles et al. [8] and Kichenassamy et al. [16] the function u is embedded into an energy functional that can be related to the explicit snake model. The corresponding evolution equation is given by

$$\begin{aligned}\frac{\partial u}{\partial t} = |\nabla u| \left(\operatorname{div}\left(g(x)\frac{\nabla u}{|\nabla u|}\right) + kg(x)\right) \quad &\text{on} \quad \Omega \times (0,\infty),\\ u(x,0) = u_0(x) \qquad\qquad &\text{on} \quad \Omega. \qquad (5)\end{aligned}$$

Figure 1 gives an example of a geodesic active contour evolution.

3 Numerical Implementation

While implicit active contour models avoid several of the difficulties known from explicit models, their main disadvantage is poor efficiency. 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 require very small time steps. While the first limitation can be addressed by narrow-band and/or multi-scale techniques [1, 35, 29], 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 the additive operator splitting (AOS) scheme. Note that narrow-band and multi-scale techniques can be easily combined with our implementation.

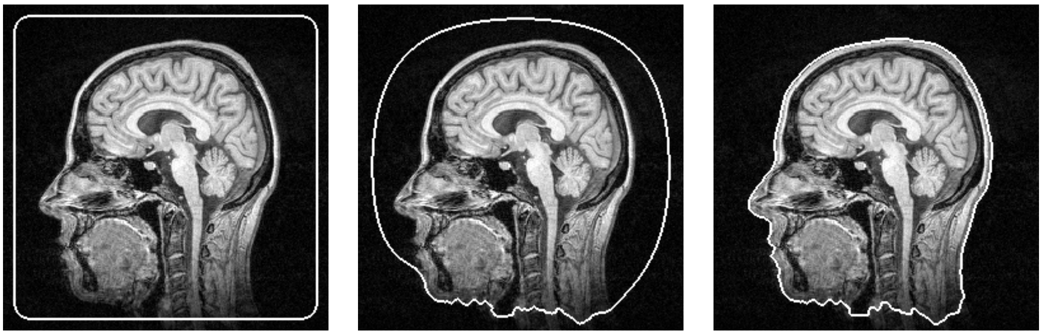

Figure 1: Temporal evolution of a geodesic active contour superimposed on the original image with $\Omega = (0, 256)^2$. From left to right: $t = 0$, 1500, 7500. From [39].

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

$$\frac{\partial u}{\partial t} = a(x)\,|\nabla u|\,\mathrm{div}\left(\frac{b(x)}{|\nabla u|}\nabla u\right) + |\nabla u|\,k\,g(x). \tag{6}$$

Setting $a := g$ and $b := 1$ yields the geometric model, while $a := 1$ and $b := g$ results in the geodesic model. Moreover, for $a := b := 1$ and $k := 0$, we obtain the mean curvature motion

$$\frac{\partial u}{\partial t} = |\nabla u|\,\mathrm{div}\left(\frac{\nabla u}{|\nabla u|}\right), \tag{7}$$

which plays an important role in image denoising and morphological scale-space analysis [2, 3, 17, 18].

Semi-Implicit Scheme. Now we are in a position to derive suitable numerical schemes for our unified model. For the sake of clarity we assume a constant force $k = 0$ in the following and discuss the integration of the balloon force later on. Interpreting the term $\frac{b(x)}{|\nabla u|}$ as "diffusivity" we can employ techniques similar to those as described in [43] in the context of nonlinear diffusion filtering.

To derive 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 (i, j). Using standard notation, u_{ij}^n denotes the approximation of $u(ih, jh, t_n)$.

Let us recall a simple spatial discretization of the term $\mathrm{div}\left(\frac{b}{|\nabla u|}\nabla u\right)$ since it is a prerequisite for our semi-implicit scheme. Setting $c := \frac{b}{|\nabla u|}$ the term may be approximated as follows:

$$\begin{aligned}
\operatorname{div}(c\nabla u) &\approx \partial_x \left(c_{ij} \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{h} \right) + \partial_y \left(c_{ij} \frac{u_{i,j+\frac{1}{2}} - u_{i,j-\frac{1}{2}}}{h} \right) \\
&\approx c_{i+\frac{1}{2},j} \frac{u_{i+1j} - u_{ij}}{h^2} - c_{i-\frac{1}{2},j} \frac{u_{ij} - u_{i-1j}}{h^2} \\
&\quad + c_{i,j+\frac{1}{2}} \frac{u_{ij+1} - u_{ij}}{h^2} - c_{i,j-\frac{1}{2}} \frac{u_{ij} - u_{ij-1}}{h^2} \qquad (8)
\end{aligned}$$

The values for $c_{i\pm 1/2,j}$ and $c_{i,j\pm 1/2}$ can be determined by linear interpolation. To simplify the notation in the following, a discrete image is represented further on by a vector $f \in \mathbb{R}^N$, whose components $f_i, i \in \{1, \ldots, N\}$, contain the pixel values. For instance, this vector might be constructed by concatenating the rows of an image. Consequently, pixel i corresponds to some grid node x_i. Thus, u_i^n denotes the approximation of $u(x_i, t_n)$. Hence, following [43] and using the above mentioned discretization, Equation 6 with $k = 0$ reads in its semi-implicit formulation as

$$u_i^{n+1} = u_i^n + \tau\, a_i \, |\nabla u|_i^n \sum_{j \in \mathcal{N}(i)} \frac{\left(\frac{b}{|\nabla u|}\right)_i^n + \left(\frac{b}{|\nabla u|}\right)_j^n}{2} \, \frac{u_j^{n+1} - u_i^{n+1}}{h^2}, \qquad (9)$$

where $\mathcal{N}(i)$ denotes the 4-neighbourhood of the pixel at position x_i. However, in order to compute Equation 9 one must assure that $|\nabla u|$ does not vanish in the 4-neighbourhood. Here, straightforward finite difference implementations would give rise to problems. These problems do not appear if one uses a finite difference scheme with *harmonic* averaging [39], thus replacing the arithmetic mean $\frac{1}{2}\left(\left(\frac{b}{|\nabla u|}\right)_i^n + \left(\frac{b}{|\nabla u|}\right)_j^n \right)$ in Equation 9 by its harmonic counterpart. This yields

$$u_i^{n+1} = u_i^n + \tau\, a_i \, |\nabla u|_i^n \sum_{j \in \mathcal{N}(i)} \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_j^n} \, \frac{u_j^{n+1} - u_i^{n+1}}{h^2}. \qquad (10)$$

Note that by evaluating only image positions with $|\nabla u|_i \neq 0$, the denominator in this scheme cannot vanish. If $|\nabla u|_i = 0$, one sets $u_i^{n+1} := u_i^n$. In general such an harmonic averaging scheme may turn out to be rather dissipative. For the specific application to active contour models, however, this

does not seem to create specific problems.
In matrix-vector notation, Equation 10 becomes

$$u^{n+1} = u^n + \tau \sum_{l \in \{x,y\}} A_l(u^n)\, u^{n+1}, \tag{11}$$

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

$$\hat{a}_{ijl}(u^n) := \begin{cases} a_i |\nabla u|_i^n \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_j^n}, & j \in \mathcal{N}_l(i) \\ -a_i |\nabla u|_i^n \sum\limits_{m \in \mathcal{N}_l(i)} \frac{2}{\left(\frac{|\nabla u|}{b}\right)_i^n + \left(\frac{|\nabla u|}{b}\right)_m^n}, & j = i \\ 0, & \text{else}, \end{cases} \tag{12}$$

where $\mathcal{N}_l(i)$ represents the neighbouring 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 the linear system of equations

$$\Big(I - \tau \sum_{l \in \{x,y\}} A_l(u^n)\Big) u^{n+1} = u^n. \tag{13}$$

where I denotes the unit matrix.
Since the system matrix is strictly diagonally dominant, it follows from Gerschgorin's theorem that it is invertible [38]. In practice, however, it may be rather expensive to solve such a linear system in the 2-D case. Since its number of unknowns coincides with the pixel number, it is typically a very large sparse system with at most five nonvanishing entries per row. Although the actual structure of the system matrix depends on the pixel numbering, it is not possible to order the pixels in such a way that in the i-th row all nonvanishing matix elements can be found within the positions $[i, i-2]$ to $[i, i+2]$: Usually, the matrix reveals a much larger bandwidth. Applying direct algorithms such as Gaussian elimination would destroy the zeros within the band and would lead to an immense storage and computation effort.
Hence, iterative algorithms should be applied. Classical methods like Gauß–Seidel or SOR iterations [45] do not need additional storage, and convergence can be guaranteed for the special structure of the system matrix. This convergence, however, may be rather slow since the condition number of the system matrix increases with the image resolution. Faster iterative methods such as preconditioned conjugate gradient methods [32] need significantly more storage, which can become prohibitive for very large images or 3-D problems. Iterative methods suffer also from the fact that their convergence

slows down for increasing τ, since this increases the condition number of the system matrix. Multigrid methods [5] appear to be one possibility to circumvent many of these problems, but their implementation is more complicated. In the following we shall focus on a splitting-based alternative. It is simple to implement and does not require to specify any additional parameters. This may make it attractive in a number of practical applications.

AOS Scheme. Instead of using the semi-implicit scheme

$$u^{n+1} = \Big(I - \tau \sum_{l \in \{x,y\}} A_l(u^n)\Big)^{-1} u^n \tag{14}$$

we may consider its *additive operator splitting (AOS)* variant

$$u^{n+1} = \frac{1}{2} \sum_{l \in \{x,y\}} \left(I - 2\tau A_l(u^n)\right)^{-1} u^n. \tag{15}$$

By means of a Taylor expansion it is easy to see that the semi-implicit scheme and its AOS version differ by an $\mathrm{O}(\tau^2)$ term. However, this does not create any problems since the time discretization in the semi-implicit scheme has already introduced an error of the same order. Hence, from an approximation viewpoint, both schemes have the same order of numerical consistency to the continuous equation.

The AOS scheme, however, offers one important advantage: The operators $B_l(u^k) := I - 2\tau A_l(u^n)$ lead to strictly diagonally dominant tridiagonal linear systems which can be solved very efficiently with a Gaussian algorithm (also called Thomas algorithm in this context [25]). This algorithm has linear complexity and can be implemented very easily. Details are given in the appendix.

In order to implement Equation 15, one proceeds in three steps:

1. Evolution in x direction with step size 2τ:
   Solve the tridiagonal system $(I - 2\tau A_x(u^n))\, v^{n+1} = u^n$ for v^{n+1}.

2. Evolution in y direction with step size 2τ:
   Solve the tridiagonal system $(I - 2\tau A_y(u^n))\, w^{n+1} = u^n$ for w^{n+1}.

3. Averaging:
   Compute $u^{n+1} := 0.5\,(v^{n+1} + w^{n+1})$.

The fact that AOS schemes are based on an *additive* splitting guarantees that both axes are treated in exactly the same manner. This is in contrast to conventional splitting techniques from the literature such as ADI methods,

D'yakonov splitting or LOD techniques [22, 24, 44]: they are *multiplicative* and may produce different results in the nonlinear setting if the image is rotated by 90 degrees, since the operators do not commute.
One should notice that AOS schemes are also well-suited for parallel computing as they possess two granularities of parallelism:

- Coarse grain parallelism: The evolution in different directions can be performed simultaneously on different processors.
- Mid grain parallelism: Each 1-D evolution decouples into independent evolutions along each row or column.

AOS schemes are not only efficient, they are also unconditionally stable. This can be seen as follows: Since $B_l(u^k)$ is strictly diagonally dominant, $b_{iil} > 0$ for all i and $b_{iil} \leq 0$ for $i \neq j$, we may conclude from [23, p. 192] that B_l^{-1} is nonnegative in all its arguments. Moreover, from the fact that all row sums of $A_l(u^n)$ vanish, it follows that $B_l(u^n)$ and $Q(u^n) := \frac{1}{2}\sum_l B_l^{-1}(u^n)$ have row sums 1. Together with the nonnegativity this implies that the AOS scheme $u^{n+1} = Q(u^n)u^n$ computes u^{n+1} from convex combinations of the elements of u^n. This guarantees the discrete maximum–minimum principle

$$\min_j u_j^n \leq u_i^{n+1} \leq \max_j u_j^n \qquad \forall\, i. \tag{16}$$

which implies stability of the scheme in the maximum norm for all time step sizes τ.
In practice, it makes of course not much sense to use extremely large time steps, since the accuracy will deteriorate significantly and splitting artifacts may become visible. Experiments show that, if a spatial grid size of 1 is used and if $a(x)b(x) \leq 1$, a time step size of $\tau = 5$ is a good compromise between accuracy and efficiency.

Supplementing the balloon force. So far we have neglected the constant force term $|\nabla u|kg$ (cf. Equation 6). This term stems from the hyperbolic dilation/erosion equation $\partial_t u = \pm|\nabla u|$. Consequently, it is advantageous to approximate the gradient by an upwind scheme [26]:

$$|\nabla u|_i^n \approx \begin{cases} |\nabla^- u|_i^n = & (\max(D^{-x}u_i^n, 0)^2 + \min(D^{+x}u_i^n, 0)^2 + \\ & \max(D^{-y}u_i^n, 0)^2 + \min(D^{+y}u_i^n, 0)^2)^{1/2}\,, \text{ if } k \leq 0 \\ & \\ |\nabla^+ u|_i^n = & (\min(D^{-x}u_i^n, 0)^2 + \max(D^{+x}u_i^n, 0)^2 + \\ & \min(D^{-y}u_i^n, 0)^2 + \max(D^{+y}u_i^n, 0)^2)^{1/2}\,, \text{ if } k > 0 \end{cases}, \tag{17}$$

where D^{+x}, D^{+y}, D^{-x}, and D^{-y} denote forward resp. backward approximations of the spatial derivatives (see e. g. [35]). Integrating the constant force term into Equation 15 is straightforward and yields for $k < 0$:

$$u^{n+1} = \frac{1}{2} \sum_{l \in \{x,y\}} \left(I - 2\tau A_l(u^n)\right)^{-1} \left(u^n + \tau |\nabla^- u|^n kg\right). \tag{18}$$

Since the dilation/erosion equation approximated on a grid with size $h = 1$ can be shown [26] to be stable only for $\tau \leq 0.5$, the constant force term limits the applicable time step. Consequently, Equation 18 is stable only for $|\tau kg| \leq 0.5$. 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 the accuracy of the AOS scheme significantly, this constraint is not severe.

4 Experimental Results

In this section we evaluate the accuracy and the efficiency of AOS schemes for implicit active contour models. For the accuracy evaluation we focus on a specific example where an analytic solution is known, and efficiency is studied by comparing the semi-implicit AOS schemes to a corresponding explicit (Euler forward) discretization in time.

4.1 Accuracy Evaluation in Case of Mean Curvature Motion

In order to assess the numerical errors of our AOS scheme, we consider in our first experiment an evolution of a disk under mean curvature motion. Since it is well-known that a disk-shaped level set with area S(0) shrinks under mean curvature motion such that

$$S(t) = S(0) - 2\pi t, \tag{19}$$

simple accuracy evaluations are possible. To this end we use a distance transformation of some disk-shaped initial image and consider the evolution of a level set with an initial area of 24845 pixels. Table 1 shows the area errors for different time step sizes τ and two stopping times. We observe that for $\tau \leq 5$, the accuracy is sufficiently high for typical image processing applications. Figure 2 demonstrates that in this case no violations regarding rotational invariance are visible.

Table 1: Area errors for the evolution of a disk-shaped area of 24845 pixels under mean curvature motion using an AOS-based semi-implicit scheme with harmonic averaging. The pixels have size 1 in each direction. From [39].

step size τ	stopping time $T = 2250$	stopping time $T = 3600$
0.5	-0.27 %	-0.60 %
1	-0.26 %	-0.88 %
2	-0.27 %	-0.88 %
5	-0.34 %	-1.73 %
10	-5.18 %	51.20 %

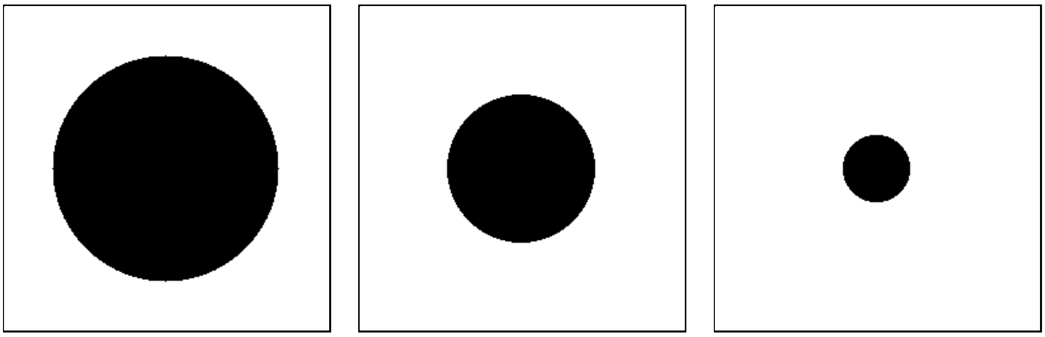

Figure 2: Temporal evolution of a disked shaped level set under mean curvature motion. The results have been obtained using an AOS-based semi-implicit scheme with harmonic averaging and step size $\tau = 5$. From left to right: $t = 0$, 2250, 3600. From [39].

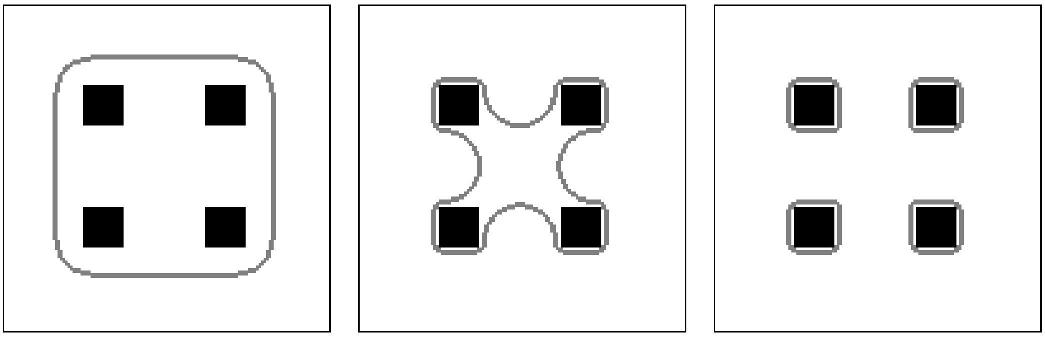

Figure 3: AOS-based geometric active contour model on a synthetic image (size 128 × 128, $\tau = 5.0$, $k = -0.1$, $\sigma = 0.5$, $\lambda = 1$). From left to right: 10, 150, 250 iterations. From [19].

4.2 Efficiency Gain for Implicit Active Contour Models

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 Equation 18. For the explicit model we employed standard techniques [35, 4]. In addition, we used a stopping criterion to indicate that the curve location has stabilized. 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 distances over all result contour pixels yields the average 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 3–5). A stopping

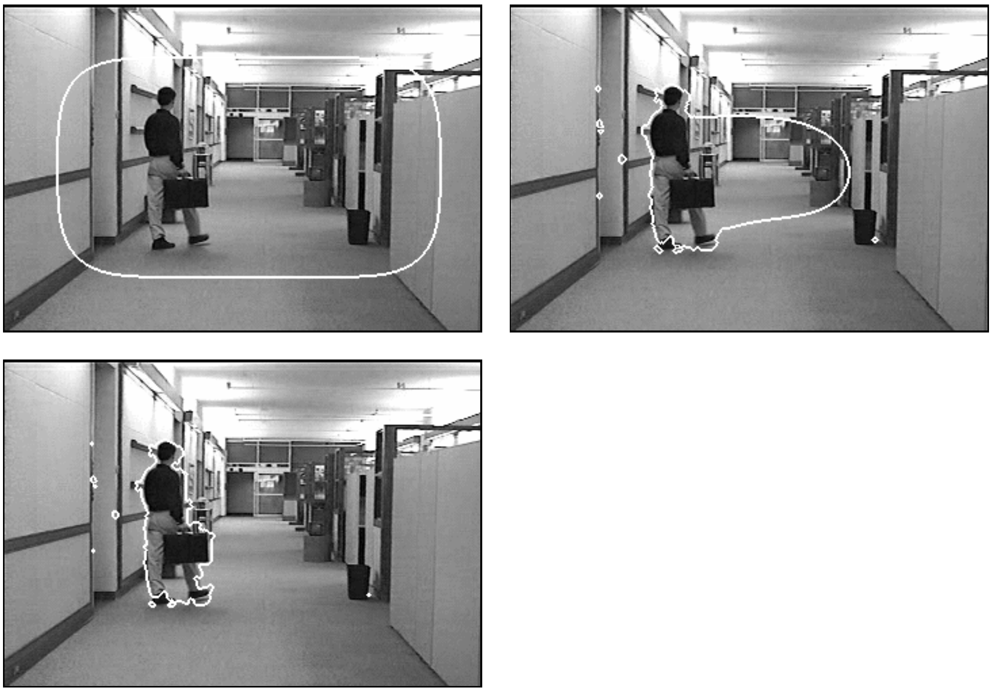

Figure 4: AOS-based geodesic active contour model on hall-and-monitor image (size 352 × 240, $\tau = 5.0$, $k = -0.02$, $\sigma = 0.5$, $\lambda = 1$). Top left: 100 iterations. Top right: 500 iterations. Bottom left: 1000 iterations. From [19].

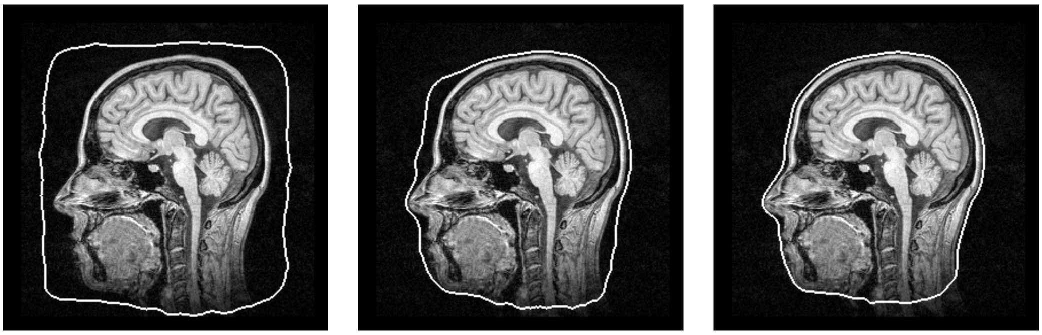

Figure 5: AOS-based geometric active contour model on medical image (size 284 × 284, $\tau = 5.0$, $k = -0.1$, $\sigma = 1$, $\lambda = 1.5$). From left to right: 50, 150, 300 iterations. From [19].

function according to the Perona-Malik diffusivity [30] was used:

$$g(x) = \frac{1}{1 + |\nabla f_\sigma(x)|^2/\lambda^2}, \tag{20}$$

where f_σ denotes the convolution of image f with a Gaussian kernel of standard deviation σ, and λ is a contrast factor. Close to edges (high gradient magnitudes) of the image f, the stopping function approaches 0, whereas it reaches 1 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 [20]. In each case the image u_0 was intialized to a signed distance function [36, 29, 35] from a mask that covered nearly the complete image domain. In addition, we did not employ any reinitialization procedure throughout the computations to prove the stability of our scheme. However, we should note that for certain applications it is necessary to maintain the signed distance function during curve evolution [36, 13].

Table 2 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 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 2 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 placements are 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

In this paper we have surveyed an additive operator splitting (AOS) algorithm for a class of PDEs that comprises mean curvature motion, geometric and geodesic active contour models. This algorithm uses harmonic averaging and it does not require any recomputations of the distance transformation in each iteration step.

Table 2: Comparison of explicit and AOS-based schemes. From [19].

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

AOS schemes have the same approximation order as their corresponding semi-implicit schemes. They come down to solving tridiagonal linear systems of equations which can be done in linear complexity with a very simple algorithm. If no balloon forces are present, then they are absolutely stable in the maximum norm. Under typical accuracy requirements one can gain a speed-up by one order of magnitude compared to the widely used explicit time discretization.

Further speed up may be possible by exploiting one of the following options:

- Implementing AOS schemes on a parallel system. Recent experiments with AOS-based nonlinear diffusion filtering on a PC cluster with 256 processors showed that speed up factors of 209 are possible [6].

- Embedding AOS schemes in a pyramid framework. This may yield an accelation by one order of magnitude [40].

- The integration of narrow-band techniques is another possibility for improving the computational efficiency [11].

Finally it should be mentioned that we have focussed on the 2-D case for didactical reasons only. It is straightforward to generalize all ideas in this paper to higher dimensions.

6 Appendix: The Thomas Algorithm

The semi-implicit scheme requires to solve a linear system, where the system matrix is tridiagonal and diagonally dominant. The most efficient way to achieve this goal is the so-called *Thomas algorithm*, a Gaussian elimination algorithm for tridiagonal systems. It can be found in many textbooks on numerical analysis, e.g. [33, pp. 43–45]. However, since it builds the backbone of our AOS algorithm and since we want to keep this paper selfcontained, we survey its algorithmic features here.
The principle is as follows. Suppose we want to solve a tridiagonal linear system $Bu = d$ with an $N \times N$ matrix

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ & & & \gamma_{N-1} & \alpha_N \end{pmatrix}. \tag{21}$$

Then the Thomas algorithm consists of three steps.

Step 1: LR decomposition.
We decompose B into the product of a lower bidiagonal matrix

$$L = \begin{pmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{N-1} & 1 \end{pmatrix} \tag{22}$$

and an upper bidiagonal matrix

$$R = \begin{pmatrix} m_1 & r_1 & & \\ & \ddots & \ddots & \\ & & m_{N-1} & r_{N-1} \\ & & & m_N \end{pmatrix} \tag{23}$$

Comparing the coefficients shows that $r_i = \beta_i$ for all i, and m_i and l_i can be obtained as follows:

```
$m_1 := \alpha_1$
for $i = 1, 2, ..., N-1$:
    $l_i := \gamma_i / m_i$
    $m_{i+1} := \alpha_{i+1} - l_i \beta_i$
```

Solving $LRu = d$ for u is done in two steps:

Step 2: Forward substitution.
We solve $Ly = d$ for y. This gives

```
$y_1 := d_1$
for $i = 2, 3, ..., N$:
    $y_i := d_i - l_{i-1} y_{i-1}$
```

Step 3: Backward substitution.
We solve $Ru = y$ for u. This leads to

```
$u_N := y_N / m_N$
for $i = N-1, N-2, ..., 1$:
    $u_i := (y_i - \beta_i u_{i+1}) / m_i$
```

This completes the Thomas algorithm. It is stable for every strictly diagonally dominant system matrix. One may also regard it as a recursive filtering: The LR decomposition determines the filter coefficients, Step 2 is a causal filter and Step 3 an anticausal one. The whole scheme is very efficient: it requires only

$$2(N-1) + (N-1) + 1 + 2(N-1) \;=\; 5N - 4 \qquad (24)$$

multiplications/divisions, and

$$(N-1) + (N-1) + (N-1) \;=\; 3N-3 \qquad (25)$$

subtractions. Hence the CPU effort is *linear* in N. The same holds for the memory requirement.

References

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

[2] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, *Axioms and fundamental equations in image processing*, Archive for Rational Mechanics and Analysis, 123 (1993), pp. 199–257.

[3] L. Alvarez, P.-L. Lions, and J.-M. Morel, *Image selective smoothing and edge detection by nonlinear diffusion. II*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 845–866.

[4] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, vol. 147 of Applied Mathematical Sciences, Springer, New York, 2002.

[5] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, second ed., 2000.

[6] A. Bruhn, T. Jacob, M. Fischer, T. Kohlberger, J. Weickert, U. Brüning, and C. Schnörr, *Designing 3-D nonlinear diffusion filters for high performance cluster computing*, in Pattern Recognition, L. Van Gool, ed., Lecture Notes in Computer Science, Springer, Berlin, 2002. To appear.

[7] V. Caselles, F. Catté, T. Coll, and F. Dibos, *A geometric model for active contours in image processing*, Numerische Mathematik, 66 (1993), pp. 1–31.

[8] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, International Journal of Computer Vision, 22 (1997), pp. 61–79.

[9] L. D. Cohen, *On active contour models and balloons*, CVGIP: Image Understanding, 53 (1991), pp. 211–218.

[10] B. Fischer and J. Modersitzki, *Fast diffusion registration*, Tech. Rep. A-01-18, Institute of Mathematics, Medical University of Lübeck, Germany, 2001. To appear in Contemporary Mathematics.

[11] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, *Cortex segmentation – a fast variational geometric approach*, in Proc. First

IEEE Workshop on Variational and Level Set Methods in Computer Vision, Vancouver, Canada, July 2001, IEEE Computer Society Press, pp. 127–133.

[12] ——, *Fast geodesic active contours*, IEEE Transactions on Image Processing, 10 (2001), pp. 1467–1475.

[13] J. Gomes and O. Faugeras, *Reconciling distance functions and level sets*, Journal of Visual Communication and Image Representation, 11 (2000), pp. 209–223.

[14] C. Herbe, *Numerical methods for nonlinear diffusion models of sandpile growth*, Master's thesis, Department of Mathematics, University of Kaiserslautern, Germany, Jan. 1999.

[15] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, International Journal of Computer Vision, 1 (1988), pp. 321–331.

[16] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, *Conformal curvature flows: from phase transitions to active vision*, Archive for Rational Mechanics and Analysis, 134 (1996), pp. 275–301.

[17] B. B. Kimia and K. Siddiqi, *Geometric heat equation and non-linear diffusion of shapes and images*, Computer Vision and Image Understanding, 64 (1996), pp. 305–322.

[18] B. B. Kimia, A. Tannenbaum, and S. W. Zucker, *On the evolution of curves via a function of curvature. I. The classical case*, Journal of Mathematical Analysis and Applications, 163 (1992), pp. 438–458.

[19] G. Kühne, J. Weickert, M. Beier, and W. Effelsberg, *Fast implicit active contour models*, in Pattern Recognition, L. Van Gool, ed., Lecture Notes in Computer Science, Springer, Berlin, 2002. To appear.

[20] G. Kühne, J. Weickert, O. Schuster, and S. Richter, *A tensor-driven active contour model for moving object segmentation*, in Proc. 2001 IEEE International Conference on Image Processing, vol. 2, Thessaloniki, Greece, Oct. 2001, pp. 73–76.

[21] T. Lu, P. Neittaanmäki, and X.-C. Tai, *A parallel splitting up method and its application to Navier–Stokes equations*, Applied Mathematics Letters, 4 (1991), pp. 25–29.

[22] G. I. Marchuk, *Splitting and alternating direction methods*, in Handbook of Numerical Analysis, P. G. Ciarlet and J.-L. Lions, eds., vol. I, North Holland, Amsterdam, 1990, pp. 197–462.

[23] T. Meis and U. Marcowitz, *Numerische Behandlung partieller Differentialgleichungen*, Springer, Berlin, 1978.

[24] A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations*, Wiley, Chichester, 1980.

[25] K. W. Morton and L. M. Mayers, *Numerical Solution of Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1994.

[26] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.

[27] 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 (2000), pp. 266–280.

[28] N. Paragios, O. Mellina-Gottardo, and V. Ramesh, *Gradient vector flow fast geodesic active contours*, in Proc. Eighth International Conference on Computer Vision, vol. 1, Vancouver, Canada, July 2001, IEEE Computer Society Press, pp. 67–73.

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

[30] P. Perona and J. Malik, *Scale space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 629–639.

[31] J. Rexilius, *Anisotrope nichtlineare Diffusion für die Bildverarbeitung*, Tech. Rep. B-99-07, Institute of Mathematics and Computer Science, Medical University of Lübeck, Germany, Dec. 1999.

[32] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Thomson Learning, London, 1996.

[33] H. R. Schwarz, *Numerische Mathematik*, Teubner, Stuttgart, 1988.

[34] J. A. Sethian, *Curvature and the evolution of fronts*, Communications in Mathematical Physics, 101 (1985), pp. 487–499.

[35] ——, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, second ed., 1999.

[36] M. Sussman, P. Smereka, and S. Osher, *A level-set approach for computing solutions to incompressible two-phase flow*, Journal of Computational Physics, 114 (1994), pp. 146–159.

[37] R. Temam, *Sur l'approximation de la solution des équations de Navier–Stokes par la méthode de pas fractionnaires (I)*, Archive for Rational Mechanics and Analysis, 32 (1969), pp. 135–153.

[38] R. A. Varga, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, 1962.

[39] J. Weickert, *Applications of nonlinear diffusion in image processing and computer vision*, Acta Mathematica Universitatis Comenianae, 70 (2001), pp. 33–50.

[40] ——, *Efficient image segmentation using partial differential equations and morphology*, Pattern Recognition, 34 (2001), pp. 1813–1824.

[41] J. Weickert, J. Heers, C. Schnörr, K. J. Zuiderveld, O. Scherzer, and H. S. Stiehl, *Fast parallel algorithms for a broad class of nonlinear variational diffusion approaches*, Real-Time Imaging, 7 (2001), pp. 31–45.

[42] J. Weickert and C. Schnörr, *Variational optic flow computation with a spatio-temporal smoothness constraint*, Journal of Mathematical Imaging and Vision, 14 (2001), pp. 245–255.

[43] 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 (1998), pp. 398–410.

[44] N. N. Yanenko, *The Method of Fractional Steps: the Solution of Problems of Mathematical Physics in Several Variables*, Springer, New York, 1971.

[45] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.