A Multigrid Approach for Fast Geodesic Active Contours

A. Kenigsberg[*], R. Kimmel and I. Yavneh
Department of Computer Science, Technion - I.I.T, Haifa 32000, Israel
aviko||ron||irad||@cs.technion.ac.il
Phone: +972-4-8294898, Fax: +972-4-8293900
EDICS: 2-MOTD, 2-SEGM

November 4, 2004

Abstract

The geodesic active contour is a recent geometric approach for image segmentation, which is motivated by previous snake and geometric models. Segmentation in this model is performed by a dynamic curve which minimizes several internal and external forces. These forces smooth the curve and attract it to the boundaries of objects. The conventional framework for computing geodesic active contours is the level-set method, where the evolving contour is represented as a level-set of a surface. This gives a stable solution, which naturally handles segmentation of several objects in one image. To overcome the relatively high computational requirements of this approach, an implicit formulation is proposed, which reduces the required number of time-steps drastically. An efficient adaptive multigrid algorithm is developed and implemented for the solution of the resulting nonlinear system.

1 Introduction

Image segmentation is a basic and important task in computer vision. High quality segmentation plays a key role in higher level algorithms such as medical image analysis, pattern recognition, and tracking of objects in a sequence of images. In this paper, an innovative approach is introduced for solving the geodesic active contour flow by an adaptive multigrid method.

1.1 Geodesic Active Contour Review

The geodesic active contour, introduced in [1], is a geometric version of the snake model, which proved to be an accurate and robust segmentation and

[*]This work was first presented at the Copper Mountain Conference on Multigrid Methods, April, 2001, and won a best student paper award.

edge integration technique. The snake model, also known as the active contour, first introduced by Kass, Witkin, and Terzopoulos (1987) [2], is an energy minimizing curve guided by external constraint forces, which draw it towards features such as lines and edges. An important addition is the balloon force, introduced by Cohen (1991) [3], which accelerates the evolution of the snake far from the object's boundaries. It also allows the curve to skip over weak edges. A geometric version of the active contour model was introduced by Caselles *et al.* (1993) [4] and Malladi *et al.* (1993, 1995) [5, 6]. It is independent of the curve's parameterization, and thus intrinsic and stable. The present work is based on the geodesic active contour, introduced by Caselles, Kimmel and Sapiro (1995, 1997) [7, 1]. The geodesic approach to object segmentation links between the snake model and the geometric active contours. Minimal distance curves are computed using the Osher and Sethian level-set approach (1988) [8]. This method treats evolving curves implicitly, by embedding the propagating interface as a level-set of a higher-dimensional function. In our case, the curve is embedded as an equal height contour of a surface. The curve's flow is computed by a numerical algorithm on a fixed grid. This approach easily handles segmentation of several objects, since, even if the curve splits, its level-set implicit surface representation remains continuous.

Recently, the geodesic active contour flow was solved by Goldenberg *et al.* [9]. They use an unconditionally stable Additive Operator Splitting (AOS) numerical scheme [10] to implement a fast version of the geodesic active contour model. It is applied in small regions, motivated by the Adalsteinsson-Sethian level-set narrow band approach [11], and uses a fast marching method for re-initialization.

1.2 Multigrid Review

The multigrid idea was introduced by Fedorenko (1964) [12] and generalized by Bakhvalov (1966) [13]. The first practical algorithms were introduced by Brandt in the early 70's [14]. In [15], Brandt developed the method and introduced a nonlinear multigrid algorithm (FAS), adaptive techniques (MLAT), a discussion of general domains, a systematic application of the nested iteration idea (FMG), and a general theoretical analysis. Subsequently, multigrid methods have become popular worldwide for solving problems in various fields. [16] provides a tutorial, and [17] offers a more detailed study.

Multigrid algorithms are commonly used to accelerate the convergence of traditional relaxation methods, such as Jacobi and Gauss-Seidel, by efficiently eliminating both local (small-scale) and global (large-scale) error components, using a hierarchy of grids. The multigrid method can be combined with the so-called pyramidal approach (which is inefficient as a stand-alone technique) to yield the Full Multigrid (FMG) algorithm, which is often optimal for discretized Partial Differential Equations (PDE).

Multigrid techniques have been used successfully for problems in image processing and vision, e.g., [18, 19, 20, 21, 22, 23, 24].

1.3 Application to the Geodesic Active Contour

We propose an implicit solution to the level-set formulation of the geodesic active contour by an adaptive multigrid method. This is a time-dependent problem, which means that we have an initial solution and a time evolution rule. Every application of the discretized evolution rule is called a *time-step*. The solution associated with each time-step provides an initial approximation for the next time-step. That is, at every time-step we solve the same problem with a new initial condition, which is closer to the target segmentation. Explicit time-discretizations of such problems suffer from a limitation on the size of the allowable time-step magnitude, Δt, known as the CFL (Courant, Friedrichs, Lewy) restriction [25], resulting in a large number of time-steps. We therefore use an implicit formulation, which is unconditionally stable and thereby free of the time-step magnitude limitation. Hence, we can use a large Δt and converge in just a few steps, but we need to solve a system of equations at every time-step.

Our problem involves several inherent difficulties, e.g., it is nonlinear, it contains rapidly-varying coefficients, and it is not symmetric. These properties guide our algorithmic preferences.

In Section 2 we present the numerical scheme, and in Section 3 we apply it to the geodesic active contour problem. Section 4 presents the adaptive multigrid approach. In Section 5 we discuss our treatment of rapidly-varying coefficients. In Section 6 we show results for several images, and Section 7 presents an application of the geodesic active contour for tracking a moving object in a movie. Section 8 contains conclusions. The discretization and a discussion of numerical stability appear in appendices.

2 The Numerical Scheme

This section presents the numerical scheme. It can be applied to other grid-based time-dependent problems, such as image diffusion. The details of the discretization are discussed in Appendix B.

2.1 Implicit Representation of Time-Dependent Problems

In a time-dependent problem, with a solution $\mathbf{\Phi} : \mathbf{R^2} \times [\mathbf{0}, \mathbf{T}] \rightarrow \mathbf{R}$, we typically have an initial condition Φ^0 at $t = t_0$, and a time evolution rule Φ_t, which drives $\mathbf{\Phi}$ towards some steady state. We shall denote by $\Phi : \mathbf{R^2} \rightarrow \mathbf{R}$, with or without superscripts, snapshots of $\mathbf{\Phi}$ at specific times. A classical scheme for time-dependent problems is the explicit (Forward Euler) scheme

$$\Phi^{n+1} = \Phi^n + \Delta t \Phi_t^n, \tag{1}$$

where Δt is the magnitude of the time-step and Φ^n is the solution at time $t = t_0 + n\Delta t$. The evolution rule Φ_t^n is generally a spatial PDE for Φ. Explicit numerical schemes significantly limit the magnitude of Δt, due to the CFL restriction [25], and usually require a large number of time-steps in order to converge. However, the solution at every time-step is computed easily.

An implicit (Backward Euler) scheme is given by

$$\Phi^{n+1} = \Phi^n + \Delta t \Phi_t^{n+1}. \tag{2}$$

For many problems the implicit scheme is unconditionally stable (see also Appendix A), and the solution for a given time can be obtained in just a few time-steps. However, the solution at each time-step requires solving the system

$$\Phi^{n+1} - \Delta t \Phi_t^{n+1} = \Phi^n. \tag{3}$$

A common approach for solving this PDE is by discretizing on a grid, obtaining a large algebraic system of equations. These can be solved iteratively, for example, by classical relaxation methods such as Gauss-Seidel or Jacobi. The solution is advanced by using *update-steps*, in which we change each unknown in turn to satisfy its equation. A cycle in which we correct every unknown once is called an iteration or relaxation sweep. A major problem of the traditional relaxation methods is the slow elimination rate of global error, that is, large-scale errors which cannot be detected locally. The multigrid approach overcomes this problem.

2.2 Multigrid

The multigrid approach [16, 17] enables us to efficiently eliminate both local and global errors, by employing a hierarchy of grids. This significantly reduces the computational cost of solving grid-based problems.

We solve each time-step as a separate problem. The system (3) can be written more generally as

$$A(\Phi) = F, \tag{4}$$

where $A(\Phi)$ is the left-side discretized operator, and F contains the right-side given values, that is, Φ from the previous time-step. We start with Φ from the previous time-step on the finest grid and perform the following recursive multigrid routine.

Multigrid Routine

- Relax on the current grid ν_1 times.
- Restrict the problem to the next coarser grid.
- Execute **Multigrid Routine** recursively on the coarser grid γ times.
- Correct the current grid values using the coarser grid solution.
- Relax on the current grid ν_2 times.

Common values for ν_1, ν_2 and γ are 1 or 2. When γ=1 the recursion is called a V cycle, and for γ=2 it is called a W cycle. The parameters are chosen for approximately optimal efficiency. The action of the relaxation is to reduce local errors, while the coarse-grid correction reduces the global ones. However, there is generally some coupling between the local and global errors. Therefore, applying more than just three or four relaxation sweeps is not cost-effective, because the coarse-grid correction will reintroduce some local errors anyway through interpolation and aliasing. For nearly all applications, including the present one, the optimal value for $\nu_1 + \nu_2$ is found to range from 2 and 4, and the differences are not great. The choice for γ depends on how well the coarser grid problem approximates the fine-grid one. With a very good approximation, $\gamma = 1$ is optimal, and indeed this is the most common value. But when the coarse-grid approximation is weaker, as in our case (due to strongly varying coefficients), $\gamma = 2$ may be necessary for good convergence. Larger values of γ are rarely cost-effective.

For nonlinear systems, as in the geodesic active contour, we use a multigrid version called the Full Approximation Scheme (FAS) [15]. We need to define an appropriate *Restriction Routine*, which transfers the problem to coarser grids, and a *Prolongation Routine*, which corrects the values of the finer grid using the solution of the coarser grid.

After the initial relaxation on the fine grid, we obtain an approximation to the solution, which we denote $\widehat{\Phi}$. The error E is defined as

$$E = \Phi - \widehat{\Phi}. \tag{5}$$

The Residual R is defined as

$$\begin{aligned} R &= F - A(\widehat{\Phi}) \\ &= A(\Phi) - A(\widehat{\Phi}) \\ &= A(\widehat{\Phi} + E) - A(\widehat{\Phi}). \end{aligned} \tag{6}$$

We thus have

$$A(\widehat{\Phi} + E) = R + A(\widehat{\Phi}), \tag{7}$$

which we use to obtain an approximation to $\widehat{\Phi} + E$. This process is carried out on a coarser grid, after E has been smoothed by relaxation.

Henceforth, $(\cdot)_{\downarrow}$ denotes (fine to coarse) restriction, typically full weighting[1], and $(\cdot)_{\uparrow}$ denotes (coarse to fine) prolongation, bilinear interpolation. The subscript l is used to define the grid level, where $l+1$ is the next-coarser grid-level, and $l = 0$ at the finest grid.

The Restriction and Prolongation Routines which follow from (7) are

The Restriction Routine

Initialize Φ_{l+1} values

$$\widehat{\Phi}_{l+1} \leftarrow (\widehat{\Phi}_l)_{\downarrow}. \tag{8}$$

Initialize F_{l+1} values

$$F_{l+1} \leftarrow A_{l+1}((\widehat{\Phi}_l)_{\downarrow}) + (R)_{\downarrow}. \tag{9}$$

The Prolongation Routine

After we solve the problem approximately on the coarser grid, we correct the fine-grid approximation by

$$\widehat{\Phi}_l \leftarrow \widehat{\Phi}_l + (E_{l+1})_{\uparrow}, \tag{10}$$

where

$$E_{l+1} = \Phi_{l+1} - (\widehat{\Phi}_l)_{\downarrow}. \tag{11}$$

We follow this by ν_2 fine-grid relaxation sweeps.

2.3 Schedule of Time-Dependent Problems

At each time-step, we solve a system of equations of the form (3), using one or more multigrid cycles. The total number of time-steps can be fixed (pre-determined) or it may depend on some stopping criterion. The present application did not require any elaborate stopping criteria, as an approximate steady state was always reached, in our examples, in five time steps or less. The is no reason to reduce the residuals to very small values, as these indicate some global error measure, which is not very relevant to the segmentation problem. Once an approximate steady state is reached and maintained, there is no reason to continue with the time-stepping, as any further improvement in the solution to the system yields no discernible improvement in the segmentation.

We only require an approximate solution at each time-step, because it is only used as an initial approximation for the next time-step. Therefore, one or a few multigrid cycles per time-step suffice.

In Figure 1 we show the schedule of a problem which is solved in three time-steps. In each time-step we apply two multigrid cycles. In each step of the cycle we can calculate the residual (6) and thereby estimate how far we are from the desired solution. In the lower part of Figure 1 we show

[1]Full weighting is the (scaled) transpose of bilinear interpolation.

the typical behavior of the residual norm. The multigrid cycles reduce the residuals, but there is some increase with every new time-step, because the right-hand side changes. Overall, the residuals are reduced in time as the solution tends to a steady state.

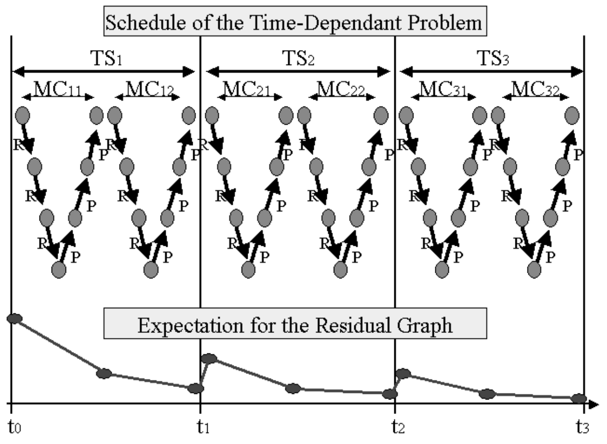

Schedule of the Time-Dependant Problem
TS1
TS2
TS3
MC11
MC12
MC21
MC22
MC31
MC32
R
P
Expectation for the Residual Graph
t0
t1
t2
t3

Figure 1: TS_i - time-step i, MC_{ij} - multigrid cycle j of time-step i, R - Restriction Routine, P - Prolongation Routine

3 The Geodesic Active Contour

The segmentation of image I, is given by the curve C which brings the geometrical functional,

$$\oint_C g(C(s))ds + \alpha \iint_\Omega g da, \tag{12}$$

to a local minimum, where s is the arc-length parameterization of the curve C, g indicates the presence of edges in the image, Ω is the area in the interior of the curve, and α is a parameter that controls the ratio between the two terms. Note that this functional is geometric, and therefore independent of the parameterizations. A simple implementation of g for the image I can be

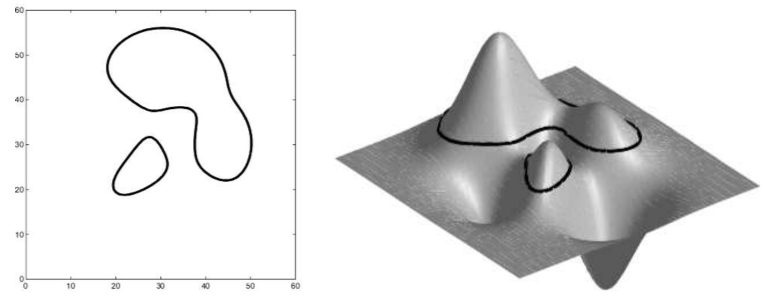
$$g(x,y) = \frac{1}{1+|\nabla I(x,y)|^2}. \tag{13}$$

Here, g has values close to 0 where edges exist, and higher values, close to 1, elsewhere. The function g is the main input of the problem, and a good g, which captures the features of edges and contains minimal noise, is necessary for an accurate segmentation.

The Euler-Lagrange equations of (12) yield the following steepest-descent evolution rule:

$$C_t = (\kappa g - \langle \nabla g, \vec{N} \rangle + \alpha g)\vec{N}, \tag{14}$$

where κ is the curvature of C, and $\langle \nabla g, \vec{N} \rangle$ is the inner product of the normal vector of C and the gradient vector of g. In order to efficiently implement the evolution so that we have a stable solution which handles curvature singularities and topological changes of the curve C, as in the case where there are several objects in the image, we use the Osher-Sethian level-set method [8], whereby the propagating interface is embedded in a higher dimensional function; see Figure 2 for an illustration.



(a) (b)

Figure 2: Level-set illustration: (a) the curve (b) the curve embedded as a level set of a function.

The evolution $C_t(s;t)$ of the curve C becomes, in its level-set formulation, the evolution Φ_t of the function $\Phi(x, y; t)$,

$$\Phi_t = g \operatorname{div}\left(\frac{\nabla \Phi}{|\nabla \Phi|}\right)|\nabla \Phi| + \langle \nabla g, \nabla \Phi \rangle + \alpha g|\nabla \Phi|. \tag{15}$$

From the function Φ, we can extract the curve C using the appropriate level-set, that is, the set of points (x, y) which satisfy $\Phi(x, y) = const$.

3.1 The Semi-Internal, Semi-External and Balloon Forces

We refer to the first part of (15),

$$\mathcal{I} = g \operatorname{div}\left(\frac{\nabla \Phi}{|\nabla \Phi|}\right)|\nabla \Phi|, \tag{16}$$

as a *semi-internal force* which causes the active contour to flow by its weighted curvature vector and minimize its length. This force causes the curve to be smoother and simpler. The second part of (15),

$$\mathcal{E} = \langle \nabla g, \nabla \Phi \rangle, \tag{17}$$

is a *semi-external force* which attracts the active contour to the locations of low g, which are the boundaries of the objects in the image. The third part,

$$\mathcal{B} = \alpha g |\nabla \Phi|, \tag{18}$$

is called the balloon force. It accelerates the motion of the active contour when it is far from object boundaries. One problem that arises when using the balloon force is that the contour may overshoot significant boundaries. Therefore, we turn off the balloon force near boundaries. Accordingly, we change (18) to

$$\mathcal{B} = \tilde{g} |\nabla \Phi|, \tag{19}$$

where $\tilde{g}$ uses a location-dependent indicator that detects boundaries of objects. A simple possible implementation of $\tilde{g}$ is a threshold of g,

$$\tilde{g} = \begin{cases} \alpha g & \text{if } g_{(x,y)} > T \\ 0 & \text{otherwise} \end{cases}, \tag{20}$$

.

The flow Φ_t, is given by the sum of the semi-internal force, the semi-external force and the weighted balloon force,

$$\Phi_t = (\mathcal{I} + \mathcal{E} + \mathcal{B}). \tag{21}$$

We can express $\mathcal{I}$, $\mathcal{E}$ and $\mathcal{B}$ explicitly as

$$\begin{aligned} \mathcal{I} &= g \left(\tfrac{\Phi_{xx}\Phi_y^2 - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_{yy}\Phi_x^2}{\Phi_x^2 + \Phi_y^2} \right), & (22) \\ \mathcal{E} &= \Phi_x g_x + \Phi_y g_y, & (23) \\ \mathcal{B} &= \tilde{g}\sqrt{\Phi_x^2 + \Phi_y^2}. & (24) \end{aligned}$$

The function Φ is defined on a uniform two-dimensional grid, and we use finite differences to approximate its derivatives.

Two additional considerations need to be addressed:

- The denominator of (22) vanishes where the gradient of Φ is equal to zero. In order to avoid division by zero we add a small positive constant, ϵ, to the denominator.
- In order to achieve a stable discretization of the semi-external force, we use upwind discretization for Φ_x and Φ_y (see e.g. [26]).

At this point the general mathematical model of the geodesic active contour flow, Φ_t, is well defined. Next, we consider how to evolve Φ efficiently towards the desired solution.

3.2 Implicit Representation of the Geodesic Active Contour

Application of the implicit scheme, (3), to the evolution, (21), produces a nonlinear system of equations,

$$\Phi^{n+1} - \Delta t \left(\mathcal{I} + \mathcal{E} + \mathcal{B}\right) = F, \tag{25}$$

where F is Φ^n. We use second-order central differences to discretize first and second derivatives, except for the first derivatives in the semi-external force, for which we use first-order upwind differences (with the "wind direction" determined by the signs of the derivatives of g). For each grid point there is one unknown and one equation. Each unknown represents the value of Φ in the new time-step and every equation includes nine unknowns (at the point itself and the eight neighboring points) satisfying the discrete form of (25).

3.3 The Update-Step

In order to perform the update-steps in the relaxation, we must isolate $\Phi_{i,j}$ in (25). It turns out that this can be done explicitly, because $\Phi_{i,j}$ appears linearly in the (i,j)th equation, even though the system is nonlinear. The relaxation is carried out by scanning the variables in some prescribed order, and updating each variable $\Phi_{i,j}$ in turn so as to satisfy its equation.

3.4 Pyramid of the Edge Indicator Function

Since we must discretize our problem on every grid-level, we need to restrict g to all the coarse grids. The function g can be calculated directly from the input image only for the finest grid; see image (a) in Figure 3. For the coarser grids, g is calculated from the finer-level g using restriction, $g_{l+1} \leftarrow (g_l)_\downarrow$; see images (b)-(d) in Figure 3.

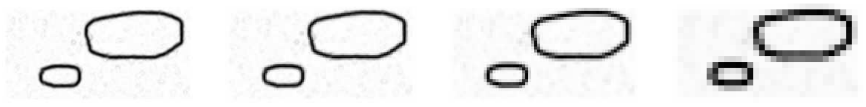


Figure 3: Example of a g pyramid, (a)–finest, (d)–coarsest.

4 Adaptive Multigrid

In many cases it makes sense to use a fine grid only in specific locations where high precision is required. There are several different treatments for such local refinement in the multigrid framework. Two notable approaches are the Multi-Level Adaptive Technique (MLAT) [15] and the Fast Adaptive Composite grid (FAC) [27]. Each allows the options of *static refinement* and *dynamic refinement*. In static refinement the locally-refined grids are defined prior to the solution of the problem, whereas in dynamic refinement we determine the refinement locations during the solution process according to some solution-dependent criteria. Here, we choose to use MLAT with static refinement. We choose MLAT because it is naturally compatible with FAS, which we need to employ due to the nonlinearity. And we use static refinement since we require high accuracy in the vicinity of object boundaries, which we know beforehand from the edge indicator function, g. See Figure 4 for an illustration.

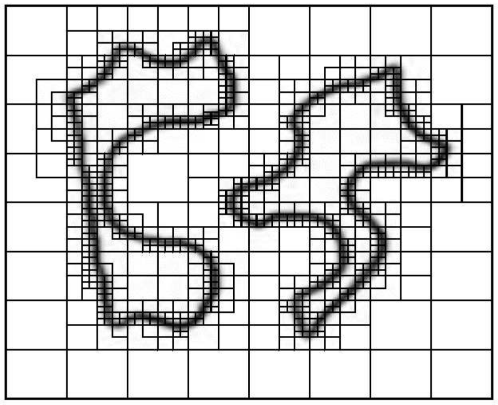


Figure 4: An illustration of local refinement

4.1 Constructing a Grid Hierarchy

In adaptive multigrid we use a hierarchy of grids where finer grids are restricted to local sub-domains. The topmost (finest) grid is local, and we gradually descend in the hierarchy of local grids until we reach the coarsest grids, which are global.

These are the steps we perform in order to construct the grids:

- Define the sub-domains.
- Expand each sub-domain using morphological procedures.
- Define ghost points which are needed in order to provide numerical support to the interface points of the sub-domains.

To initialize the finest local grid we choose a threshold, T, for g. Every point that has a value lower than T is included in the finest local grid. We expect to obtain a band which surrounds the object boundaries. Sometimes, however, there may be additional small patches on the finest local grid, due to noise in g. To eliminate these, we can use the morphological procedures [28, 29]: erosion and dilation. More dilations mean more computation later, but better stability; see Figure 5.

For the relaxation update-steps in each sub-domain, all neighboring values need to be defined. For this we use ghost-values (marked white in Figure 5), which are interpolated from the next-coarser grid.

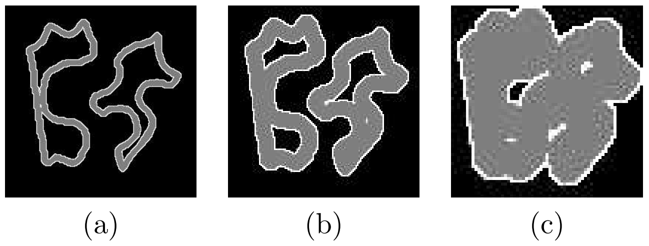


Figure 5: Adaptive multigrid sub-domains, (a)-(c) fine to coarse; gray - points that are included in the sub-domains; white - ghost points; black - points that are not involved in any computation.

4.2 Adaptive Multigrid Routines

We need to modify slightly the basic multigrid routines due to the use of local grids.

- Initialization: set the initial solution, Φ, and the right-hand side, F, at all grid points belonging to a locally finest grid.
- Relaxation: calculate ghost values, and perform update-steps only in sub-domains.
- Restriction and Prolongation: perform in sub-domains only.

To summarize, relaxation is performed only in the sub-domains, values that are not available on a given sub-domain are interpolated from the next-coarser grid, and values at each locally-finest grid-point are initialized by values from the previous time-step. Restriction of residuals is carried out by local averaging with weights that sum up to one (see Figure 6). Away from sub-domain boundaries, we use the standard full-weighting operator (which is a scaled transpose of bilinear interpolation). At sub-domain boundaries we need to compensate for missing data which is undefined at the finer scale, in order to retain the weight-sum of one, which is required for consistency of the coarse-grid problem. Motivated by algebraic multigrid (AMG) methods [16, Chapter 8], we compensate using diagonal values—here meaning values at the fine-grid point that corresponds to the coarse-grid point to which the residuals are being restricted. The rationale is that, after the residuals have been smoothed by the relaxation, they do not vary much between neighboring grid points, so not much is lost by representing the missing neighboring residuals by the residual at this point.

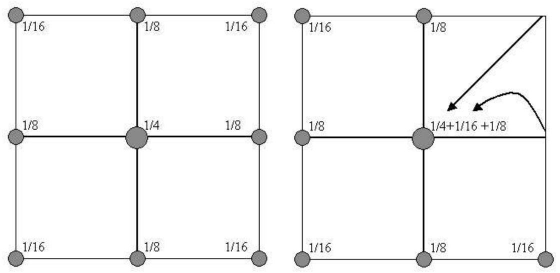


Figure 6: Restriction at the sub-domain interface; left – regular full weighting; right – restriction where not all the nine points are in the sub-domain.

4.3 Time Complexity Analysis

Here, we analyze the time complexity of the multigrid algorithm. As usual (see, e.g., [15, §6]), we neglect all but the computation required for relaxation and residual calculation, as these dominate the computational cost. We define a *work-unit* as a single residual calculation on the finest grid. A work unit requires about 60 operations per grid point, and it is approximately equivalent to the cost of a single explicit time-step. A single update step costs about 50% more than this, due to the splitting employed (see discretization and splitting in Appendix B). Evidently, the number of operations performed in the entire computation is proportional to the number of time-steps, denoted N_{TS}, and to the number of multigrid cycles per time-step, denoted N_{MC}. For 2D multigrid, the complexity of a single cycle in work-units, whether V cycle or W cycle, is proportional to the number of grid points. Specifically, if the number of relaxations performed before and after coarse-grid correction is ν_1 and ν_2, respectively, then the total number of work units is bounded by $C(\nu_1 + \nu_2)$, where $C = 4/3$ for V cycles and $C = 2$ for W cycles. (We assume here that the complete grid is used at every level; the decrease in work due to the adaptive local refinement is mentioned later). The overall work thus depends to some extent on the particular problem and initial conditions, as this effects the total number of cycles required. As we use W(2,1) cycles in our implementation (see below), the total number of work-units, N, adds up to approximately

$$N = 10 N_{TS} N_{MC} \,. \tag{26}$$

N_{MC} is a small constant number, typcially two or three, while N_{TS} is problem dependent - in simple cases 3 or 4 time-steps are sufficient. In problems with several non-convex objects more time-steps are needed, but generally less than a dozen are sufficient.

To summarize, the number of operations required for obtaining the final solution using the multigrid algorithm is generally less (and often much less) than 10000 times the number of mesh-points on the finest grid, when each mesh covers the full domain. Further iterations would do not alter the solution perceptibly. In contrast to this, an explicit approach typically requires several thousands of time-steps. The number of operations decreases very significantly when the adaptive algorithm is employed, since the finer grids then cover only a small part of the domain.

5 Locally Damped Prolongation

In the multigrid FAS algorithm, the problem is discretized on all the grids. When the image contains objects with fine details, the edge indicator function, g, cannot resolve these details on the coarser grids. In severe cases, the resulting poor coarse-grid approximation then impairs the convergence. Such a situation is illustrated in Figure 7, where fine details in (g_1), (g_2) are not present in (g_3), (g_4).

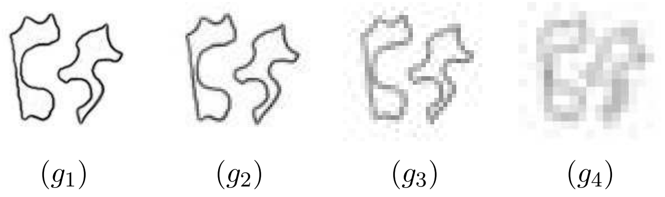


Figure 7: Puzzle pieces g pyramids.

One approach for handling this problem is by employing the Algebraic Multigrid (AMG) method [30, 31]. However, AMG is designed for linear problems, while our system is nonlinear, and moreover, the computational overhead would be very substantial for this problem. Instead, we therefore choose to use the far simpler "geometric" multigrid approach, and overcome the problem by adapting the prolongation operator. We define the *difference matrix* d for non-coarsest level l as

$$d_l = |g_l - (g_{l+1})_{\uparrow}|. \tag{27}$$

We expect poor coarse-grid approximations in places where d_l is large, which implies that g_{l+1} fails to represent g_l accurately.

We use d_l to construct a matrix r_l, which is used to locally damp the prolongation.

$$r_l = 1 - (\alpha(d_l)_\sigma + \beta), \tag{28}$$

where α and β restrict r_l to the range $[0, 1]$. The highest value in $(d_l)_\sigma$ receives the value 0 and the lowest value receives the value 1. $(d_l)_\sigma$ is equal to d_l, smoothed by a Gaussian with variance σ. Figure 8 shows r_l for three grids. With the damped prolongation, the false coarse-grid corrections are avoided.

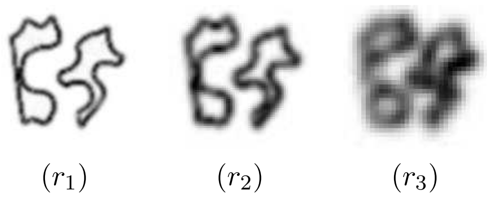


Figure 8: Puzzle pieces prolongation damping matrix, r_l.

In Figure 9 we see the effects of prolongation damping. Without the damping, there is loss of detail, but the damping results in an accurate segmentation.

6 Results

In this section and the next we report results obtained with the multigrid algorithm for several test cases. The total number of multigrid cycles required for reaching the approximate steady state never exceeded a dozen. On a standard 1GHz PC the CPU time required was about one second per time-step, hence typically two to four seconds per application. In contrast, traditional explicit schemes require several minutes to converge to the same solutions.

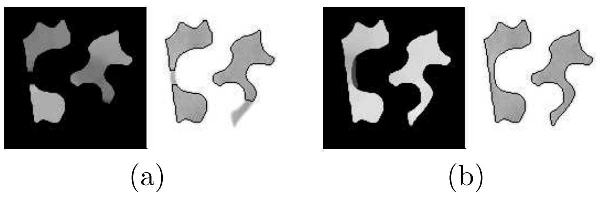


Figure 9: (a) - segmentation without prolongation damping, (b) - segmentation with damped prolongation.

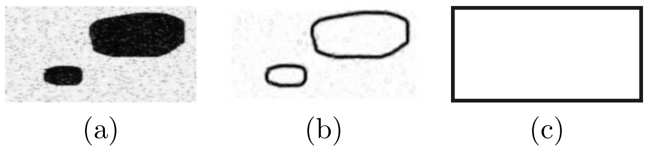


Figure 10: Simple case: (a) input image, (b) g function, (c) initial solution.

6.1 A Simple Case

First we demonstrate the efficiency of the multigrid approach on a simple example, with two clear convex objects. We compare the first four time-steps, using a different number of grid-levels in each case.

In Figure 10, we show the input image, (a), its g function (b), and the initial contour, (c). In Figure 11 we see that the multigrid algorithm has a significant effect on the convergence rate. The only difference between the four cases is the number of grid-levels employed. In all cases, the number of time-steps is $n = 4$, the number of multigrid cycles is $c = 3$ (we use W cycles), the number of pre-relaxations is $\nu_1 = 2$, and the number of post-relaxations is $\nu_2 = 1$.

In the first test we use a single grid-level. The number of relaxations in every time-step is $c(\nu_1 + \nu_2)$, which is identical to the number of relaxations on the finest grid in the multi-level cases (to yield a "fair" comparison). The results of the first four time-steps, which are presented in Figure 11 (a1)-(a4), show that the differences between successive iterations are small and Φ is very far from its final solution, even after the fourth time-step.

In the second test we add a second grid-level. The results of the first four time-steps, presented in Figure 11 (b1)-(b4), are better than those of the first test. The propagating interface of Φ starts to capture the objects' boundaries.

In the third test we use three grid-levels. The results for the first four time-steps, presented in Figure 11 (c1)-(c4), are significantly better than those of the second test. The propagating interface of Φ now captures the objects' boundaries in the fourth time-step.

In the fourth test we use four grid-levels. The results for the first four time-steps are presented in Figure 11 (d1)-(d4). The propagating interface of Φ captures the objects' boundaries in the third time-step. Note that if we add more time-steps after the convergence, the solution does not change.

6.2 Real-Life Examples

The geodesic active contour finds a local minimum for the propagating interface, in which the problem's main input is the edge indicator g function. We focus on constructing an efficient method for computing the contour,

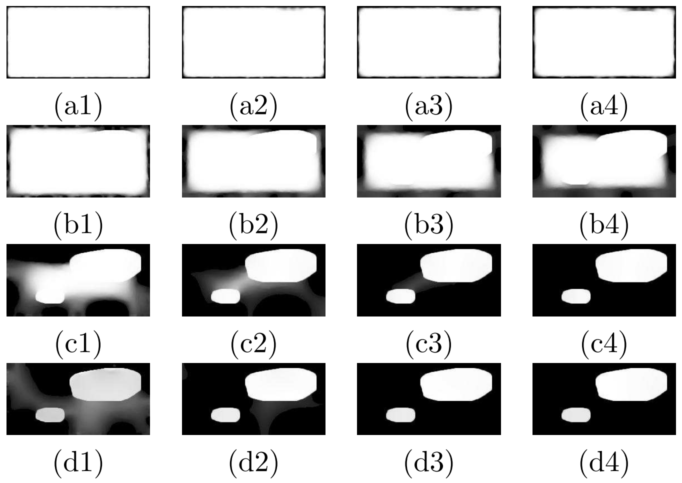


Figure 11: The first four time-steps, with different numbers of grid-levels. (a1)-(a4) one level, (b1)-(b4) two levels, (c1)-(c4) three levels, (d1)-(d4) four levels.

subject to a given g function. However, we also need to obtain an appropriate g function. A noisy g, or one that fails to represent the goal well, may cause the active contour to be attracted to false edges in some cases, or may skip over weak edges in others. In order to obtain a good g function, we first enhance the input image with an anisotropic diffusion (Beltrami flow [32]), which reduces noise while preserving edges. Alternative methods exist. One such method employs an adaptive g, in which edges are amplified in low variation areas, and are weakened in high variation areas. Another method gets user input, which provides an initial estimation that bounds the final solution.

We present three test cases: Two images of galaxies, and one medical image. The sizes of the images are 257^2 pixels. For the galaxies, we compare the computation time of the non-adaptive scheme and the adaptive scheme. The computational improvement of the adaptive approach is proportional to the area of the local grids, compared to the area of these grids before refinement. In these cases we achieved approximately 40% savings in time by using the adaptive approach. In cases where the local grids' area is smaller, we find greater improvements. We ran the cases on a standard 1GHz PC, taking about one second per time-step.

In the third case, we show segmentation of a medical image, IVUS (Intra Vascular Ultrasound). The objective is to capture the vein's inner and outer walls. This time, we apply the active contour so that it expands outwards, by assigning a negative value to the weighted balloon force. The initial condition is the ultrasound sensor at the center of the vein. We perform two time-steps to reach the vein's inner wall. In order to reach the outer wall's

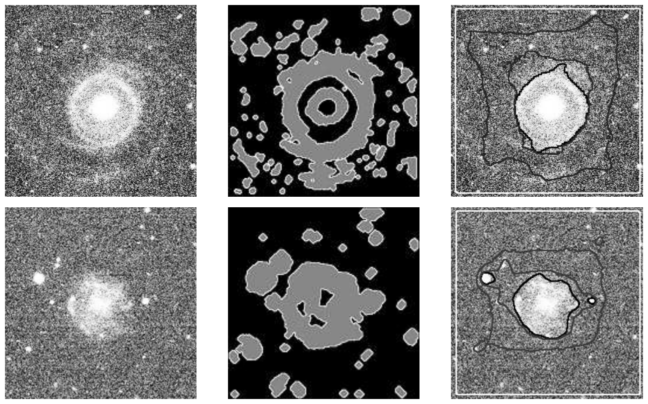

(a) (b) (c)

Figure 12: Results for Galaxy u6614 (top) and Galaxy f5611 (bottom)
(a) input image, (b) adaptive grid: the sub-domain of the finest grid, (c) solution: white contour - initial solution, gray contours - first and second time-steps, black contour - final solution.

solution we expand the inner wall's solution (by performing a morphological dilation procedure). In order to avoid converging to the inner wall again, we give the edge indicator function a value of 1 (no edge) in the area of the expanded inner solution. We then perform two additional time-steps and reach the outer wall as a second solution.

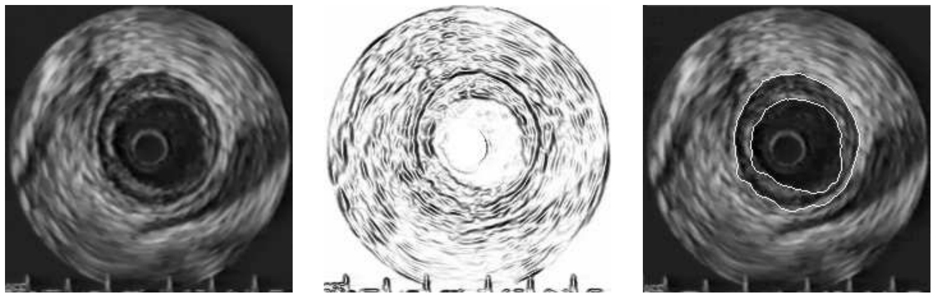

(a) (b) (c)

Figure 13: Results for IVUS
(a) input image of the vein, the sensor located in the center (b) the corresponding edge indicator function (c) output image: the white contours capture the inner and the outer walls.

7 Object Tracking in a Movie

In this section we present an implementation of the geodesic active contour in a sequence of images, in order to track a moving object. We first produce a simple Motion Detection Image (MDI) for each frame in the sequence of images. We then apply the geodesic active contour to the MDI in order to track the moving object(s).

7.1 Generating the Motion Detection Images

We assume a fixed camera and no significant global changes from frame to frame except for the moving object(s). For each frame in the sequence of images we produce its corresponding MDI, which presents the regions of change. We begin by computing the difference images, of the previous frame, $diff_{i,i-1}$, and of the frame, $diff_{i,i+1}$ (see Figure 14). The input images and $diff_{i,i-1}$ and $diff_{i,i+1}$ are then enhanced with the Beltrami flow, which selectively smoothes the image while preserving edges. We use the two difference images to generate the MDI, which is basically a binary image indicating for each pixel whether it belongs to an area of change. A simple condition on the difference images is used, where the minimum value of the corresponding pixels is higher than a certain threshold, and the product of these pixels is higher than another threshold. If so, we put 1 in the corresponding pixel of the MDI; otherwise we set it to 0. Finally, we smooth the binary MDI image with a Gaussian, which reduces the isolated changes and widens the edge support from one pixel in a binary image to several pixels.

7.2 Applying the MGAC to the Motion Detection Images

The multigrid geodesic active contour scheme is an effective tool for segmentation when processing on MDI. Most of the MDI is empty (black); see Figure 14. Thus, the adaptive scheme, which saves computations at empty regions, is highly efficient. The propagating active contour quickly closes in on the objects and captures them. Segmenting a sequence of images can be even more efficient if there is a known limit on the object's movement from frame to frame, where the initial condition for a frame can be a dilation of the solution for the previous frame.

Figure 14 presents the process of tracking a moving object. The images in the upper row are three successive images in a movie, the images in the middle row are the difference images and the MDI. In the bottom row we present the final result which is the level set of Φ combined with the frame i.

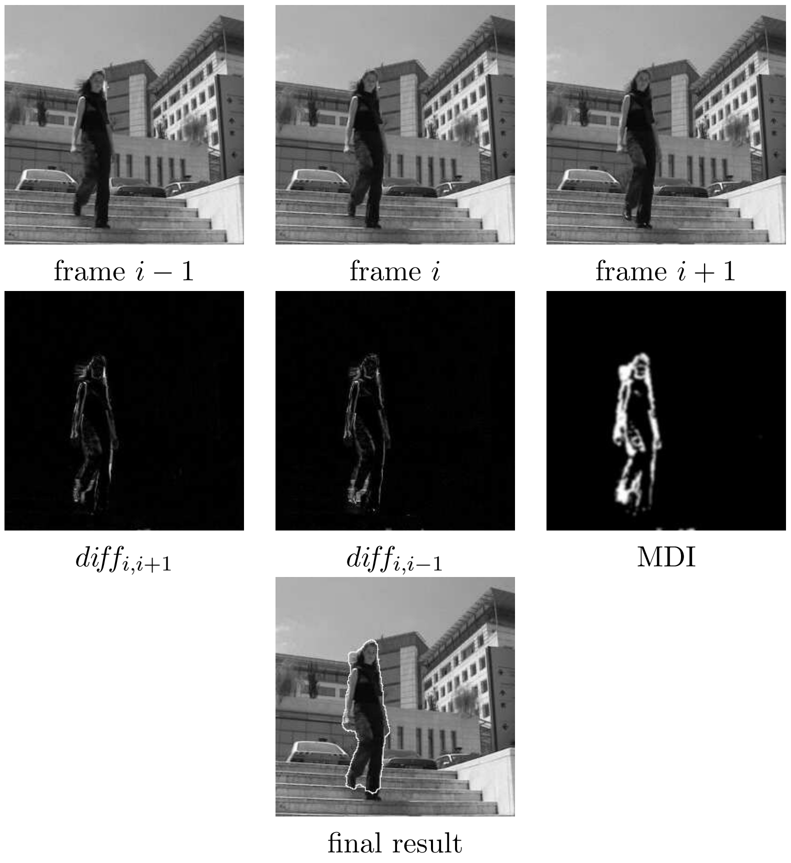


Figure 14: The process of capturing a moving object in a movie.

8 Conclusions

An efficient numerical scheme for solving time-dependent problems in image analysis has been presented. It consists of an implicit formulation of a time-dependent problem and a multigrid solver. We have also used this scheme effectively in other time-dependent problems, in particular, the single channel Beltrami flow, [32], which is an anisotropic diffusion filter that we use to enhance the input images.

The solution depends on the quality of the edge indicator function, and on its ability to represent the edges of the objects in the image. Noisy or false edges can attract the active contour to a spurious solution; alternatively, the active contour may skip over significant edges. In this work we focus on solving the active contour subject to a given simple edge indicator. Further work should concentrate on how to obtain a good edge indicator function.

The edge indicator function strongly affects the coefficient matrix of the problem. The matrix consequently often contains rapidly changing values.

This makes it difficult to approximate on coarse grids and may lead to inaccurate coarse-grid solutions, which impair convergence. We overcome this problem by damping the prolongation in regions where the edge indicator function is not approximated well on the coarse grid.

9 Appendix A

Here we show the unconditional stability of the implicit approach, for time-dependent problems which are derived from the minimization of a functional. Consider a generic minimization problem of finding a vector Φ which minimizes some convex functional, $J(\Phi)$. Given an approximate solution after time-step n—Φ^n—the $n+1$st implicit time-step can itself be written as the minimization problem:

$$\Phi^{n+1} = \text{argmin}_\phi \left[J(\phi) + \frac{1}{2\Delta t}(\phi - \Phi^n)^2 \right] . \tag{29}$$

[To see the equivalence, note that the solution satisfies $\Delta t J'(\Phi^{n+1}) + \Phi^{n+1} = \Phi^n$; compare to (2), with $\Phi_t^{n+1} \equiv -J'(\Phi^{n+1})$]. Since Φ^{n+1} is the minimizer, if we substitute Φ^{n+1} for ϕ in the functional on the right side of (29), the result cannot be larger than what we would get by substituting anything else, in particular, Φ^n, hence,

$$J(\Phi^{n+1}) + \frac{1}{2\Delta t}(\Phi^{n+1} - \Phi^n)^2 \leq J(\Phi^n) . \tag{30}$$

Evidently, therefore, $J(\Phi^{n+1})$ is strictly smaller than $J(\Phi^n)$, so the functional is decreased at each time step unless a steady state has been reached. This proves that the implicit procedure is unconditionally stable.

Of course this holds strictly only if the discrete problem is itself a minimization of a functional, for example, if the functional itself is discretized, and a minimum is then sought. In the present application we chose instead to discretize the Euler-Lagrange equations in order to reduce the cost of the discrete problem. Thus, strict monotonicity of convergence is not assured in our case, but the process nevertheless remained stable.

10 Appendix B

Here we describe the discretization and relaxation of (25). We perform Gauss-Seidel relaxation sweeps, whereby the variables are scanned and each variable, $\Phi_{i,j}$ is updated according to its neighbors. Although the equations are nonlinear, it turns out that, given our discretization (see below), the relation between $\Phi_{i,j}$ and its neighbors allows us to extract it explicitly without requiring the solution of a local nonlinear problem. That is, at each grid-point, (i, j), each term in the discretized (25) can be split in the form

$A_{free} + \Phi_{i,j} A_{bound}$, where A_{free} and A_{bound} are independent of $\Phi_{i,j}$. We next describe the discretizations of each of the terms in this form.

10.1 Splitting the Internal Force

Here we use central finite differences approximations, so $\Phi_{i,j}$ appears only in $\Phi_{xx}(i,j)$ and in $\Phi_{yy}(i,j)$. Therefore, splitting (22) yields

$$\begin{aligned}
\mathcal{I}_{free} &= g_{i,j}\left(\tfrac{(\Phi_{i+1,j}+\Phi_{i-1,j})\Phi_y^2-2h^2\Phi_x\Phi_y\Phi_{xy}+(\Phi_{i,j+1}+\Phi_{i,j-1})\Phi_x^2}{h^2(\Phi_x^2+\Phi_y^2+\epsilon)}\right), && (31)\\
\mathcal{I}_{bound} &= g_{i,j}\left(\tfrac{-2(\Phi_x^2+\Phi_y^2)}{h^2(\Phi_x^2+\Phi_y^2+\epsilon)}\right), && (32)\\
\mathcal{I} &= \mathcal{I}_{free} + \Phi_{i,j}\mathcal{I}_{bound}. && (33)
\end{aligned}$$

Here, the derivatives are approximated by the usual second-order accurate central differences.

10.2 Splitting the External Force

In order to achieve a stable solution for this force, we need to use upwind differences for the first derivatives in (23). This is done by using Φ_x^+ if $g_x > 0$, and Φ_x^- otherwise. The same form applies for Φ_y. Here,

$$\begin{aligned}
\Phi_x^+(i,j) &\approx \frac{\Phi_{i+1,j}-\Phi_{i,j}}{h}\\
\Phi_x^-(i,j) &\approx \frac{\Phi_{i,j}-\Phi_{i-1,j}}{h}\\
\Phi_y^+(i,j) &\approx \frac{\Phi_{i,j+1}-\Phi_{i,j}}{h}\\
\Phi_y^-(i,j) &\approx \frac{\Phi_{i,j}-\Phi_{i,j-1}}{h}.
\end{aligned}$$

We define the signs of $g_x(i,j)$ and $g_y(i,j)$ as follows:

$$\begin{aligned}
sg_x(i,j) &= \begin{cases} 1 & \text{if } g_x(i,j) > 0 \\ -1 & \text{otherwise} \end{cases}\\
sg_y(i,j) &= \begin{cases} 1 & \text{if } g_y(i,j) > 0 \\ -1 & \text{otherwise.} \end{cases}
\end{aligned}$$

The discretization of (23) now reads

$$\mathcal{E} = g_x(i,j)\tfrac{sg_x(i,j)\cdot(-\Phi_{i,j}+\Phi_{i+sg_x(i,j),j})}{h} + g_y(i,j)\tfrac{sg_y(i,j)\cdot(-\Phi_{i,j}+\Phi_{i,j+sg_x(i,j)})}{h}, \qquad (34)$$

and the splitting of (34) is given by

$$\begin{aligned}
\mathcal{E}_{free} &= g_x(i,j)\tfrac{sg_x(i,j)\cdot\Phi_{i+sg_x(i,j),j}}{h} + g_y(i,j)\tfrac{sg_y(i,j)\cdot\Phi_{i,j+sg_y(i,j)}}{h}, && (35)\\
\mathcal{E}_{bound} &= -\left(\tfrac{sg_x(i,j)\cdot g_x(i,j)+sg_y(i,j)\cdot g_y(i,j)}{h}\right), && (36)\\
\mathcal{E} &= \mathcal{E}_{free} + \Phi_{i,j}\mathcal{E}_{bound}. && (37)
\end{aligned}$$

10.3 Splitting the Balloon Force

The central-differenced balloon force is free of $\Phi_{i,j}$, so no splitting is needed and it is all “free”.

10.4 The Update-Step

As already mentioned, in the update-step we change the value of each variable $\Phi_{i,j}$ in turn such that it satisfies its local equation. The discretized (25) now reads

$$\Phi_{i,j}(1 - dt(\mathcal{I}_{bound} + \mathcal{E}_{bound})) - dt(\mathcal{I}_{free} + \mathcal{E}_{free} + \mathcal{B}) = F_{i,j}. \tag{38}$$

Isolating $\Phi_{i,j}$ we have

$$\Phi_{i,j} = \frac{F_{i,j} + dt(\mathcal{I}_{free} + \mathcal{E}_{free} + \mathcal{B})}{1 - dt(\mathcal{I}_{bound} + \mathcal{E}_{bound})}. \tag{39}$$

Acknowledgement

This research was partly supported by Technion V.P.R. Fund - D. Barsky Fund for Optics Research Fund

References

[1] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[2] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour model. *IJCV*, 1:321–331, 1988.

[3] L. D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, March 1991.

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

[5] R. Malladi, J. Sethian, and B. Vemuri. A topology independent shape modeling scheme. In *Proc. SPIE Conf. on Geometric Methods in Computer Vision II San Diego*, July 1993.

[6] 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):158–75, February 1995.

[7] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Proc. Fifth International Conference on Computer Vision*, pages 694–699. IEEE Comput. Soc. Press, 1995.

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

[9] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *Scale-space theories in computer vision, Lecture Notes in Computer Science*, 1682:34–45, 1999. M. Nielsen, P. Johansen, O. F. Olsen, J. Weickert (Eds.).

[10] J. Weickert, B. Romeny, and M. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(3):398–410, 1998.

[11] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys*, 118:269–277, 1995.

[12] R. P. Fedorenko. On the speed of convergence of an iterative process. *USSR Comput. Math. and Math. Phys.*, 4:559–564, 1964.

[13] N. S. Bakhvalov. On the convergence of a relaxation method under natural constraints on an elliptic operator. *USSR Comput. Math. and Math. Phys.*, 6:861–883, 1966.

[14] A. Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. *Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics I. Springer-Verlag, Berlin, West Germany*, pages 82–89, 1973.

[15] A. Brandt. Multi-level adaptive solutions to boundary-value problem. *Math. Comp*, 37:333–390, April 1977.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick. *Multigrid Tutorial, Second Edition.* Philadelphia, SIAM, 2000.

[17] U. Trottenberg, C. Oosterlee, and A. Schueller. *Multigrid.* Academic Press, 2000.

[18] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on PAMI*, 8(2):129–139, 1986.

[19] S. T. Acton. Multigrid anisotropic diffusion. *IEEE Trans. on Image Procc*, 7(3):280–291, March 1998.

[20] R. Kimmel and I. Yavneh. An algebraic multigrid approach to image analysis. *SIAM J. Sci. Comput. (in press)*, 2002.

[21] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. *IEEE CVPR*, 1:70–77, June 2000.

[22] M. Unser. Multigrid adaptive image processing. In *Proc. IEEE International Conference on Image Processing (ICIP)*, 1995.

[23] R. H. Chan, T. F. Chan, and W. L. Wan. Multigrid for differential-convolution problems arising from image processing. CAM report 97-20, UCLA, 1997.

[24] S. Henn and K. Witsch. A multigrid approach for minimizing a non-linear functional for digital image matching. *Computing*, 64:339–348, 2000.

[25] R. Courant, K. O. Friedrichs, and H. Lewy. Uber die partiellen Differenzengliechungen der mathematisches Physik. *Math. Ann.*, 100:32–74, 1928.

[26] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*. Cambridge Univ. Press, 1996.

[27] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations*, volume 6. Frontiers in Applied Mathematics, SIAM, Philadelphia, 1989.

[28] E. R. Dougherty. *An Introduction to Morphological Image Processing*. SPIE Press, Bellingham, Washington, 1984.

[29] G. Sapiro, R. Kimmel, D. Shaked, B. Kimia, and A. M. Bruckstein. Implementing continuous-scale morphology via curve evolution. *Pattern Recognition*, 29(9):1363–1372, 1993.

[30] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and its Applications*, pages 257–284, 1984.

[31] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19:23–56, 1986.

[32] R. Kimmel N. Sochen and R. Malladi. A geometrical framework for low level vision. *IEEE Trans. on Image Processing*, 7:310–318, 1998.