KYBERNETIKA — VOLUME *43* (2007), NUMBER 4, PAGES 509 – 522

FINITE–VOLUME LEVEL SET METHOD AND ITS ADAPTIVE VERSION IN COMPLETING SUBJECTIVE CONTOURS

Zuzana Krivá

In this paper we deal with a problem of segmentation (including missing boundary completion) and subjective contour creation. For the corresponding models we apply the semi-implicit finite volume numerical schemes leading to methods which are robust, efficient and stable without any restriction to a time step. The finite volume discretization enables to use the spatial adaptivity and thus improve significantly the computational time. The computational results related to image segmentation with partly missing boundaries and subjective contour extraction are presented.

Keywords: image processing, nonlinear partial differential equations, numerical solution, finite volume method, adaptivity, grid coarsening

AMS Subject Classification: 35K55, 65M12

1. FORMULATION AND ASSUMPTIONS OF THE STUDIED PROBLEM

We are going to solve the problems of segmentation (including missing boundary completion) and subjective contours creation. The aim of *segmentation* is to find boundaries of a distinguished object of an image. In generic situation these boundaries correspond to edges. However, in the presence of a noise or in images with occlusions these edges can be very irregular or even interrupted. Then the analysis of the scene and segmentation of objects becomes a difficult task.

Subjective contours, created by human visual system in specific images, are not a property of the image alone, but they depend both on the position of the point of view and on the geometric properties of the image. The perception of spatial patterns is dependent on the location of the gaze.

However, the basic technique for finding missing boundary and subjective contours remains the same. In brief it can be described as follows [12, 13]: *input into the algorithm is an image I_0 to be segmented. Initially, a segmentation surface (or point-of-view surface), given by an observer (user) chosen fixation point inside the image is constructed and taken as u^0. Then this initial state of the segmentation function is evolved by equation (1), when the "point-of-view" surface is attracted by the existing boundaries and steepens around the edges. The surface evolves towards* *the piecewise constant solution by continuation and closing of the boundary fragments (if they are missing) and the filling in the homogeneous regions – the so called subjective surface arises.*

Both the subjective surface and the segmentation can be defined as a graph that varies rapidly across the boundary between different objects and stays flat within it. Finding the boundary of objects then corresponds to finding an isoline or isosurface for a proper value.

The model solving these problems is expressed by the following partial differential equation

$$\frac{\partial_t u}{\sqrt{\varepsilon^2+|\nabla G_\sigma * u|^2}} - \nabla.(g(|\nabla I^\rho|)\frac{\nabla u}{\sqrt{\varepsilon^2+|\nabla G_\sigma * u|^2}}) = 0 \quad \text{in } Q_T \equiv I \times \Omega, \tag{1}$$

$$u(0,x) = u^0(x) \ \text{ in } \Omega, \tag{2}$$

$$u(t,x) = u^0(x) \quad \text{on } I \times \partial\Omega, \tag{3}$$

where u is a computed *segmentation function* or a *subjective surface*, $\Omega \subset \mathbb{R}^d$ is a rectangular computational domain (a subdomain of the image domain), $\varepsilon > 0$ is a parameter, $I = [0,T]$ is a time interval and I^ρ is a function given by

$$I^\rho = \begin{array}{l} \text{(i) } I^0 \text{ (no regularization)}, \ \rho = 0, \\ \text{(ii) } G_\rho * I^0 \text{ (regularization by a smoothing kernel)}, \ \rho \neq 0. \end{array} \tag{4}$$

The convolution of $H(x)$ and $h(x)$ is defined by $(H * h)(x) = \int_{R^d} H(s)h(x-s)\,\mathrm{d}s$. Further assumptions on the data of the model are summarized in

$$g(s) \text{ is a decreasing smooth function, } g(0) = 1, 0 < g(s) \to 0 \text{ for } s \to \infty, \tag{5}$$

$$G_\sigma \in C^\infty(\mathbb{R}^d) \text{ is a smoothing kernel with } \int_{\mathbb{R}^d} G_\sigma(x)\,\mathrm{d}x = 1 \tag{6}$$
$$\text{and } G_\sigma(x) \to \delta_x \text{ for } \sigma \to 0, \ \delta_x\text{–Dirac function at point } x,$$

$$u^0 \in L_\infty(\Omega), \tag{7}$$

$$I^0 \in L^2(\Omega). \tag{8}$$

If we replace $|\nabla G_\sigma * u|$ by $|\nabla u|$ in (1), then we can interpret the equation in two ways: like a modification of a well-known level set equation [11, 15, 16]

$$\frac{\partial_t u}{|\nabla u|} = \nabla.\left(\frac{\nabla u}{|\nabla u|}\right) \tag{9}$$

regularized in a sense of Evans and Spruck

$$|\nabla u| \approx |\nabla u|_\varepsilon = \sqrt{\varepsilon^2 + |\nabla u|^2}, \tag{10}$$

or, as a mean curvature flow of graphs with respect to the Riemann metric given by the image features. The idea to use Riemannian mean curvature flow of graphs to compute the subjective contours originates in [12, 13, 14]. Then, ε is not only the regularization factor, but also a modelling parameter which can help in suitable denoising and completing of missing boundaries. For small ε, the subjective surface closes gaps in image object boundaries and is stabilized, i. e. almost unchanging by further evolution, so it is easy to stop the segmentation process. For $\varepsilon = 1$ we come to the mean curvature problem for graphs, where the normal speed mean normal to the graph (i. e. segmentation function) itself and not normal to its isolines or isosurfaces ([7]).Our usage of $|\nabla G_\sigma * u|$ instead of $|\nabla u|$ in (1) is motivated by the effort to properly compute gradients in the finite volume method (dealing with piecewise constant approximations) and will be discussed later in more details.

For the numerical solution of (1) we have chosen the semi-implicit finite volume method. The so-called co-volume method based on semi-implicit time discretization for level-set-like problems was given in [2] and it has been applied to the segmentation problem in [8, 9] where also efficiency of the semi-implicit approach has been strongly emphasized. The co-volume technique as well as the finite volume technique is based on integral (weak, variational) formulation of (1). In such way, the discretization scheme naturally respects a variational structure of the problem. The semi-implicit discretization in time yields L_∞-stability property (i. e. no spurious oscillations appear in a solution) for any length of a discrete time step. This is a main advantage in comparison with explicit time stepping, where the stability is often achieved only under severe time step restriction. In every time update we solve linear system of equations which can be done efficiently using, e. g., suitable preconditioned iterative linear solvers. The advantage of the finite volume method in comparison with the co-volume method is, that in many cases the computational time can be even fastened by using spatial adaptivity [5, 6, 4], that appears to be easier to implement using finite volume grids.

2. DERIVING THE FV SCHEMES

First, to obtain the *integral formulation* of the diffusion equation (1), let us integrate it over a finite region $p \subset \Omega$. Then, using Green's theorem, for any p we get the integral identity

$$\int_p \frac{\partial_t u}{\sqrt{\varepsilon^2 + |\nabla G_\sigma * u|^2}}\, \mathrm{d}x - \int_{\partial p} g(|\nabla I^\rho|) \frac{\nabla u}{\sqrt{\varepsilon^2 + |\nabla G_\sigma * u|^2}} . \vec{\nu}\, \mathrm{d}s = 0. \qquad (11)$$

where $\vec{\nu}$ is the outward unit normal vector to ∂p.

To discretize the integral form (11) in time we choose N as the number of time steps and obtain the length of a uniform discrete time step $k = \frac{T}{N}$. Then at any discrete time $t_n = nk, n = 1, \ldots, N$ we replace the time derivative by the backward difference, i. e. $\partial_t u$ by $(u^n - u^{n-1})/(k)$, where u^n, u^{n-1} are solutions of (11) at times $t_n = nk$, $t_{n-1} = (n-1)k$, respectively. Treating the spatial nonlinear terms of the equation using solution from the previous time step and using approximation of the linear terms at the current time level leads to the so-called *semi-implicit method.*

For discretization in space we use the so-called *finite volume method (FVM).* We have two inputs into the model, namely the point-of-view surface (initial condition) u^0 and the regularized image I^ρ. Our spatial discretization is given by the pixel/voxel structure of the input image and thus it is the same for both u^0 and I^ρ. It means, the continuous image domain is subdivided into the rectangular regions p and at every time t_n we look for a solution u^n constant over each such *control volume* p. Instead of (1) we use its integral formulation (11) in these control volumes p. Now, let us introduce further notation to be able to give the precise formulation of our schemes. Let τ_h be a mesh of Ω with cells (control volumes) p of measure $m(p)$, where h is the maximal diameter of a cell in the mesh. For every cell p we consider a set of neighbors $N(p)$ consisting of all cells $q \in \tau_h$ for which common interface of p and q is a line segment, denoted by e_{pq}, of non-zero measure $m(e_{pq})$ (see Figure 1 for 2D case).

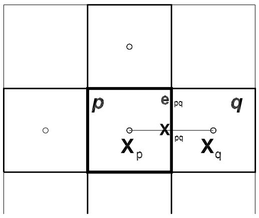


Fig. 1. Notation for the finite volume scheme in 2D case.

We assume that for every p, there exists a representative point $x_p \in p$ such, that for every pair $p, q, q \in N(p)$, the vector $\frac{x_q - x_p}{|x_q - x_p|}$ is equal to unit vector $\vec{\nu}_{pq}$ which is normal to the common interface e_{pq} and oriented from p to q. In the simple case of a uniform grid we can take x_p just as a center of the pixel. Let x_{pq} be the intersection of the line segment e_{pq} and the segment $\overline{x_p x_q}$. Then we define *transmitivity coefficients*

$$T_{pq} := \frac{m(e_{pq})}{|x_q - x_p|}. \tag{12}$$

Now, we are ready to write the fully discrete finite volume scheme for solving the segmentation problem (1)–(3):

Let $0 = t_0 \le t_1 \le \cdots \le t_{N_{\max}} = T$ *denote the time discretization with* $t_n = t_{n-1} + k$, *where* k *is the time step. For* $n = 0, \ldots, N_{\max} - 1$ *we look for* $\overline{u}_p^{n+1}$, $p \in \tau_h$ *satisfying*

$$\frac{\overline{u}_p^{n+1} - \overline{u}_p^n}{k|\nabla G_\sigma * \tilde{u}_p^n|_\varepsilon} m\,(p) = \sum_{q \in N(p)} T_{pq}\ g(|\nabla I^\rho(x_{pq})|) \frac{\left(\overline{u}_q^{n+1} - \overline{u}_p^{n+1}\right)}{|\nabla G_\sigma * \tilde{u}_{pq}^n|_\varepsilon}. \tag{13}$$

Applying the Dirichlet boundary conditions (3), the scheme (13) gives the system of linear equations with coefficients given by the following equivalent form

$$\left(\frac{m(p)}{k} \frac{1}{|\nabla G_\sigma * \tilde{u}_p^n|_\varepsilon} + \sum_{q \in N(p)} T_{pq} \frac{g(|\nabla I^\rho(x_{pq})|)}{|\nabla G_\sigma * \tilde{u}_{pq}^n|_\varepsilon} \right) \overline{u}_p^{n+1}$$
$$- \sum_{q \in N(p)} T_{pq} \frac{g(|\nabla I^\rho(x_{pq})|)}{|\nabla G_\sigma * \tilde{u}_{pq}^n|_\varepsilon} \overline{u}_q^{n+1} = \frac{m(p)}{k} \frac{1}{|\nabla G_\sigma * \tilde{u}_p^n|_\varepsilon} \overline{u}_p^n. \tag{14}$$

One can easily see that the system matrix is an M-matrix, so the linear system is uniquely solvable and its solution fulfils discrete minimum-maximum principle (L_∞-stability) at every discrete scale step. We start the computations by $\overline{u}_p^0$, $p \in \tau_h$, corresponding to given values of the initial point-of-view surface. In the scheme we work with two types of gradients: $\nabla G_\sigma * \tilde{u}_{pq}^n$ and $\nabla I^\rho(x_{pq})$ computed on the control volume boundary ∂p and $\nabla G_\sigma * \tilde{u}_p^n$ computed on the control volume p. Here, $\tilde{u}^n$ is a mirror extension of the data computed in the nth discrete time step and $\tilde{u}_p^n$, $\tilde{u}_{pq}^n$ denotes its approximate values inside p and along ∂p, respectively.

Remark 1. Computing the gradients on ∂p

Since we consider piecewise constant approximations, we can replace the convolution (integral) by a weighted average over the neighbouring pixels. The convolved gradients of u along the boundary ∂p are approximated by values in $x_{pq}, q \in N(p)$ and then the corresponding vector $\nabla G_\sigma * \tilde{u}_{pq}^n$ is evaluated using the following property of convolution

$$\frac{\partial (G_\sigma * \tilde{u})}{\partial x}(x_{pq}) = \left(\frac{\partial G_\sigma}{\partial x} * \tilde{u}\right)(x_{pq}) .$$

In such way we get

$$\nabla G_\sigma * \tilde{u}^n\,(x_{pq}) = \left(\sum_r \overline{u}_r^n \int_r \frac{\partial G_\sigma}{\partial x}(x_{pq} - s)\,\mathrm{d}s, \sum_r \overline{u}_r^n \int_r \frac{\partial G_\sigma}{\partial y}(x_{pq} - s)\,\mathrm{d}s\right). \qquad (15)$$

The previous sum is evaluated over the control volumes r which surround the point x_{pq}. If we choose a compactly supported smoothing kernel with the support in a ball $B_\sigma(0)$ with radius σ, then the sum is restricted only to control volumes contained in $B_\sigma(x_{pq})$, the ball centered at x_{pq}. In our experiments with FVM we use the function

$$G_\sigma(x) = \frac{1}{Z} \mathrm{e}^{\frac{|x|^2}{(|x|^2 - \sigma^2)}},$$

where the constant Z is chosen so that G_σ has a unit mass. Coefficients of the sum (15), namely $\int_r \nabla G_\sigma\,(x_{pq} - s)\,\mathrm{d}s$ can be precomputed in advance using a computer algebra system, e. g. Mathematica. In our numerical experiments we used exclusively 2D square and 3D cubic grids with cells of unit size and $\sigma = 1$. We use weights given by Figure 2, the more exact values of weights for various values of σ in 2D and 3D can be found e. g. in [6].

Remark 2. Computing the gradients on control volume p

Using idea of Walkington [17], for approximation of the convolved gradient inside finite volume we use simple averaging given (in case of regular square grids) by

$$|\nabla G_\sigma * \tilde{u}_p^n| \approx \frac{1}{\mathrm{card}(N(p))} \sum_{q \in N(p)} |\nabla G_\sigma * \tilde{u}_{pq}^n| \qquad (16)$$

3. DISCUSSION ON THE BASIC LEVEL SET MODEL (9) AND ITS SOLUTION BY THE FV SCHEME

Aim of this subsection is to compare the work of the scheme (13) adjusted for (9) with a known exact solution.

Let u_0 in 2D be given by

$$u_0(x,y) = 1 - \frac{\sqrt{(x-s_x)^2 + (y-s_y)^2}}{r(0)}.$$

The level line for $u_0(x,y) = 0$ is a circle of a radius $r(0)$ centered in (s_x, s_y). For this u_0, the exact solution obtained by applying the level set equation (9) is known to be a circle with the radius

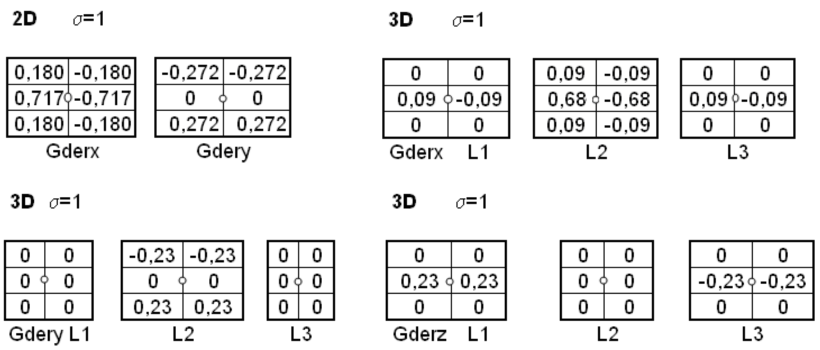
$$r(t) = \sqrt{r(0)^2 - 2t}, \quad t \in (0,T), \quad T = r(0)^2/2. \tag{17}$$

Let us set $g \equiv 1$ in (1) and apply the finite volume scheme (13) to u_0. We compare the numerical solution obtained in this way with (17). The setting of parameters for FV scheme for this example was as follows: the size h of a square control volume was set to 1, u^0 was created over 256×256 control volumes, $\varepsilon = 10^{-8}$ and the size of time step k is 65. The radius $r(0)$ is set to 102, $s_x = s_y = 128$. In this example and all the computations introduced in this paper σ was set to 1. Then we have three different weights (see Figure 2) and six neighbors involved in computing $|\nabla G_\sigma * \tilde{u}^n_{pq}|$. The Figure 3, on the left, shows the exact solution at chosen time steps $t_0, t_1, t_2, t_3, t_4, t_5$, where $t_i = i * 65$, drawn with dashed lines and the solution obtained by (13) drawn with solid lines.

In 3D,

$$r(t) = \sqrt{r(0)^2 - 4t}, \quad t \in (0,T), \quad r(0)^2/4. \tag{18}$$

If we set σ again to 1, for each sum of the gradient we have 18 mutually rotated



2D σ=1

Gderx

0,180	-0,180
0,717	-0,717
0,180	-0,180

Gdery

-0,272	-0,272
0	0
0,272	0,272

3D σ=1

Gderx L1

0	0
0,09	-0,09
0	0

L2

0,09	-0,09
0,68	-0,68
0,09	-0,09

L3

0	0
0,09	-0,09
0	0

3D σ=1

Gdery L1

0	0
0	0
0	0

L2

-0,23	-0,23
0	0
0,23	0,23

L3

0	0
0	0
0	0

3D σ=1

Gderz L1

0	0
0,23	0,23
0	0

L2

0	0
0	0
0	0

L3

0	0
-0,23	-0,23
0	0

Fig. 2. Precomputed weights for $\sigma = 1$ used in computations. L1, L2 and L3 are horizontal layers of voxels involved in computation of the gradient.

weights [6]. Some of these weights are zeroes, some can be neglected and some were set to the same value, though they slightly differ. At the end we got three different weights and reduced the number of multiplications to 9 for a gradient and to 3 to compute its magnituded. The visual results are presented in Figure 3 on the right for $t = 100, 200 \ldots, 500$.

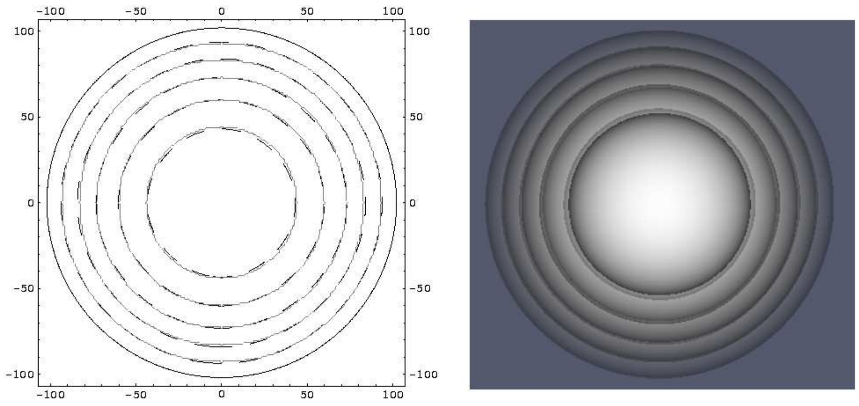


Fig. 3. On the left, the comparison with the exact solution for $\sigma = 1$ (dashed lines) in 2D, on the right, the spheres of the similar experiment in 3D.

4. ADAPTIVE FVM SCHEMES

The aim of the adaptive algorithm is to reduce the number of unknowns (i. e. control volumes) in the resulting linear system. First we describe the way, in which the adaptive grid is created and then, how an nonadaptive algorithm is modified into the adaptive one. We try to stress the main ideas used for creating the adaptive grid.

First, the criterion saying when the cells can be merged into bigger ones, is based on comparing the intensities: *cells can be merged into a bigger one, if all subcells' intensities are within a prescribed tolerance.* We created a square "balanced" grid. i. e. such grid, where every cell is a square and for any two neighboring cells it holds, that ratio of their sides's sizes is $1:1, 1:2$ or $2:1$. For adjacent cells of nonequal size, the "hanging point" is always in the middle of the bigger one. Such a grid can be easily obtained by a quadtree technique. In more details, this technique is described e. g. in [4, 5, 6]. Examples od adaptive grids are depicted un Figure 12 and 13. Modification of a nonadaptive scheme into an adaptive one has been done with help of in transmisivity coefficients T_{pq}. We can set the distnace, over which we compute the gradient and which is in the denominator of T_{pq}, to the average length of sides of two neighboring cells. Since the grids are balanced, in 2D we have

$$\begin{aligned} T_{pq} &:= 1 && \text{if two inspected adjacent cells } p, q \text{ are of equal size} \\ T_{pq} &:= \frac{2}{3} && \text{otherwise.} \end{aligned} \tag{19}$$

The second possibility how to modify the nonadaptive algorithm into an adaptive one is such, that working on the same grid as described above, we make the diffusion faster in regions with greater size of cells. We can achieve this by making the distance for gradients smaller (now we mean the gradient, which is involved in T_{pq}, equal to the size h of cells on the basic level. If l_p and l_q denote lengths of sides of control volumes p and q, in 2D we come to a formula

$$T_{pq} = \min\left\{\frac{l_p}{h}, \frac{l_q}{h}\right\}. \tag{20}$$

Scheme 1

$$\left(\frac{l_p^2}{k}\frac{1}{|\nabla G_\sigma * \overline{u}_p^n|_\varepsilon} + \sum_{q\in N(p)} T_{pq}\frac{g_{pq}^{\sigma,n}(\overline{u}_{h,k})}{|\nabla G_\sigma * \overline{u}_p^n\,(x_{pq})\,|_\varepsilon}\right)\overline{u}_p^{n+1} - \sum_{q\in N(p)} T_{pq}\frac{g_{pq}^{\sigma,n}(\overline{u}_{h,k})\overline{u}_q^{n+1}}{|\nabla G_\sigma * \overline{u}_p^n\,(x_{pq})\,|_\varepsilon} = \frac{l_p^2}{k}\frac{1}{|\nabla G_\sigma * \overline{u}_p^n|_\varepsilon}\overline{u}_p^n \tag{21}$$

with T_{pq} defined by (19) and

Scheme 2

$$\left(\frac{l_p}{k}\frac{1}{|\nabla G_\sigma * \overline{u}_p^n|_\varepsilon} + \sum_{q\in N(p)} T_{pq}\frac{g_{pq}^{\sigma,n}(\overline{u}_{h,k})}{|\nabla G_\sigma * \overline{u}_p^n\,(x_{pq})\,|_\varepsilon}\right)\overline{u}_p^{n+1} - \sum_{q\in N(p)} T_{pq}\frac{g_{pq}^{\sigma,n}(\overline{u}_{h,k})\overline{u}_q^{n+1}}{|\nabla G_\sigma * \overline{u}_p^n\,(x_{pq})\,|_\varepsilon} = \frac{l_p}{k}\frac{1}{|\nabla G_\sigma * \overline{u}_p^n|_\varepsilon}\overline{u}_p^n \tag{22}$$

where T_{pq} modifies into

$$T_{pq} := 1 \quad \text{if two inspected adjacent cells } p, q \text{ are of equal size} \tag{23}$$

$$T_{pq} := \frac{1}{2} \quad \text{otherwise.} \tag{24}$$

The 3D schemes can be obtained in a similar manner [6]. The second scheme gives usually a better visual effect, because diffusion on cells with size greater than 1 is faster than if we use the scheme 1. Anyway, both schemes lead to the “staircase effect”, caused by the fact, that greater cells have zero curvature and isolines stop moving. Though this problem is just visual – the aim is to extract isolines or isosurfaces – it can be surpressed in two ways: we can impose another demand on the adaptive grid – to merge cells into a bigger one, also the adjacent cells of a half size must have their intensities within a prescribed tolerance (not necessarily the same as the *threshold*) or we can use ideas described in experiment 5.

5. NUMERICAL EXPERIMENTS

In all algorithms used for these experiments, we use unit square grids and the size of scale step k is set to 60. In all experiments except of 3D ones, the size of pictures was set to 256×256. For the Perona–Malik function g we use the function $g(s) = \frac{1}{1+Ks^2}$ with $K > 0$. We run computations until the change in the segmentation function is below a certain threshold determining stopping criterion *tol*, usually set to 0.5. For u_0 we use the function $u_0(x, y) = 1/\sqrt{(x - s_x)^2 + (y - s_y)^2 + height}$. The coefficients in linear system (14) were adjusted so that the diagonal term is equal to 1. The algorithms are controlled by following parameters: ε in regularization term (10), *height* in u_0, K in Perona–Malik function g and σ in convolution term is set to 1 with weights from Figure 2. The adaptive algorithms use a parameter *threshold* to set the coarsening criterion.

Experiment 1. This experiment is a simple example of segmentation and is depicted in Figure 4. An object – a circle – is given by its boundary, which is properly closed and there is no noise in the image. During the evolution all the level lines of the function u shrink with the speed depending on their curvature, except of the level lines in the vicinity of the image edges, where, due to the function g, the speed is slowed down. The "steady" state of a particular level line corresponds to a boundary of a segmented object.

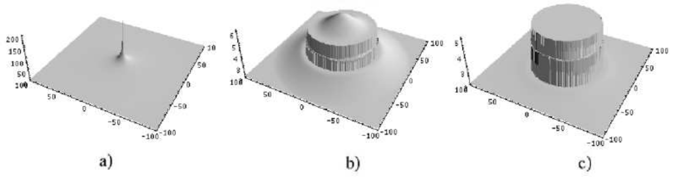


Fig. 4. a) the initial "peak" function b) 10 scale steps of the algorithm c) 60 scale steps of the algorithm.

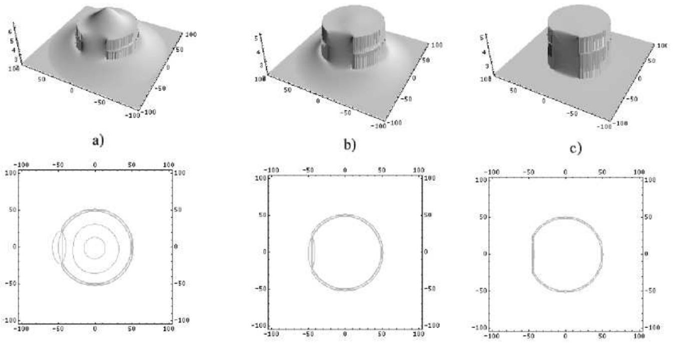


Fig. 5. Experiment 2. Segmentation of an object with an open boundary.

Experiment 2. If the boundary of a segmented object is not closed, the algorithm is still able to detect the object and complete the missing part of the boundary with a line segment. If there is a missing part of the boundary of the segmented object, the intensity function is "spilled out", but the isolines are straightened by the mechanism of the mean curvature motion (9). To achieve this behavior of the algorithm, ε must be set to be small (10^{-8}) (see Figure 4).

Experiment 3. In 3D, we present examples showing completion of missing parts of the boundary for a sphere and for a cube (Figure 6 and Figure 7). Parameter

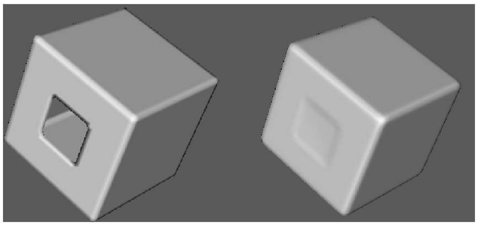


Fig. 6. Experiment 3. The original data and the result after 45 scale steps.

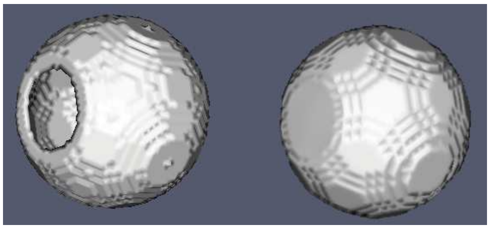


Fig. 7. Experiment 3. The original data and the result after 40 scale steps.

ε for (10) was set to 10^{-12}, *height* in u_0 equals 1, K in Perona–Malik function g is set to 800 and ω for Gauss–Seidel SOR to 1.7. In these 3D examples, size of data was $60 \times 60 \times 60$.

Experiment 4. This experiment demonstrates, that the model (1)–(2) is well suited for segmentation of noisy images as well, even for objects having oclusions in their boundaries. Figure 8 shows the original object, which is disturbed by the additive noise. The segmentation function obtained by the model is in the middle and the extracted boundary on the right of the picture.

Experiment 5. The Figure 9 demonstrates, that in segmentation both principles (levelset and mean curvature for graphs) can be used. Let us have an initial image for segmentation whose boundary is not closed. Let us process the problem with (14) first for $\varepsilon = 10^{-12}$ (Figure 9 a)) and then for $\varepsilon = 1$ (Figure 9 b)). The case a) shows, that though missing boundary is completed rather well, the segmentation function is not flat and e. g. thresholding using histogram would be problematic. In the case b) the segmentation function is more flat, but it spills out through the

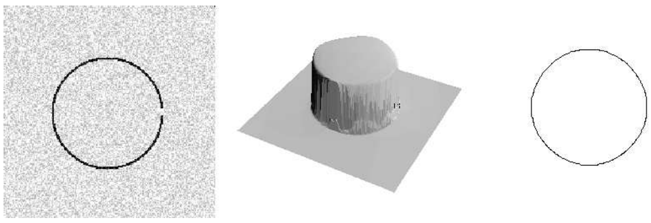


Fig. 8. The segmentation of noisy image.

hole in the boundary. However, if we look at the corresponding isolines, we can observe, that switching to the level set principle, i. e. making ε small, leads to their straightening and thus to the Figure 9 c).

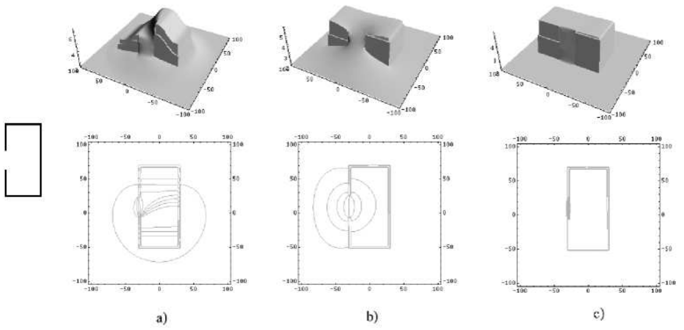


Fig. 9. Meaning of ε. a) Segmentation function after 20 scale steps with ε small, b) 50 scale steps with ε =1, c) switching to ε small – result after 150 scale steps.

On the other hand, we can start with $\varepsilon = 10^{-12}$ for a certain number of steps necessary to form shocks on the part of the missing boundary (coming to Figure 9 a)). Then, to make the segmentation function flat, we use the following model:

$$\frac{\partial_t u}{\sqrt{(g(|\nabla u|))^2 + |\nabla u|^2}} = \nabla . \left(\frac{\nabla u}{\sqrt{(g(|\nabla u|))^2 + |\nabla u|^2}} \right), \tag{25}$$

where g is the following function:

$$\begin{aligned} g(v) &= 10^{-6} && \text{for} \quad 0 \le t \le T_1 \\ g(v) &= \frac{1}{1 + K * v * v} && \text{for} \quad T_1 < t \le T. \end{aligned}$$

Figure 10 displays the following experiment: the image on the left depicts the result at T_1 ($T_1 = 40 * k,\ k = 60$), when shocks are developed. Then ε in the

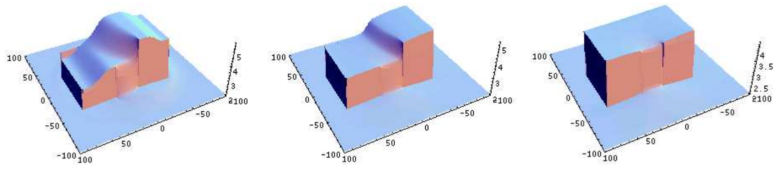


Fig. 10. Model (25). a) forming the shock, b) and c) flattening f the segmentation function.

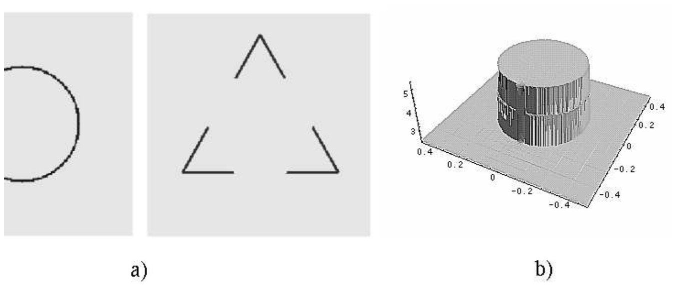


Fig. 11. a) Data for experiments 6 and 7, b) staircase effect of the adaptive algorithms.

algorithm changes according to the gradients: this parameter stays small on the highest gradients and is greater for smaller gradients. Images b) and c) show the evolution in the middle and final steps.

Experiment 6. In this example we use adaptive algorithm I to process the data depicted in Figure 11 a) on the left. In the adaptive algorithm, the *threshold* is set to 0.005. The linear system is solved by Gauss–Seidel SOR with relaxation factor ω set to 1.75, ε is set to 10^{-4}. For $tol = 0.1$ the nonadaptive algorithm achieved the time 21,2 s. The adaptive algorithm I achieved time 8.1 s ending with 4867 grid elements (the initial number of elements is 65536). The visual result, the decrease of elements and the final grid are depicted in Figure 12.

Experiment 7. It processes the data from Figure 11 a) on the right. The time achieved by corresponding *nonadaptive* algorithm was 52,8 s. In this experiment, adaptive algorithm 2 is used in two ways. First we set the stopping criterion to 0.01 and ε is set to 0.0005 for all the scale steps.The algorithm needed 412 scale steps and ended with 12 562 control volumes of 65536 initial and needed 25.5 s (case 1). The similar visual and time result could be achieved, if we set stopping criterion to 0.005 and after achieving change less then 0.1 we change *eps* to 0.02. If we had set stopping criterion only to 0.01 the achieve time would be 19 s but with slightly worse visual result. The final number of grid elements is 6139. If we decreased the stopping criterion to 0.005, we would achieve better visual result and the time 24.5 s

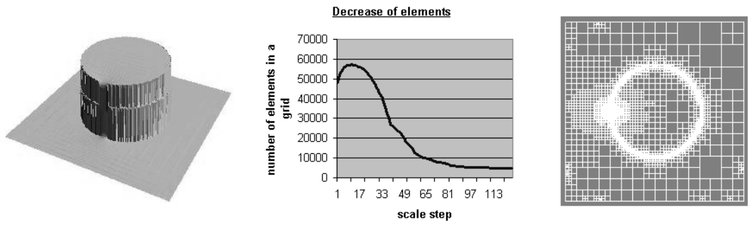


Fig. 12. The visual result of the experiment 6, the decrease of elements and the final grid.

in 640 scale steps, which is twice better than the nonadaptive algorithm. The final number of grid elements is 6106 (case 3). The visual result, decrease of grid cells resp. resulting grids are depicted in Figure 13

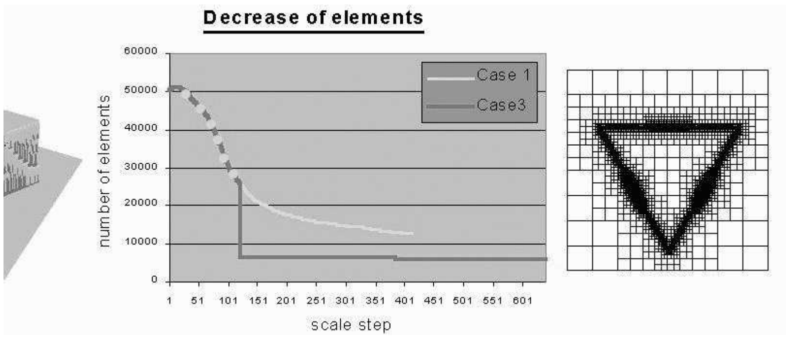


Fig. 13. The visual result of the experiment 7, the decrease of elements and the final grid.

ACKNOWLEDGEMENT

This work was supported by grant APVT-20-040902.

(Received March 13, 2006.)

REFERENCES

[1] L. C. Evans and J. Spruck: Motion of level sets by curvature I. In: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science. Cambridge University Press, Cambridge 1999.

[2] A. Handlovičová, K. Mikula, and F. Sgallari: Semi–implicit complementary volume scheme for solving level set like equations in image processing and curve evolution. Numer. Math. *93* (2003), 675–695.

[3] G. Kanizsa: Organization in Vision. Hardcover 1979.

[4] Z. Krivá and K. Mikula: An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing. J. Visual Communication and Image Representation *13* (2002), 22–35.

[5] Z. Krivá and K. Mikula: An adaptive finite volume scheme in processing of color images. In: Proc. ALGORITMY 2000, Conference on Scientific Computing, Podbanské 2000, pp. 174–188.

[6] Z. Krivá: Adaptive Finite Volume Methods in Image Processing. Edícia vedeckých prác, STU Bratislava, Stavebná fakulta 2004.

[7] Z. Krivá: Segmentation combining approaches based on mean curvature. In: Mathematical Modelling and Analysis 2005, Proc. 10th International Conference MMA2005&CMAM2, Trakai 2005, pp. 433–441.

[8] K. Mikula, A Sarti, and F. Sgallari: Co-volume method for Riemennian mean curvature flow in subjective surface multiscale segmentation. Comput. Visual Sci. *9* (2006), 1, 23–31.

[9] K. Mikula, A. Sarti, and F. Sgallari: Co-volume level set method in subjective surface based medical image segmentation. In: Handbook of Biomedical Image Analysis, Kluwer Academic/Plenum Publishers, Dordrecht 2005, pp. 583–626.

[10] K. Mikula and A. Sarti: Parallel co-volume subjective surface method for 3D medical image segmentation. In: Deformable Model (J. Suri, ed.), Springer–Verlag, Berlin 2006, to appear.

[11] S. Osher and J. A. Sethian: Front propagating with curvature dependent speed: algorithms based on the Hamilton–Jacobi formulation. J. Comput. Phys. *79* (1988), 12–49.

[12] A. Sarti, R. Malladi, and J. A. Sethian: Subjective surfaces: A method for completing missing boundaries. Proc. Nat. Acad. Sci. U.S.A. *12* (2000), 97, pp. 6258–6263.

[13] A. Sarti and G. Citti: Subjective surfaces and Riemannian mean curvature flow graphs. Acta Math. Univ. Comenianae *70* (2001), 1, 85–104.

[14] A. Sarti, R. Malladi, and J. A. Sethian: Subjective surfaces: A geometric model for boundary completion. Internat. J. Computer Vision *46* (2002), 3, 201–221.

[15] J. A. Sethian: Numerical algorithm for propagating interfaces: Hamilton–Jacobi equations and conservation laws. J. Diff. Geom. *31* (1990), 131–161.

[16] J. A. Sethian: Level set methods and fast marching methods. In: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science. Cambridge University Press, Cambridge 1999.

[17] N. J. Walkington: Algorithms for computing motion by mean curvature. In: SIAM J. Numer. Anal. *33* (1996), 6, 2215–2238.

Zuzana Krivá, Department of Mathematics, Faculty of Civil Engineering, Slovak University of Technology Bratislava, Radlinského 11, 813 68 Bratislava. Slovak Republic.
e-mail: kriva@math.sk