Co-volume level set method in subjective surface based medical image segmentation

Karol Mikula[*], Alessandro Sarti[†], Fiorella Sgallari[‡]

Abstract

We present application of the semi-implicit complementary volume numerical scheme to solving level set formulation of Riemannian mean curvature flow of graphs in image segmentation, edge detection, missing boundary completion and subjective contour extraction. Our computational method is robust, efficient and stable without any restriction on a time step. The computational results related to medical image segmentation with partly missing boundaries and subjective contour extraction are presented.

1 Introduction

It is well-known that the so-called *level set equation* [43, 54, 55, 42]

$$u_t = |\nabla u| \nabla . \left(\frac{\nabla u}{|\nabla u|} \right) \tag{1}$$

for curvature-driven motion as well as its nontrivial generalizations are well suited for image processing applications and they are often used nowadays. In this chapter we deal with a specific equation of mean curvature flow type [48, 49, 50], namely

$$u_t = \sqrt{\varepsilon^2 + |\nabla u|^2} \nabla . \left(g(|\nabla G_\sigma * I^0|) \frac{\nabla u}{\sqrt{\varepsilon^2 + |\nabla u|^2}} \right) \tag{2}$$

[*]Department of Mathematics, Slovak University of Technology, Radlinského 11, 813 68 Bratislava, Slovakia (e-mail: mikula@vox.svf.stuba.sk)

[†]DEIS, University of Bologna, Via Risorgimento 2, 40136 Bologna, Italy (e-mail: asarti@deis.unibo.it)

[‡]Department of Mathematics, University of Bologna, Piazza di Porta S. Donato 5, 40127 Bologna, Italy (e-mail: sgallari@dm.unibo.it)

where $u(t,x)$ is an unknown (segmentation) function defined in $Q_T \equiv [0,T] \times \Omega$. $\Omega \subset I\!R^d$ is a bounded domain with a Lipschitz continuous boundary $\partial\Omega$, $[0,T]$ is a time interval, I^0 is a given image and $\varepsilon > 0$ is a parameter. The equation is accompanied with zero Dirichlet boundary conditions and initial condition

$$u(t,x) = u^D \quad \text{in } [0,T] \times \partial\Omega, \tag{3}$$

$$u(0,x) = u^0(x) \quad \text{in } \Omega. \tag{4}$$

Without lost of generality we may assume $u^D = 0$. The Perona-Malik function $g : I\!R_0^+ \to I\!R^+$ is nonincreasing, $g(0) = 1$, admitting $g(s) \to 0$ for $s \to \infty$ [45]. Usually we use the function $g(s) = 1/(1 + Ks^2)$, $K \geq 0$. $G_\sigma \in C^\infty(I\!R^d)$ is a smoothing kernel, e.g. the Gauss function

$$G_\sigma(x) = \frac{1}{(4\pi\sigma)^{d/2}} e^{-|x|^2/4\sigma} \tag{5}$$

which is used in pre-smoothing of image gradients by the convolution

$$\nabla G_\sigma * I^0 = \int\limits_{I\!R^d} \nabla G_\sigma(x - \xi) \tilde{I}^0(\xi) d\xi, \tag{6}$$

with $\tilde{I}^0$ the extension of I^0 to $I\!R^d$ given by periodic reflection through the boundary of image domain. The computational domain Ω is usually a subdomain of the image domain, it should include the segmented object. In fact, in most situations Ω corresponds to image domain itself. We assume that an initial state of the segmentation function is bounded, i.e. $u^0 \in L_\infty(\Omega)$. For shortening notations, we will use abbreviation

$$g^0 = g(|\nabla G_\sigma * I^0|). \tag{7}$$

Due to smoothing properties of convolution we always have $1 \geq g^0 \geq \nu_\sigma > 0$ [5, 27].

The equation (2) is a regularization, in the sense $|\nabla u| \approx |\nabla u|_\varepsilon = \sqrt{\varepsilon^2 + |\nabla u|^2}$ [19], of the segmentation equation suggested in [7, 8, 9, 30, 31], namely

$$u_t = |\nabla u| \nabla . \left(g^0 \frac{\nabla u}{|\nabla u|} \right). \tag{8}$$

However, while in [19] the ε-regularization was used just as a tool to prove existence of a viscosity solution of the level set equation (see also [10, 12]), in our work ε is a modelling parameter. As we will see later, it can help in suitable denoising and completing of missing boundaries in images. Such regularization can be interpreted as a mean curvature flow of graphs with respect to a specific Riemann metric given by the image features [49].

The idea to use Riemannian mean curvature flow of graphs to compute the so-called subjective contours [29] originates in [48, 49, 50]. The subjective surfaces method, developed there, has been successfully used to complete missing boundaries of objects in digital 2D and 3D data sets and thus it is a powerfull method for segmentation of highly noisy, e.g. medical, images. In this chapter we follow the same idea.

Initially, a "point-of-view" surface, given by an observer (user) chosen fixation point inside the image, is taken as u^0 (see e.g. Figure 11 top right). Then this initial state of the segmentation function is evolved by equation (2), until the so-called subjective surface arises (see e.g. Figure 11 bottom right or Figure 14 top row). 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. The idea to follow evolution of the graph of segmentation function [48, 49, 50] and not to follow evolution of a particular level set of u is new in comparison with other level set methods used in image segmentation (cf. [6, 36, 7, 8, 9, 30, 31]). In standard level set approach, the redistancing [55, 42] is used to keep unit slope along the level set of interest (e.g. along segmentation curve). In such approach the evolution of u itself is forgotten at every redistancing step. Such solution prevents steepening of u and one cannot obtain the subjective surfaces. In our computational method we do not impose any specific requirements (e.g., redistancing) to solution of the level set equation, the numerically computed segmentation function can naturally evolve to a "piecewise constant steady state" result of the segmentation process.

For numerical solution of the nonlinear diffusion equation (2), governing Riemannian mean curvature flow of graphs, we use semi-implicit complementary volume (called also co-volume or finite volume-element) method. Since (2) is regularization of (8), for the curvature driven level set flow (8) or for some other form of the level set equation (1), the method can be used as well (cf. [25, 21]).

For time discretization of nonlinear diffusion equations there are basically three posibilities – implicit, semi-implicit or explicit schemes. For spatial discretization usually finite differences, finite volumes or finite element methods are used. The co-volume technique is a combination of finite element and finite volume methods. Implicit, i.e. nonlinear, time discretization and co-volume technique for solution of the level set equation was introduced in [56]. The efficient co-volume level set method based on semi-implicit, i.e. linear, time discretization was given and studied in [25]. In [25], the method was applied to image smoothing nonlinear diffusion level set equation; here we apply the method to image segmentation and completion of missing boundaries.

Let us note that equation (8) can be rewritten into an advection-diffusion form

$$u_t = g^0 |\nabla u| \nabla . \left(\frac{\nabla u}{|\nabla u|} \right) + \nabla g^0 . \nabla u. \tag{9}$$

Various finite difference schemes [7, 8, 9, 30, 31, 48, 49, 50] are usually based on this form using up-winding in advection term and explicit time stepping. Our co-volume technique relies on discretization of the basic form (8), or more precisely on its regularization (2), and we use its integral (weak, variational) formulation. In such a way, the discretization scheme naturally respects a variational structure of the problem, it gives clear discrete form of local mass balance, and it naturally fulfills discrete minimum-maximum principle (L_∞-stability). The semi-implicit discretization in time yields such stability property (i.e. no spurious oscillations appear in our solution) for any length of 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. Since in nonlinear diffusion problems (like the level set equation) the coefficients depend on the solution itself and thus they must be recomputed in every discrete time update, an overall CPU time for explicit scheme can be tremendous. On the other hand, the implicit time stepping as in [56], altough it is unconditionally stable, leads to solution of nonlinear systems in every discrete time update. For the level set like problems there is no guarantee for convergence of a fast Newton solver and fixed point like iterations are very slow [56]. From this point of view, the semi-implicit method seems to be optimal regarding stability and efficiency. In every time update we solve linear system of equations which can be done efficiently using, e.g., suitable preconditioned iterative linear solvers.

In the next Section 2 we discuss various curve evolution and level set models leading to segmentation equations (8) and (2). In Section 3 we introduce our semi-implicit co-volume level set method for solving these equations and discuss some of its theoretical properties and implementation aspects. In section 4 we discuss numerical experiments.

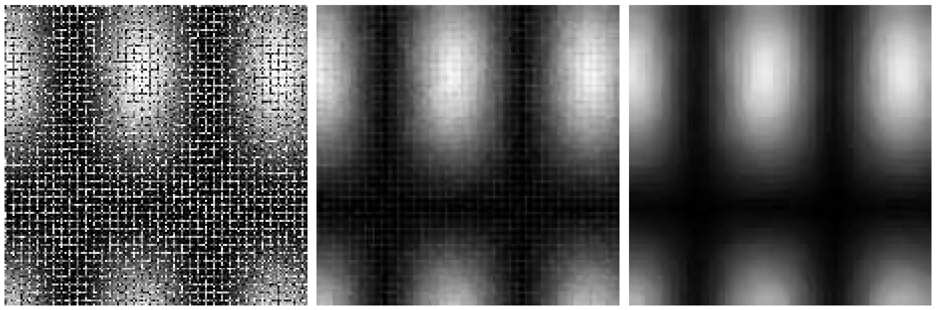


Figure 1: Image corrupted by a structural noise (left), result of filtering by level set equation after 2 (middle) and 10 (right) discrete scale steps.

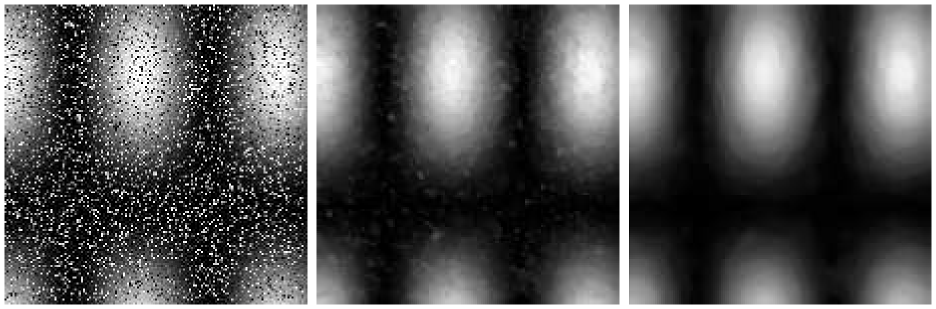


Figure 2: Initial image corrupted by salt and pepper noise (left), result of filtering by level set equation after 2 (middle) and 10 (right) discrete scale steps.

2 Discussion On Related Mathematical Models

The level set equation (1) has large significance in axiomatization of image processing and computer vision [1]. It fulfills the so-called *morphological principle*: if u is a solution then, for any nondecreasing function φ, $\varphi(u)$ is a solution as well. It means that level sets of a solution u move independently of each other, or in other words, they diffuse only intrinsically (in tangential direction) and there is no diffusion across level sets in the normal direction. In that sense it provides a directional smoothing of the image along its level lines. We illustrate the smoothing effect of the level set equation in Figures 1 (removing structural noise) and 2 (removing salt and pepper noise) [25].

In image filtration, the initial condition for the level set equation (1) is given by the image greylevel intensity I^0 itself, i.e. $u^0 = I^0$, and usually zero Neumann boundary conditions are used. The solution $u(t,x)$ gives a family of *scaled* (filtered, smoothed) versions of $I^0(x)$. The parameter t is understood as *scale*, and the process of nonlinear selective smoothing is called image multiscale analysis [1]. In [25], the linear semi-implicit co-volume method to solve image selective smoothing equation [2]

$$u_t = g(|\nabla G_\sigma * u|)|\nabla u|\nabla.\left(\frac{\nabla u}{|\nabla u|}\right) \tag{10}$$

has been suggested and studied. Equation (10) can be used for edge-preserving smoothing in a similar way as the so-called Perona-Malik equation [45, 5, 27, 28, 37, 41, 2, 1, 38, 24, 25, 26], see Figure 3.

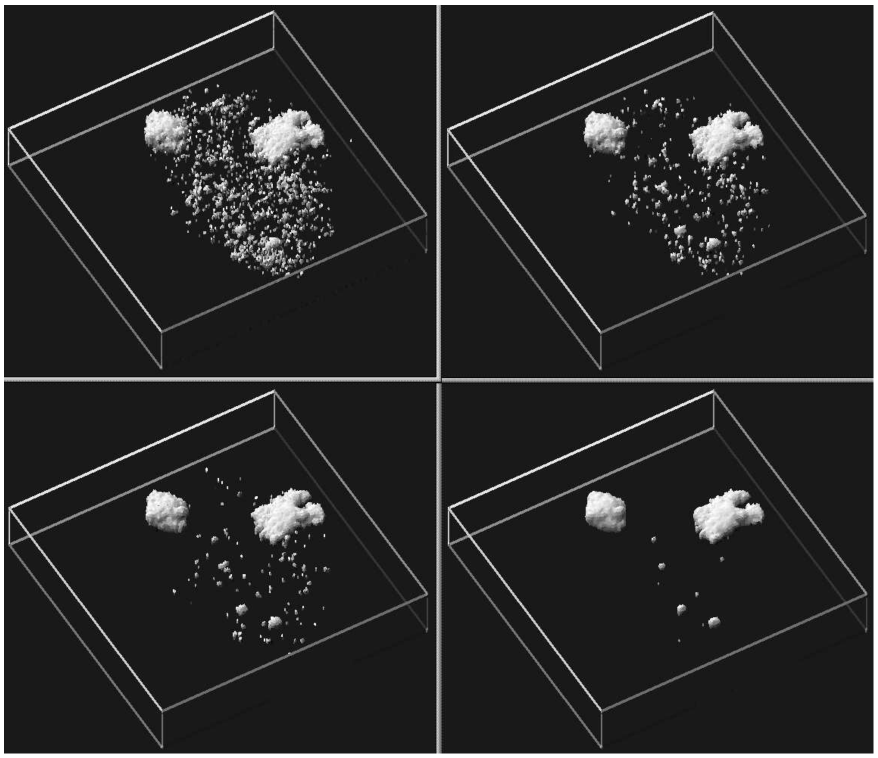


Figure 3: Extraction of two chromosomes in a human cell using geometrical diffusion (10) [24].

The aim of segmentation is to find boundaries of a distinguished object of an image. In generic situation these boundaries correspond to edges.

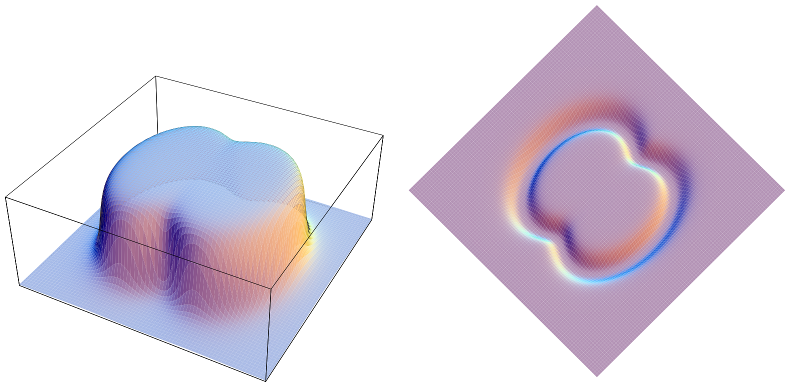


Figure 4: A graph of the image intensity function $I^0(x)$ corresponding to a "dumb-bell" image (left, see also Figure 5) and a graph of the function $g(|\nabla I^0(x)|)$ (right) where a narrow valley along the edge can be observed. (Color Slide).

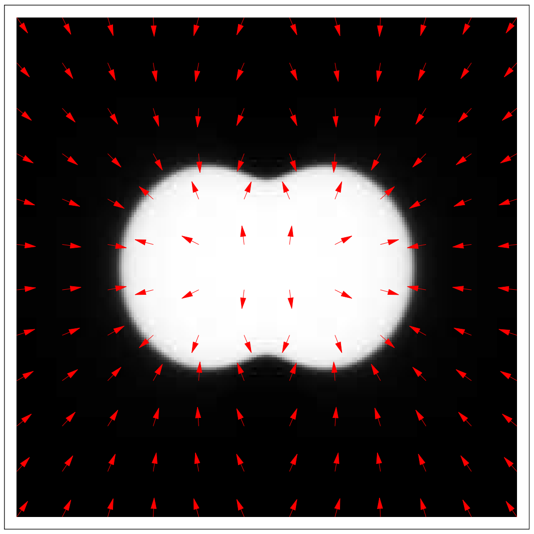


Figure 5: Image given by the intensity $I^0(x)$ from Figure 4 left and the arrows representing the vector field $-\nabla g(|\nabla I^0(x)|)$. (Color Slide).

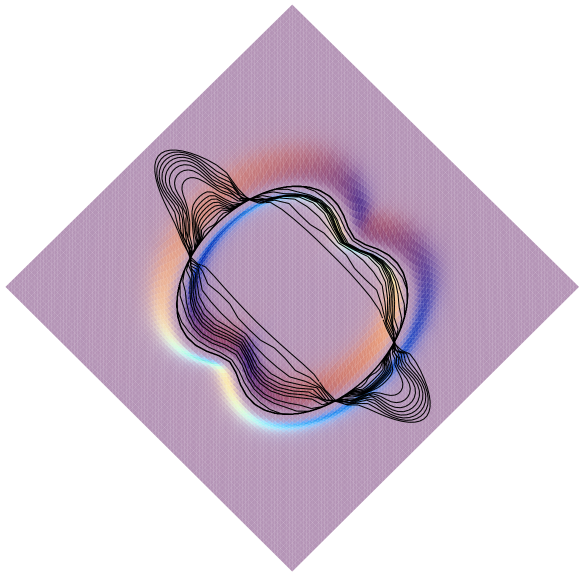


Figure 6: An initial ellipse driven by the vector field $-\nabla g(|\nabla I^0(x)|)$ down to the valley to find the edge in the image I^0. (Color Slide).

However, in the presence of a noise or in images with oclusions or subjective contours these edges can be very irregular or even interrupted. Then the analysis of the scene and segmentation of objects become a difficult task.

In the so-called active contour models [32] an evolving family of curves converging to an edge is constructed. A simple approach (similar to various discrete region growing algorithms) is to put small seed, e.g. small circular curve, inside the object and then evolve the curve to find automatically the object boundary. For such moving curves the level set models have been introduced in the last decade. A basic idea is that moving curve corresponds to a specific level line of the level set function which solves some reasonable generalization of equation (1). The level set methods have several advantages among which, independence of dimension of the image and topology of objects are probably the most important. However, a reader can be interested also in the so-called direct (Lagrangian) approaches to curve and surface evolution (see e.g. [16, 17, 18, 39, 40]).

First simple level set model with the speed of segmentation curve modulated by $g(|\nabla I^0(x)|)$ (or more precisely by $g(|\nabla G_\sigma * I^0|)$), where g is a

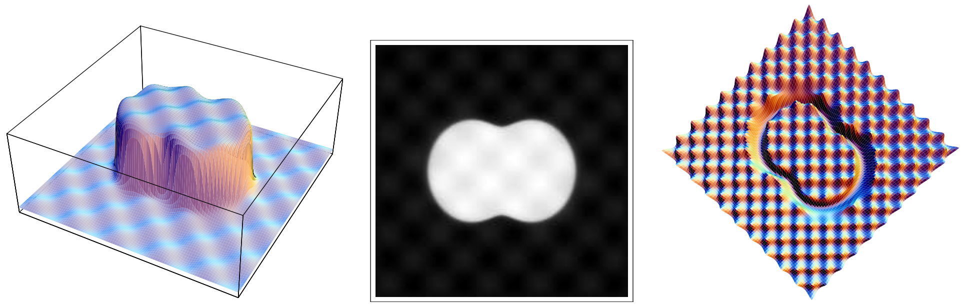


Figure 7: The situation is more complicated in case of a "noisy" image (middle); we plot also a graph of its intensity $I^0(x)$ (left) and corresponding surface $g(|\nabla I^0(x)|)$ (right). (Color Slide).

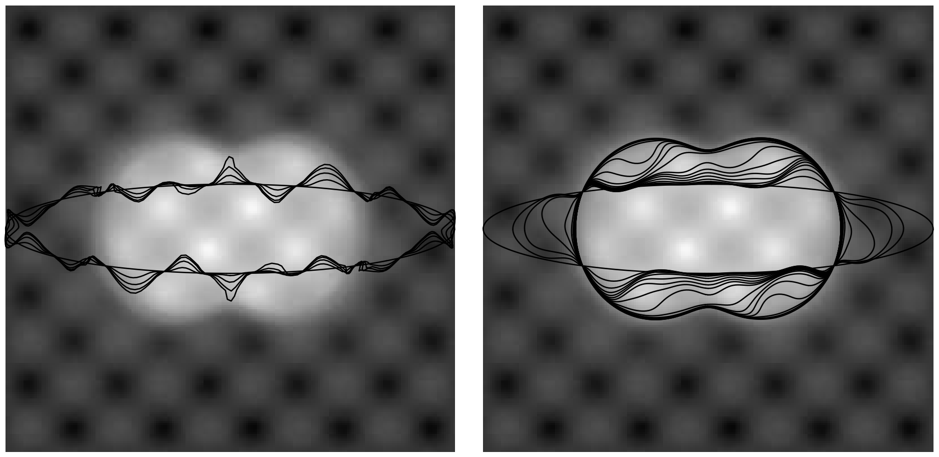


Figure 8: The evolution only by advection leads to attracting a curve (initial ellipse) to spurios edges, the evolution must be stopped without any reasonable segmentation result (left); by adding regularization term related to curvature of evolving curve the edge is found smoothly (right).

smooth edge detector function, e.g. $g(s) = 1/(1 + Ks^2)$, has been given in [6] and [36]. In such model, "steady state" of a particular level set (level line in 2D image) corresponds to boundary of a segmented object. Due to shape of the Perona-Malik function g, the moving segmentation curve is strongly slowed down in a neighbourhood of an edge leading to a segmentation result. However, if an edge is crossed during evolution (which is not a rare event in noisy images) there is no mechanism to go back. Moreover, if there is a missing part of the object boundary, the algorithm is completely unuseful (as any other simple region growing method).

Later on, the curve evolution and the level set models for segmentation have been significantly improved by finding a proper driving force in the form $-\nabla g(|\nabla I^0(x)|)$ [7, 8, 9, 30, 31]. The vector field $-\nabla g(|\nabla I^0(x)|)$ has the important geometric property: it points towards regions where the norm of the gradient ∇I^0 is large (see Figures 4 and 5). Thus if an initial curve belongs to a neighborhood of an edge, then it is driven towards this edge by this proper velocity field. Such motion can be also interpreted as a flow of the curve on surface $g(|\nabla I^0(x)|)$ subject to gravitational like force driving the curve down to the narrow valley corresponding to the edge (see Figure 6, [40]).

However, as one can see from Figures 7 and 8, the situation is much more complicated in case of noisy images. The advection process alone is not sufficient. In a noisy environment, the evolving level set can behave very irregularly, it can be attracted to spurious edges and no reasonably convergent process can be observed. This phenomenon is documented in Figure 8 left. To prevent such situation one has to regularize the evolution. A helpful regularization is to add a curvature dependence to the level set flow. If evolution of a curve in the normal direction depends on its curvature k, then the sharp irregularities are smoothed. Such motion can be interpreted as an intrinsic diffusion of the curve. A reasonable regularization term is given by $g^0 k$, where the amount of curve intrinsic diffusion is small in the vicinity of un-spurious edges. In Figure 8 right, we present initial ellipse evolution to successfull segmentation result using such advection-(intrinsic)diffusion model which was computed by the direct method from [40]. The level set formulation of such curve evolution is given by the equation (9) which is of course only another form of equation (8).

Altough model (8) behaves very well if we are in a vicinity of an edge, it is sometimes difficult to drive the segmentation curve there. If we start with a small circular seed, it has large curvature and diffusion dominates advection so the seed disappear (curve shrinks to a point [22, 23]). Then some constant speed must be added to dominate diffusion at the beginning of the process, but it is not clear at all when to switch off this driving force to have just the mechanism of the model (8). Moreover, in case of missing boundaries of image objects, there is no criterion for such a switch, so the segmentation curve cannot be well localized to complete the missing boundaries.

An important observation now is that equation (8) moves not only one particular leveline (segmentation curve) but all levelines by the above mentioned advection-diffusion mechanism. So, in spite of all previously mentioned segmentation approaches, we may start to think not on evolution of one particular level set but on evolution of the whole surface composed by those level sets. This idea to look on the solution u itself, i.e. on a behaviour of our segmentation function, can help significantly.

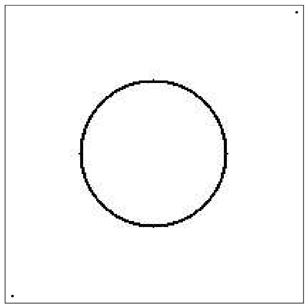


Figure 9: Image of a solid circle.

Let us look on a simple numerical experiment presented in Figure 10 representing extraction of the solid circle depicted in Figure 9. The starting point-of-view surface u^0 is plotted on the top left. The subsequent evolution is depicted in the next subfigures. First, isolines which are close to the edge, i.e. in a neighbourhood of the solid circle where the advection term is nonzero, are atracted from both sides to this edge. A small shock (steep gradient) is formed due to accumulation of these level lines (see Figure 10 top right). In the regions outside a neighbourhood of the circle the advection term is vanishing and $g^0 \equiv 1$, so only intrinsic diffusion of level sets plays a role. It means that all inside level sets are shrinking and finally they disappear. Such process is nothing else than a decreasing of the maximum of our segmentation function until the upper level of the shock is achieved. It is clear that a flat region in the profile of segmentation function inside the circle is formed. Outside of the circle, level sets are also shrinking until they are attracted by nonzero velocity field and then they contribute to the

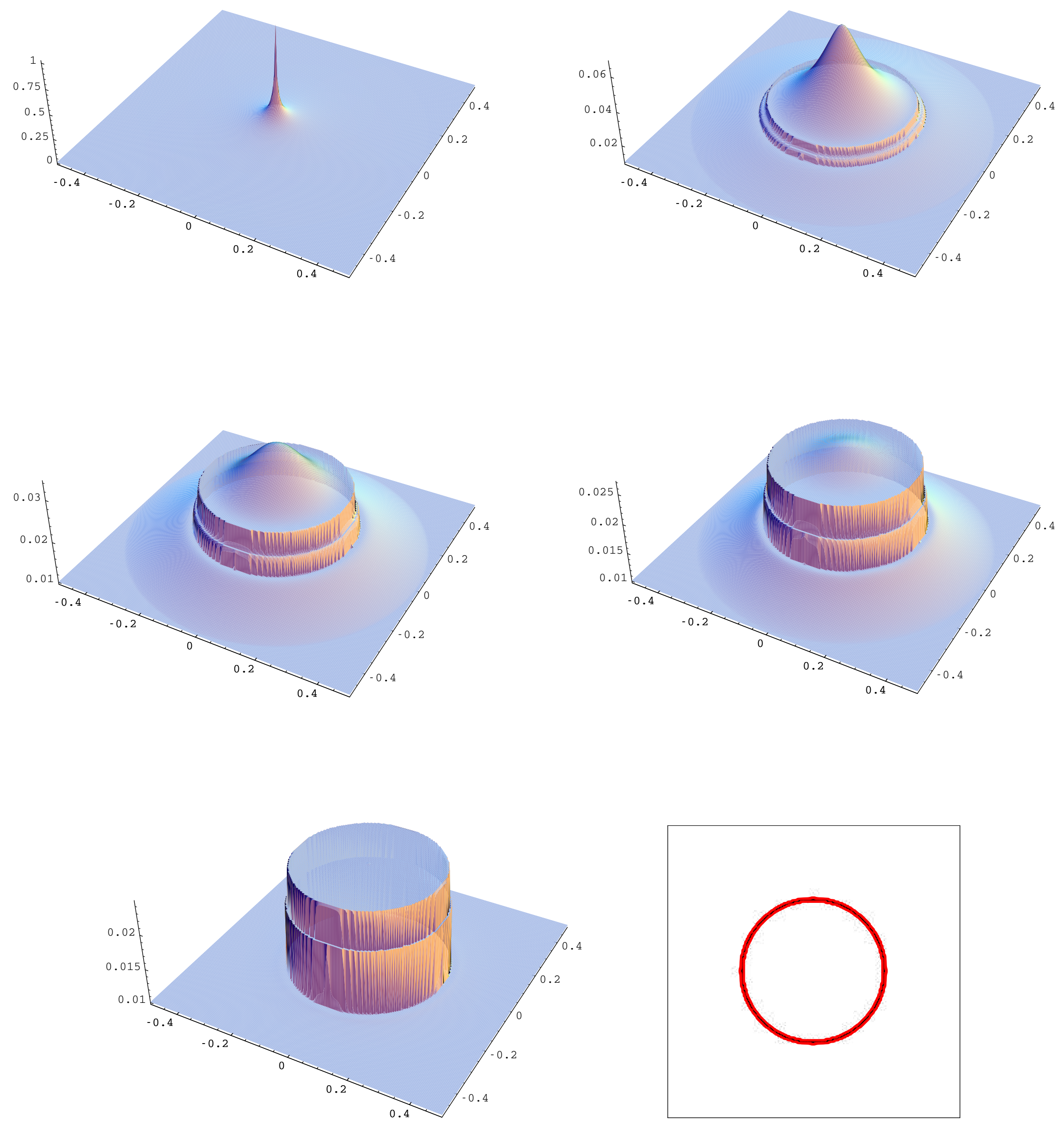


Figure 10: Subjective surface based segmentation of solid circle. We plot numerically computed time steps 0, 2, 10, 20 and 100. In the bottom right we see accumulation of level lines of segmentation function on the edges. In this experiment $\varepsilon = 10^{-10}$ so we are very close to level set flow equation (8). (Color Slide).

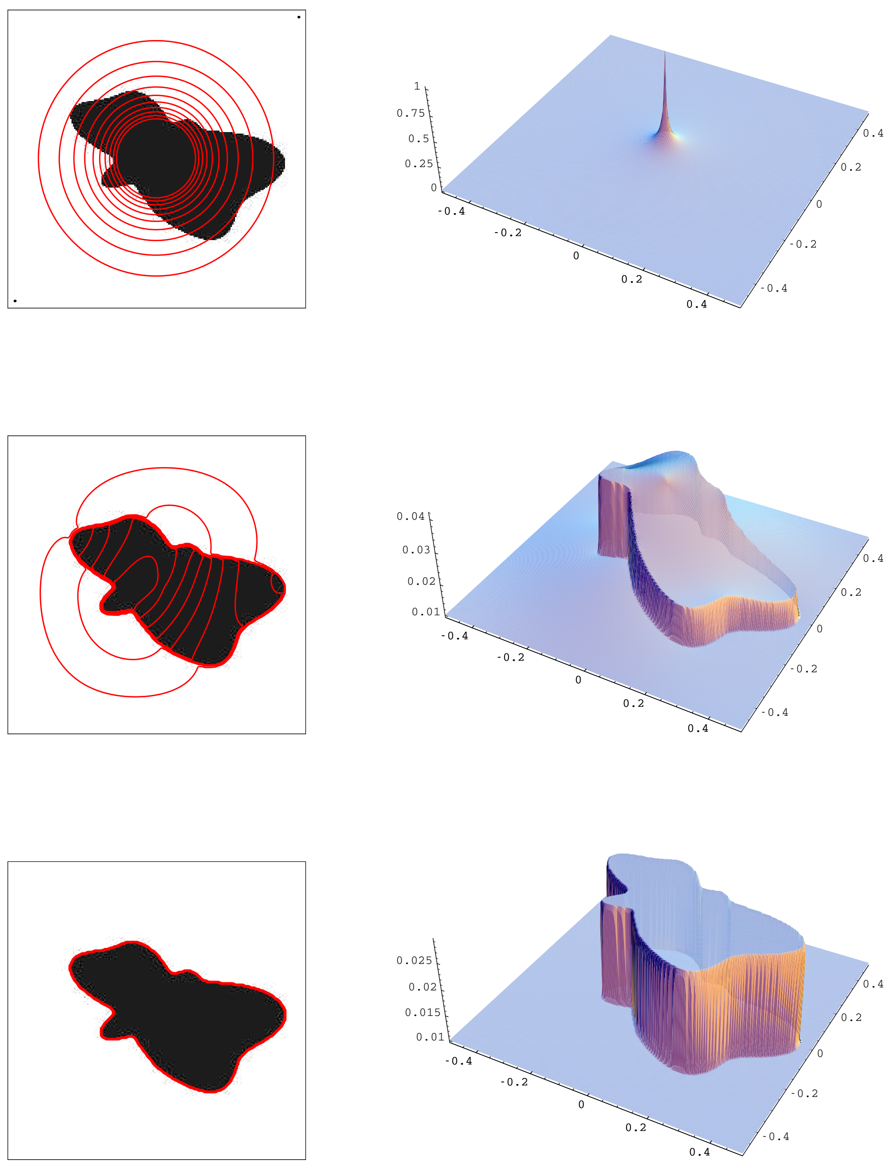


Figure 11: Subjective surface based segmentation of a "batman" image. In the left column we plot the black and white image to be segmented together with isolines of the segmentation function. In the right column there is a shape of the segmentation function. The rows correspond to time steps 0,1 and 10 which gives the final result; $\varepsilon = 1$. (Color Slide).

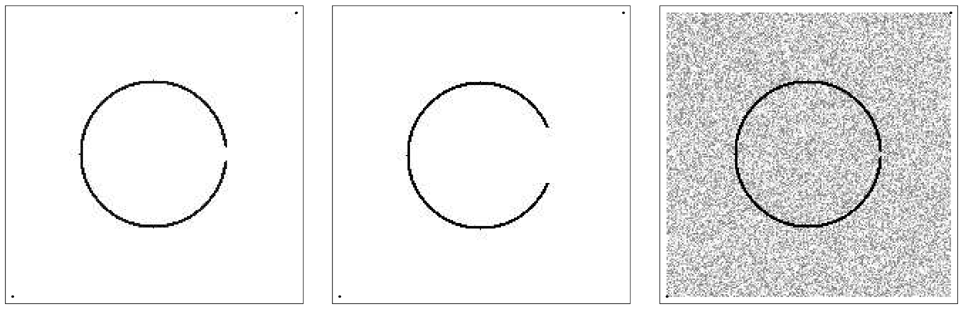


Figure 12: Three testing images. Circle with a smaller (left) and a big (middle) gap, noisy circle with a gap.

shock. In the bottom left of Figure 10 we see the shape of segmentation function u after such evolution, in the bottom right there are iso-contours of such function accumulated on the edges. It is very easy to use one of them, e.g., $(\max(u) + \min(u))/2$, to get the circle.

The situation is not so straightforward for the highly nonconvex image depicted in Figure 11. Our numerical observation leads to formation of steps in subsequent evolution of the segmentation function, which is understandable, because very different level sets of initial surface u^0 are attracted to different parts of the boundary of "batman". Fortunately, we are a bit free in choosing precise form of diffusion term in the segmentation model. After expansion of divergence, the equations (2) and (8) give the same advection term, $\nabla g^0.\nabla u$ (cf. equation (9)), so important advection mechanism which accumulates segmentation function along the shock is the same. However, diffusion mechanisms are a bit different. The equation (2), in case $\varepsilon = 1$, gives diffusion which is known as mean curvature flow of graphs. It means that, not level sets of segmentation function move in normal direction proportionally to curvature, but the graph of segmentation function moves (as 2D surface in 3D space) in the normal direction proportionally to the mean curvature. The large variations in the graph of segmentation function are then smoothed due to large mean curvature. Of course, the smoothing is applied only outside edges. On edges the advection dominates, since the mean curvature term is multiplied by a small value of g^0. In Figure 11 bottom we may see formation of a piecewise flat profile of the segmentation function which can be again very simply used for extraction of "batman", altough, due to Dirichlet boundary data and $\varepsilon = 1$, this profile moves slowly downwards in subsequent evolution. In this (academic) example, the only goal was to smooth (flatten) the segmentation function inside and outside the edge, so the choice $\varepsilon = 1$ was really satisfactory. In case $\varepsilon = 1$, the equation (2) can be interpreted as a time relaxation for the minimazion of the weighted area functional

$$A_{g^0} = \int_\Omega g^0 \sqrt{1 + |\nabla u|^2} dx,$$

or as the mean curvature motion of a graph in Riemann space with metric $g^0 \delta_{ij}$ [48].

In the next three testing images plotted in Figure 12 we illustrate the role of the regularization parameter ε. The same choice, $\varepsilon = 1$, as in the previous image with complete edge, is clearly not appropriate for image object with a gap (Figure 12 left), as seen in Figure 13. We see that minimal surface like diffusion closes the gap with a smoothly varying "waterfall" like shape. Altough this shape is in a sense stable (it moves downwards in a "selfsimilar form"), it is not appropriate for segmentation purposes. However, decreasing ε, i.e., if we stay closer to the curvature driven level set flow (8), or in other words, if we stretch the Riemannian metric $g^0 \delta_{ij}$ in vertical z direction [49], we get very good segmentation results as presented in Figure 14. Of course, smaller ε is needed to close larger gaps (see Figure 15).

If there is a noisy image as in Figure 12 right, the motion of level lines to shock is more irregular, but finally the segmentation function is smoothed as well (see Figures 16-17). If the regularization parameter ε is small, then piecewise flat profile of the segmentation function is moving very slowly downwards, so it is easy to stop the evolution and get the result of segmentation process.

By the presented experiments we have seen that solution of equation (2) is well suited for finding and completing edges in (noisy) images. Its advection-diffusion mechanism leads to promissing results. In the next section we give efficient and robust computational method for its solution.

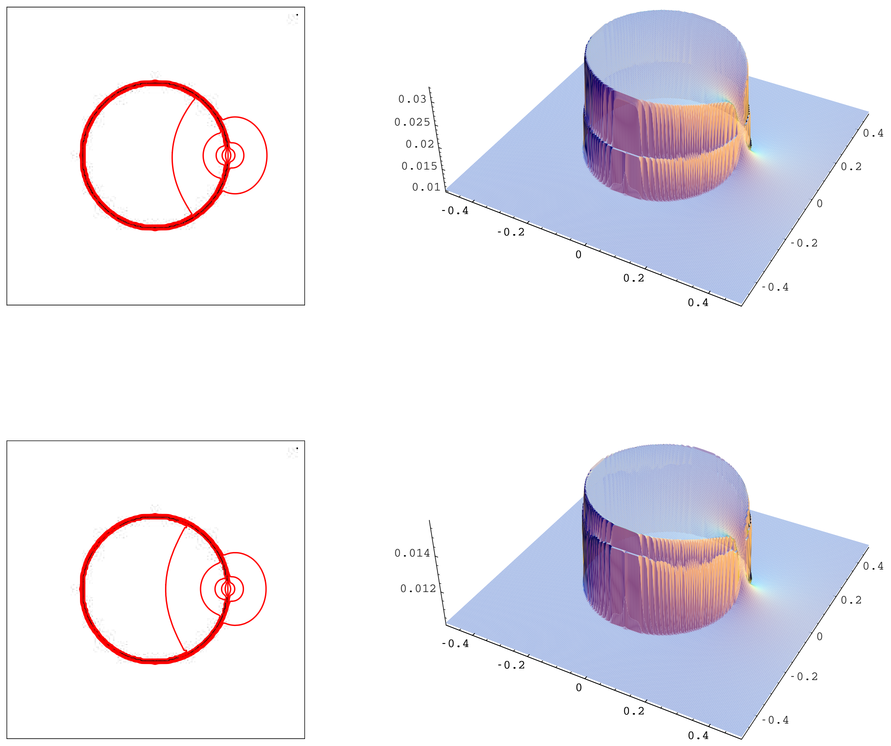


Figure 13: Experiment on testing image plotted in Figure 12 left. The results of evolution of the segmentation function (in the left isolines, in the right graph) after 10 (top row) and 100 (bottom row) time steps; $\varepsilon = 1$. The shape is stable (in a "selfsimilar form" moving downwards) but not well suited as segmentation result. (Color Slide).

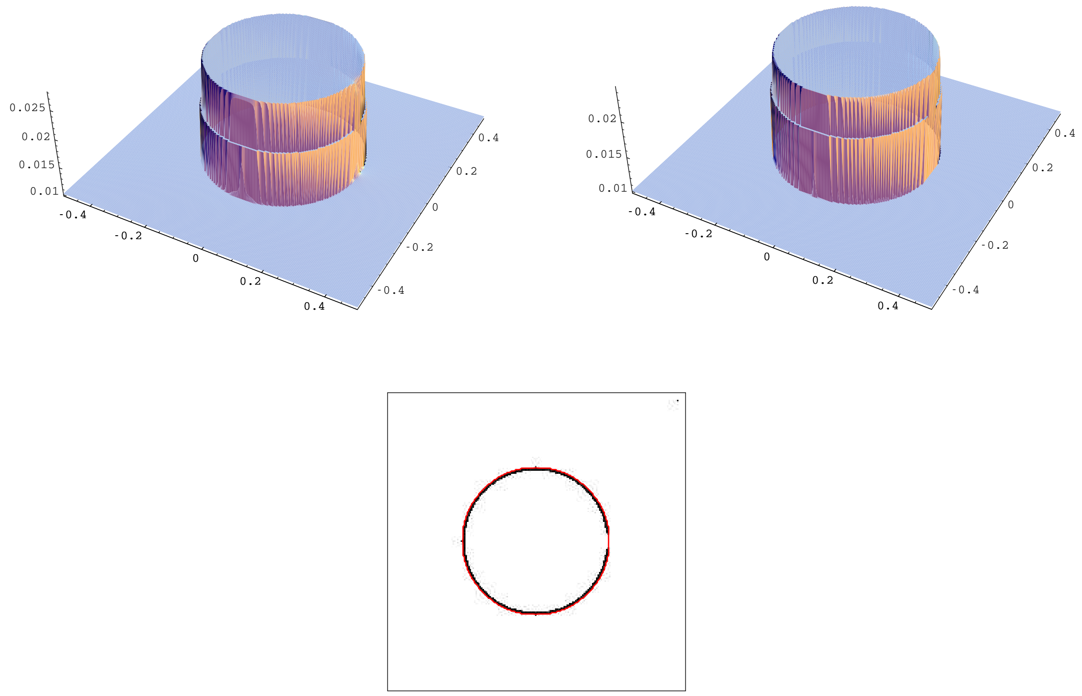


Figure 14: Results of the segmentation process for testing image plotted in Figure 12 left using $\varepsilon = 10^{-2}$ (top left) and $\varepsilon = 10^{-5}$ (top right). The isoline $(\max(u) + \min(u))/2$ well represents the segmented circle (bottom red line). For large range of ε we get satisfactory results. (Color Slide).

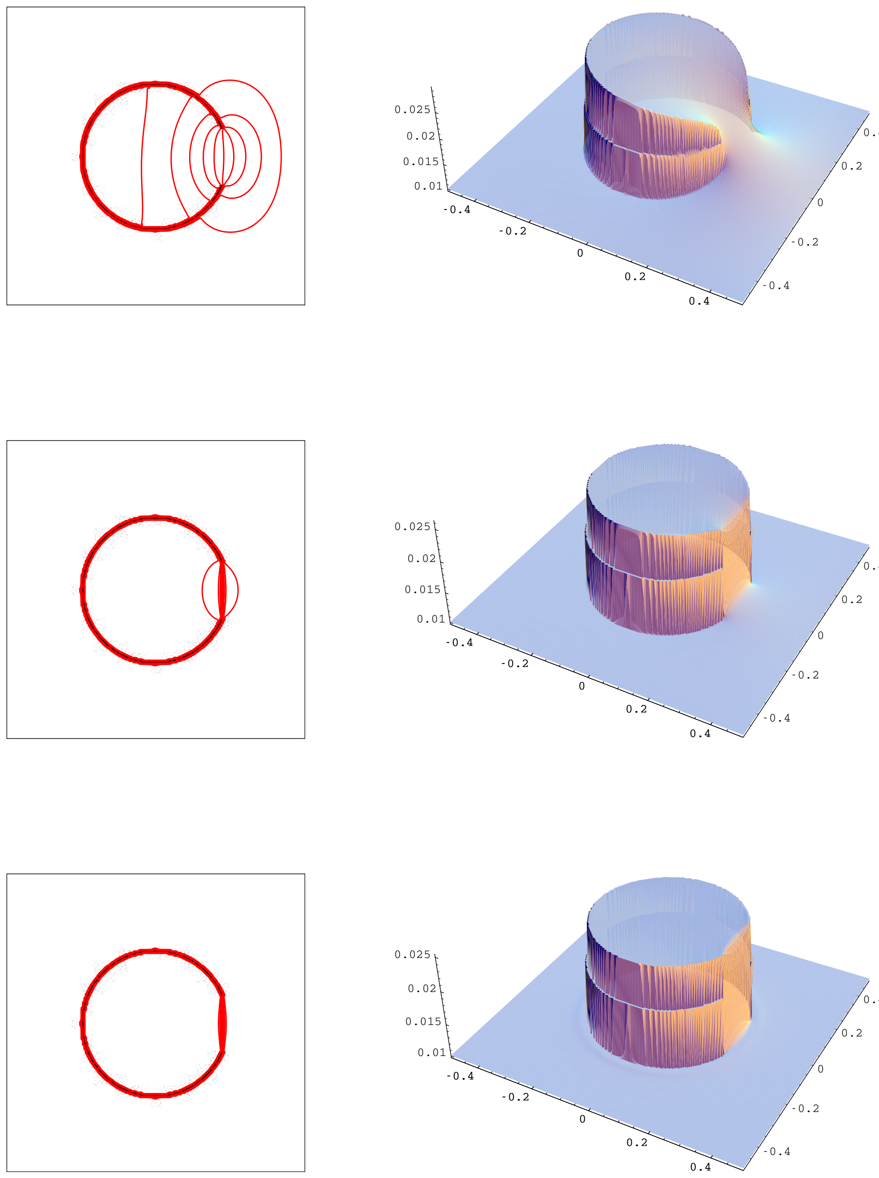


Figure 15: Segmentation of the circle with a big gap (Figure 12 middle) using $\varepsilon = 1$ (top), $\varepsilon = 10^{-2}$ (middle) and $\varepsilon = 10^{-5}$ (bottom). For bigger missing part a smaller ε is desirable. In the left collumn we see how closely to edges the isolines are accumulating and closing the gap, in the right we see how steep the segmentation function is along the gap. (Color Slide).

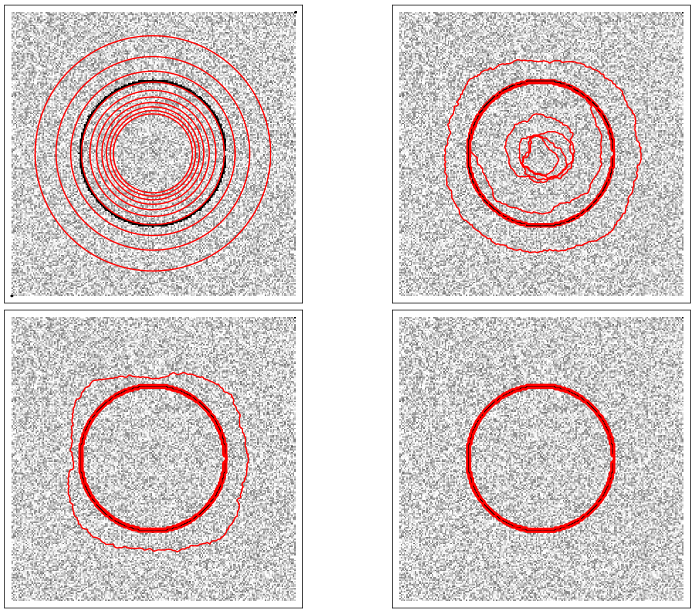


Figure 16: Isolines of the segmentation function in the segmentation of the noisy circle (Figure 12 right) are shown in time steps 0, 50, 100 and 200. Since the gap is not so big we have chosen $\varepsilon = 10^{-1}$. (Color Slide).

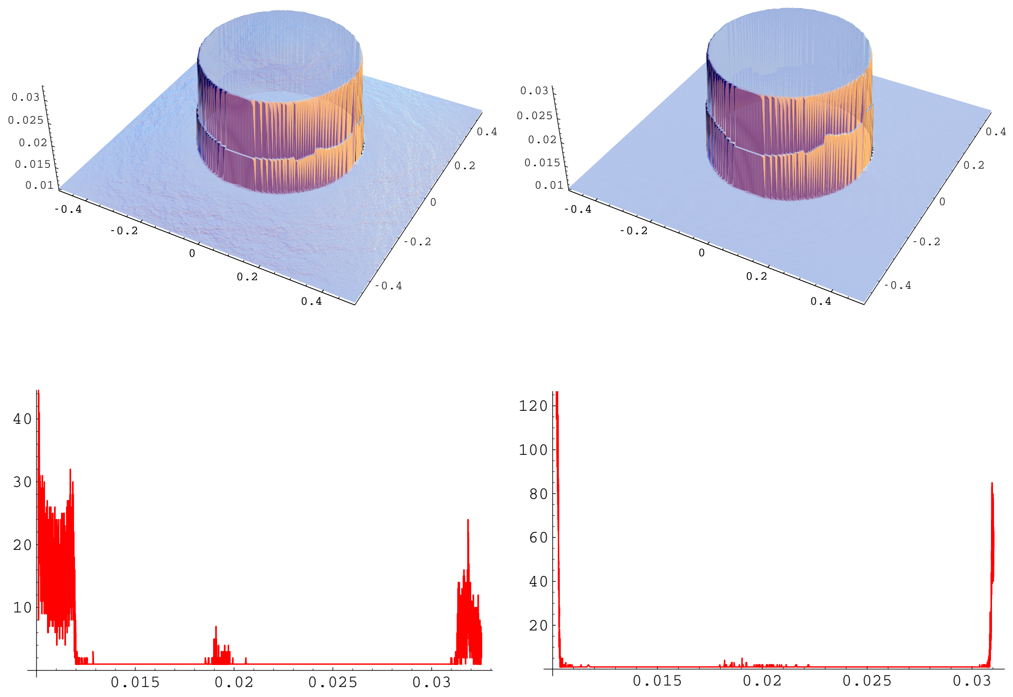


Figure 17: The graph of the segmentation function and its histograms in time steps 100 and 200 for the same experiment as presented in Figure 16. The histograms give a practical advise to shorten the segmentation process in case of noisy images. For a noisy image a formation of completely piecewise flat subjective surface takes longer time. However, the gaps in histogram of the segmentation function are developed soon. It allows to take any level inside these gaps and to visualize corresponding level line to get desirable segmentation result. (Color Slide).

3 Semi-implicit Co-volume Scheme

We present our method in discretization of equation (8), although we always use its ε-regularization (2) with a specific $\varepsilon > 0$. The notation is simpler in case of (8) and it will be clear where regularization appears in the numerical scheme.

First we choose a uniform discrete time step τ and a variance σ of the smoothing kernel G_σ. Then we replace time derivative in (8) by backward difference. The nonlinear terms of the equation are treated from the previous time step while the linear ones are considered on the current time level, this means semi-implicitness of the time discretization. In the last decade, semi-implicit schemes have become a powerful tool in image processing, we refer e.g. to [27, 3, 4, 58, 57, 37, 33, 25, 26, 51].

Semi-implicit in time discretization: *Let τ and σ be fixed numbers, I^0 be a given image and u^0 be a given initial segmentation function. Then, for $n = 1, \dots N$, we look for a function u^n, solution of the equation*

$$\frac{1}{|\nabla u^{n-1}|}\frac{u^n - u^{n-1}}{\tau} = \nabla . \left(g^0 \, \frac{\nabla u^n}{|\nabla u^{n-1}|} \right). \tag{11}$$

A digital image is given on a structure of pixels with rectangular shape, in general (red rectangles in Figure 18). Since discrete values of I^0 are given in pixels and they influence the model, we will relate spatially discrete approximations of the segmentation function u also to image pixels, more precisely, to their centers (red points in Figure 18). In every discrete time step of the method (11) we have to evaluate gradient of the segmentation function at the previous step $|\nabla u^{n-1}|$. For that goal, it is reasonable to put a triangulation (dashed black lines in Figure 18) inside the pixel structure and take a piecewise linear approximation of the segmentation function on this triangulation. Such approach will give a constant value of gradient per triangles allowing simple and clear construction of fully-discrete system of equations. It is a main feature of the co-volume [56, 25] and finite element [13, 14, 15] methods in solving mean curvature flow in the level set formulation.

As it can be seen in Figure 18, in our method the centers of pixels are connected by a new rectangular mesh and every new rectangle is splitted into four triangles. The centers of pixels will be called degree of freedom (DF) nodes. By this procedure we get also further nodes (at crossing of red lines in Figure 18) which, however, will not represent degrees of freedom. We

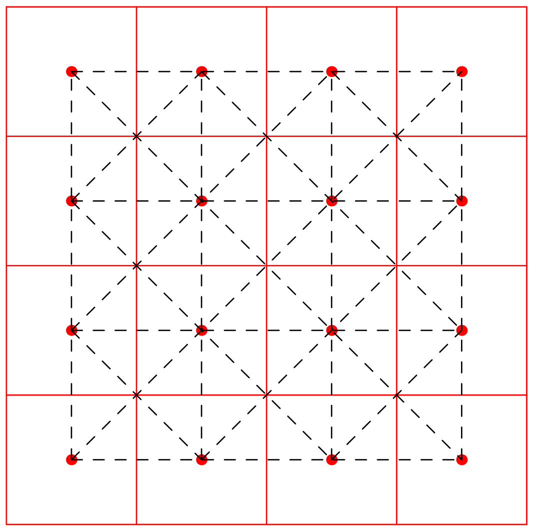


Figure 18: The image pixels (red solid lines) corresponding to co-volume mesh. Triangulation (black dashed lines) for the co-volume method with degree of freedom nodes (red round points) corresponding to centers of pixels. (Color Slide).

will call them non-degree of freedom (NDF) nodes. Let a function u be given by discrete values in the pixel centers, i.e. in DF nodes. Then in additional NDF nodes we take the average value of the neighbouring DF nodal values. By such defined values in NDF nodes a piecewise linear approximation u_h of u on the triangulation can be built. Let us note that we restrict further considerations in this chapter only to this type of grids. For triangulation $\mathcal{T}_h$, given by the previous construction, we construct a co-volume (dual) mesh. We modify a basic approach given in [56, 25] in such a way that our co-volume mesh will consist of cells p associated only with DF nodes p of $\mathcal{T}_h$, let say $p = 1, \dots, M$. Since there will be one-to-one correspondence between co-volumes and DF nodes, without any confusion, we use the same notation for them. In this way we have excluded the boundary nodes (due to Dirichlet boundary data) and NDF nodes.

For each DF node p of $\mathcal{T}_h$ let C_p denote the set of all DF nodes q connected to the node p by an edge. This edge will be denoted by σ_{pq} and its length by h_{pq}. Then every *co-volume* p is bounded by the lines (co-edges) e_{pq} that bisect and are perpendicular to the edges $\sigma_{pq}, q \in C_p$. By this construction, the co-volume mesh corresponds exactly to the pixel structure of the image inside the computational domain Ω where the segmentation is provided. We denote by $\mathcal{E}_{pq}$ the set of triangles having σ_{pq} as an edge. In situation depicted in Figure 18 every $\mathcal{E}_{pq}$ cosists of two triangles. For each $T \in \mathcal{E}_{pq}$ let c_{pq}^T be the length of the portion of e_{pq} that is in T, i.e., $c_{pq}^T = m(e_{pq} \cap T)$, where m is measure in $I\!R^{d-1}$. Let $\mathcal{N}_p$ be the set of triangles that have DF node p as a vertex. Let u_h be a piecewise linear function on triangulation $\mathcal{T}_h$. We will denote a constant value of $|\nabla u_h|$ on $T \in \mathcal{T}_h$ by $|\nabla u_T|$ and define regularized gradients by

$$|\nabla u_T|_\varepsilon = \sqrt{\varepsilon^2 + |\nabla u_T|^2}. \tag{12}$$

We will use notation $u_p = u_h(x_p)$ where x_p is the coordinate of the node p of triangulation $\mathcal{T}_h$.

With these notations, we are ready to derive co-volume spatial discretization. As it is usual in finite volume methods [34, 20, 44], we integrate (11) over every co-volume p, $i = 1, \ldots, M$. We get

$$\int_p \frac{1}{|\nabla u^{n-1}|} \frac{u^n - u^{n-1}}{\tau} dx = \int_p \nabla . \left(g^0 \frac{\nabla u^n}{|\nabla u^{n-1}|} \right) dx. \tag{13}$$

For the right hand side of (13) using divergence theorem we get

$$\begin{aligned} \int_p \nabla . \left(g^0 \frac{\nabla u^n}{|\nabla u^{n-1}|} \right) dx &= \int_{\partial p} \frac{g^0}{|\nabla u^{n-1}|} \frac{\partial u^n}{\partial \nu} ds \\ &= \sum_{q \in C_p} \int_{e_{pq}} \frac{g^0}{|\nabla u^{n-1}|} \frac{\partial u^n}{\partial \nu} ds. \end{aligned}$$

So we have an integral formulation of (11)

$$\int_p \frac{1}{|\nabla u^{n-1}|} \frac{u^n - u^{n-1}}{\tau} dx = \sum_{q \in C_p} \int_{e_{pq}} \frac{g^0}{|\nabla u^{n-1}|} \frac{\partial u^n}{\partial \nu} ds \tag{14}$$

expressing a "local mass balance" property of the scheme. Now the exact "fluxes" on the right hand side and "capacity function" $\frac{1}{|\nabla u^{n-1}|}$ in the left hand side (see e.g. [34]) will be approximated numerically using piecewise linear reconstruction of u^{n-1} on trangulation $\mathcal{T}_h$. If we denote g_T^0 approximation of g^0 on a triangle $T \in \mathcal{T}_h$, then for the approximation of the right hand side of (14) we get

$$\sum_{q \in C_p} \left(\sum_{T \in \mathcal{E}_{pq}} c_{pq}^T \frac{g_T^0}{|\nabla u_T^{n-1}|} \right) \frac{u_q^n - u_p^n}{h_{pq}}, \tag{15}$$

and the left hand side of (14) is approximated by

$$M_p m(p) \frac{u_p^n - u_p^{n-1}}{\tau} \tag{16}$$

where $m(p)$ is a measure in $I\!R^d$ of co-volume p and either

$$M_p = \frac{1}{|\nabla u_p^{n-1}|}, \quad |\nabla u_p^{n-1}| = \sum_{T \in \mathcal{N}_p} \frac{m(T \cap p)}{m(p)} |\nabla u_T^{n-1}| \tag{17}$$

or

$$M_p = \sum_{T \in \mathcal{N}_p} \frac{m(T \cap p)}{m(p)} \frac{1}{|\nabla u_T^{n-1}|}. \tag{18}$$

The averaging of the gradients (17) has been used in [56, 25], the approximation (18) is new and we have found it very useful regarding good convergence properties in solving the linear systems (see below) iteratively for $\varepsilon << 1$. Regularizations of both the approximations of the capacity function are as follows, either

$$M_p^\varepsilon = \frac{1}{|\nabla u_p^{n-1}|_\varepsilon} \tag{19}$$

or

$$M_p^\varepsilon = \sum_{T \in \mathcal{N}_p} \frac{m(T \cap p)}{m(p)} \frac{1}{|\nabla u_T^{n-1}|_\varepsilon}. \tag{20}$$

Now we can define coefficients, where the ε-regularization is taken into account, namely

$$b_p^{n-1} = M_p^\varepsilon m(p) \tag{21}$$

$$a_{pq}^{n-1} = \frac{1}{h_{pq}} \sum_{T \in \mathcal{E}_{pq}} c_{pq}^T \frac{g_T^0}{|\nabla u_T^{n-1}|_\varepsilon} \tag{22}$$

which together with (15)-(16) give

Fully-discrete semi-implicit co-volume scheme: *Let $u_p^0, p = 1, \ldots, M$ be given discrete initial values of the segmentation function. Then, for $n = 1, \ldots, N$ we look for $u_p^n, p = 1, \ldots, M$, satisfying*

$$b_p^{n-1}\, u_p^n + \tau \sum_{q \in C_p} a_{pq}^{n-1} (u_p^n - u_q^n) = b_p^{n-1}\, u_p^{n-1}. \tag{23}$$

Theorem. *There exists unique solution* $(u_1^n, \dots, u_M^n)$ *of the scheme* (23) *for any* $\tau > 0$, $\varepsilon > 0$ *and for every* $n = 1, \dots, N$. *Moreover, for any* $\tau > 0$, $\varepsilon > 0$ *the following stability estimate holds*

$$\min_p \ u_p^0 \le \min_p \ u_p^n \le \max_p \ u_p^n \le \max_p \ u_p^0, \quad 1 \le n \le N. \tag{24}$$

Proof. The system (23) can be rewritten into the form

$$\left(b_p^{n-1} + \tau \sum_{q \in C_p} a_{pq}^{n-1} \right) u_p^n - \tau \sum_{q \in C_p} a_{pq}^{n-1} u_q^n = b_p^{n-1} \ u_p^{n-1}. \tag{25}$$

Applying Dirichlet boundary conditions, it gives the system of linear equations with a matrix, off diagonal elements of which are symmetric and negative. Diagonal elements are positive and dominate the sum of absolute values of the nondiagonal elements in every row. Thus, the matrix of the system is symmetric and diagonally dominant M-matrix which imply that it always has unique solution. The M-matrix property gives us the minimum-maximum principle, which can be seen by the following simple trick. We may temporary rewrite (23) into the equivalent form

$$u_p^n + \frac{\tau}{b_p^{n-1}} \sum_{q \in C_p} a_{pq}^{n-1} (u_p^n - u_q^n) = u_p^{n-1} \tag{26}$$

and let $\max(u_1^n, \dots, u_M^n)$ be achieved in the node p. Then the whole second term on the left hand side is nonnegative and thus $\max(u_1^n, \dots, u_M^n) = u_p^n \le u_p^{n-1} \le \max(u_1^{n-1}, \dots, u_M^{n-1})$. In the same way we can prove the relation for minimum and together we have

$$\min_p \ u_p^{n-1} \le \min_p \ u_p^n \le \max_p \ u_p^n \le \max_p \ u_p^{n-1}, \quad 1 \le n \le N \tag{27}$$

which by recursion imply the desired stability estimate (24).

So far, we have said nothing about evaluation of g_T^0 included in coefficients (22). Since image is piecewise constant on pixels we may replace the convolution by the weighted average to get $I_\sigma^0 := G_\sigma * I^0$ (see e.g. [37]) and then relate discrete values of I_σ^0 to pixel centers. Then, as above, we may construct its piecewise linear representation on triangulation and in such way we get constant value of ∇I_σ^0 on every triangle $T \in \mathcal{T}_h$. Another possibility is to solve numerically a linear heat equation for time t corresponding to variance σ with initial datum given by I^0 (see e.g. [3]). The convolution represents a preliminary smoothing of the data. It is also a theoretical tool to have bounded gradients and thus a strictly positive weighting coefficient g^0. In practice, the evaluation of gradients on discrete grid (e.g., on triagulation described above) gives always bounded values. So, working on discrete grid, one can also avoid the convolution, especially if preliminary denoising is not needed or not desirable. Then it is possible to work directly with gradients of piecewise linear representation of I^0 in evaluation of g^0_T.

Our co-volume scheme in this paper is designed for the specific mesh (see Figure 18) given by the rectangular pixel structure of 2D image. For simplicity of implementation and for reader convenience we will write the co-volume scheme in a "finite-difference notation". As it is usual for 2D rectangular grids, we associate co-volume p and its corresponding center (DF node) with a couple (i,j), i will represent vertical direction, j horizontal direction. If Ω is a regtangular subdomain of the image domain where n_1 and n_2 are number of pixels in vertical and horizontal directions, respectively, then $i = 1, \dots, m_1$, $j = 1, \dots, m_2$, $m_1 <= n_1 - 2$, $m_2 <= n_2 - 2$ and $M = m_1\ m_2$. Similarly, the unknown value u_p^n is associated with $u_{i,j}^n$. For every co-volume p, the set $\mathcal{N}_p$ consists of 8 triangles (see Figure 18). In every discrete time step $n = 1, \dots, N$, and for every $i = 1, \dots, m_1$, $j = 1, \dots, m_2$, we compute absolute value of gradient on these 8 triangles denoted by $G_{i,j}^k$, $k = 1, \dots, 8$. For that goal, using discrete values of u from the previous time step, we use the following expressions (we omit upper index $n-1$ on u)

$$G_{i,j}^1 = \sqrt{\left(\frac{0.5\ (u_{i,j+1}+u_{i+1,j+1}-u_{i,j}-u_{i+1,j})}{h}\right)^2 + \left(\frac{u_{i+1,j}-u_{i,j}}{h}\right)^2}$$

$$G_{i,j}^2 = \sqrt{\left(\frac{0.5\ (u_{i,j}+u_{i+1,j}-u_{i,j-1}-u_{i+1,j-1})}{h}\right)^2 + \left(\frac{u_{i+1,j}-u_{i,j}}{h}\right)^2}$$

$$G_{i,j}^3 = \sqrt{\left(\frac{0.5\ (u_{i+1,j-1}+u_{i+1,j}-u_{i,j-1}-u_{i,j})}{h}\right)^2 + \left(\frac{u_{i,j}-u_{i,j-1}}{h}\right)^2}$$

$$G_{i,j}^4 = \sqrt{\left(\frac{0.5\ (u_{i,j-1}+u_{i,j}-u_{i-1,j-1}-u_{i-1,j})}{h}\right)^2 + \left(\frac{u_{i,j}-u_{i,j-1}}{h}\right)^2}$$

$$G_{i,j}^5 = \sqrt{\left(\frac{0.5\ (u_{i,j}+u_{i-1,j}-u_{i,j-1}-u_{i-1,j-1})}{h}\right)^2 + \left(\frac{u_{i,j}-u_{i-1,j}}{h}\right)^2}$$

$$G^6_{i,j} = \sqrt{\left(\frac{0.5\ (u_{i,j+1}+u_{i-1,j+1}-u_{i,j}-u_{i-1,j})}{h}\right)^2 + \left(\frac{u_{i,j}-u_{i-1,j}}{h}\right)^2}$$

$$G^7_{i,j} = \sqrt{\left(\frac{0.5\ (u_{i,j}+u_{i,j+1}-u_{i-1,j}-u_{i-1,j+1})}{h}\right)^2 + \left(\frac{u_{i,j+1}-u_{i,j}}{h}\right)^2}$$

$$G^8_{i,j} = \sqrt{\left(\frac{0.5\ (u_{i+1,j}+u_{i+1,j+1}-u_{i,j}-u_{i,j+1})}{h}\right)^2 + \left(\frac{u_{i,j+1}-u_{i,j}}{h}\right)^2}$$

In the same way, but only in the beginning of the algorithm, we compute values $G^{\sigma,k}_{i,j}, k = 1,\dots,8$, changing u by I^0_σ in the previous expressions, where I^0_σ is a smoothed image as explained in the paragraph above. Then for every $i = 1,\dots,m_1,\ j = 1,\dots,m_2$ we construct (north, west, south, east) coefficients

$$n_{i,j} = \tau\frac{1}{2}\sum_{k=1}^{2}\frac{g(G^{\sigma,k}_{i,j})}{\sqrt{\varepsilon^2+(G^k_{i,j})^2}},\quad w_{i,j} = \tau\frac{1}{2}\sum_{k=3}^{4}\frac{g(G^{\sigma,k}_{i,j})}{\sqrt{\varepsilon^2+(G^k_{i,j})^2}}$$

$$s_{i,j} = \tau\frac{1}{2}\sum_{k=5}^{6}\frac{g(G^{\sigma,k}_{i,j})}{\sqrt{\varepsilon^2+(G^k_{i,j})^2}},\quad e_{i,j} = \tau\frac{1}{2}\sum_{k=7}^{8}\frac{g(G^{\sigma,k}_{i,j})}{\sqrt{\varepsilon^2+(G^k_{i,j})^2}}$$

and we use either (cf. (17))

$$m_{i,j} = \frac{1}{\sqrt{\varepsilon^2 + \left(\frac{1}{8}\sum\limits_{k=1}^{8} G^k_{i,j}\right)^2}}$$

or (cf. (18))

$$m_{i,j} = \frac{1}{8}\sum_{k=1}^{8}\frac{1}{\sqrt{\varepsilon^2+(G^k_{i,j})^2}}$$

to define diagonal coefficients

$$c_{i,j} = n_{i,j} + w_{i,j} + s_{i,j} + e_{i,j} + m_{i,j}h^2.$$

If we define right hand sides at the nth discrete time step by

$$r_{i,j} = m_{i,j}h^2u^{n-1}_{i,j},$$

then for DF node corresponding to couple (i,j) we get equation

$$c_{i,j}u^n_{i,j} - n_{i,j}u^n_{i+1,j} - w_{i,j}u^n_{i,j-1} - s_{i,j}u^n_{i-1,j} - e_{i,j}u^n_{i,j+1} = r_{i,j}. \tag{28}$$

Collecting these equations for all DF nodes and taking into account Dirichlet boundary conditions we get linear system to be solved.

We solve this system by the so-called SOR (Successive Over Relaxation) iterative method which is a modification of the basic Gauss-Seidel algorithm (see e.g. [46]). At the nth discrete time step we start the iterations by setting $u_{i,j}^{n(0)} = u_{i,j}^{n-1}$, $i = 1, \ldots, m_1$, $j = 1, \ldots, m_2$. Then in every iteration $l = 1, \ldots$ and for every $i = 1, \ldots, m_1$, $j = 1, \ldots, m_2$, we use the following two step procedure:

$$\begin{aligned} Y &= (s_{i,j} u_{i-1,j}^{n(l)} + w_{i,j} u_{i,j-1}^{n(l)} + e_{i,j} u_{i,j+1}^{n(l-1)} + n_{i,j} u_{i+1,j}^{n(l-1)} + r_{i,j})/c_{i,j} \\ u_{i,j}^{n(l)} &= u_{i,j}^{n(l-1)} + \omega(Y - u_{i,j}^{n(l-1)}). \end{aligned}$$

We define squared L_2 norm of residuum at current iteration by

$$R^{(l)} = \sum_{i,j} (c_{i,j} u_{i,j}^{n(l)} - n_{i,j} u_{i+1,j}^{n(l)} - w_{i,j} u_{i,j-1}^{n(l)} - s_{i,j} u_{i-1,j}^{n(l)} - e_{i,j} u_{i,j+1}^{n(l)} - r_{i,j})^2.$$

The iterative process is stopped if $R^{(l)} < \text{TOL}\ R^{(0)}$. Since the computing of residuum is time consuming itself, we check it, e.g., after every 10 iterations. The relaxation parameter ω is chosen by user to improve convergence rate of the method; we have very good experience with $\omega = 1.85$ for these type of problems. Of course the number of iterations depends on the chosen precision TOL, length of time step τ and a value of the regularization parameter ε also plays a role. If one wants to weaken this dependence, more sofisticated approaches can be recommended (see e.g. [46, 35, 25] and paragraph below) but their implementation needs more programming effort. The semi-implicit co-volume method as presented above can be implemented in tens of lines.

We outline shortly also further approaches for solving the linear systems given in every discrete time step by (23). The system matrix has known (penta-diagonal) structure and moreover it is symmetric and diagonally dominant M-matrix. One could apply direct methods as Gaussian elimination, but this approach would lead to an immense storage requirements and computational effort. On the contrary, iterative methods can be applied in a very efficient way. In the previous paragraph we have already presented one of the most popular iterative methods, namely SOR. This method does not need additional storage, the matrix elements are used only to multiply the old solution values and convergence can be guaranteed for our special structure and properties of the system matrix . However, if the convergence is slow due to condition number of the system matrix (which increases with number of unknowns and for increasing τ and decreasing ε), faster iterative methods can be used. E.g., the preconditioned conjugate gradient methods allow fast convergence, altough they need more storage. If the storage requirements are reduced, then they can be very efficient and robust [35, 25]. For details of implementation of the efficient preconditioned iterative solvers for co-volume level set method we refer to [25], cf. also [51]. Also an alternative direct approach based on operating splitting schemes can be recommended [58, 57].

In the next Section, comparing CPU times, we will show that semi-implicit scheme is much more efficient and robust than explicit scheme for these type of problems. The explicit scheme combined with finite differences in space is usually based on formulations like (9) [7, 8, 9, 30, 31, 48, 49, 50] where all derivatives are expanded to get curvature and advection terms. Then, e.g., equation (2) for $\varepsilon = 1$ is written in the form

$$u_t = g^0 \frac{(1+u_{x_2}^2)u_{x_1x_1} - 2u_{x_1}u_{x_2}u_{x_1x_2} + (1+u_{x_1}^2)u_{x_2x_2}}{1+u_{x_1}^2+u_{x_2}^2} + g_{x_1}^0 u_{x_1} + g_{x_2}^0 u_{x_2}$$

where u_s means partial derivative of a function u with respect to a variable s and x_1 and x_2 are spatial coordinates in the plane. In this form, it is not clear (reader may try) which terms to take from previous and which on the current time level, having in mind the unconditional stability of the method. Fully implicit time stepping would lead to a difficult nonlinear system solution, so the explicit approach is the one straightforwardly utilizable. In spite of that, the basic formulation (2) leads naturally to convenient semi-implicit time discretization.

Let us recall the usual criterion on numerical schemes for solving partial differential equations: numerical domain of dependence should contain physical domain of dependence. In diffusion processes, in spite of advection, a value of solution at any point is influenced by any other value of solution in a computational domain. This is naturally fulfilled by the semi-implicit scheme. We solve linear system of equations at every time step which, at every discrete point, takes into account contribution of all other discrete values in computational domain.

4 Discussion On Numerical Results

This Section is devoted to discussion on further numerical experiments computed by the semi-implicit co-volume level set method. In Section 2 we already discussed some examples which have been used mainly to illustrate the advection-diffusion mechanism of the segmentation equation (2) and the role of parameter ε in closing the gaps. In the sequel we will discuss a role of further model parameters as well as all aspects of our implementation. We also compare the method with different approaches to confirm efficiency of our numerical scheme.

For a given discrete image I^0 with n_1, n_2, the number of pixels in vertical and horizontal directions, respectively, we define space discretization step $h = \frac{1}{n_1}$. It means, we embed the image into a rectangle $[-0.5\frac{n_2}{n_1}, 0.5\frac{n_2}{n_1}] \times [-0.5, 0.5]$. If one wants to use $h = 1$ (which would correspond to pixel size equals to 1) all considerations can be changed accordingly. We prefer the above definition of spatial discretization step, because it is closer to standard approaches to numerical solution of PDEs.

First we give some CPU times overview of the method. Since we are interested to find a "steady state" (see discussion in Section 2) of the evolution in order to stop the segmentation process, the important property is a number of time steps needed to come to this "equilibrium" and a CPU time for every discrete time step. We discuss CPU times in experiment related to segmentation of the circle with a gap given in Figure 12 left, computed using $\varepsilon = 10^{-2}$ (see Figure 14 top left). The testing image has 200×200 pixels and the computational domain Ω coresponds to the whole image domain. Since for the boundary nodes we prescribe Dirichlet boundary conditions, we have $M = 198 \times 198$ degrees of freedom. As the criterion to recognize the "steady state" we use a change in L_2 norm of solution between subsequent time steps, i.e., we check whether

$$\sqrt{\sum_p h^2 \; (u_p^n - u_p^{n-1})^2} < \delta$$

with a prescribed threshold δ. For the semi-implicit scheme and small ε (then the downwards motion of the "steady state" is very slow) a good choice of threshold is $\delta = 10^{-5}$.

Reasonable time steps for our semi-implicit method are of order $(10h)^2$, e.g., for the discussed example very good results regarding CPU times and precision have been obtained for $\tau \in [0.001, 0.01]$. Since by a classical criterion the precision of numerical schemes for parabolic equations is optimal for $\tau \approx h^2$, we have computed also such case. But, no significant difference due to precision has been observed, only much longer CPU time was necessary. In our example $\tau = 5 * 10^{-3}$ and 20 time steps yield the segmentation result (using threshold $\delta = 10^{-5}$). On 2.4GHz Linux PC, the overall CPU time for this segmentation was 4.93 sec. (i.e., aproximately 0.25 sec. for one time step including construction of coefficients and solving the linear system). This CPU time was obtained with TOL=10^{-3}. Since we are mainly interested to find an "equlibrium", one can also decide that such precision is not necessary in every discrete time step. With increasing TOL a less number of SOR iterations are needed. Another way is to prescribe fixed number (but not too small) of iterations in every time step, e.g., ten prescribed SOR iterations lead to comparable segmentation with twice faster CPU time as mentioned above.

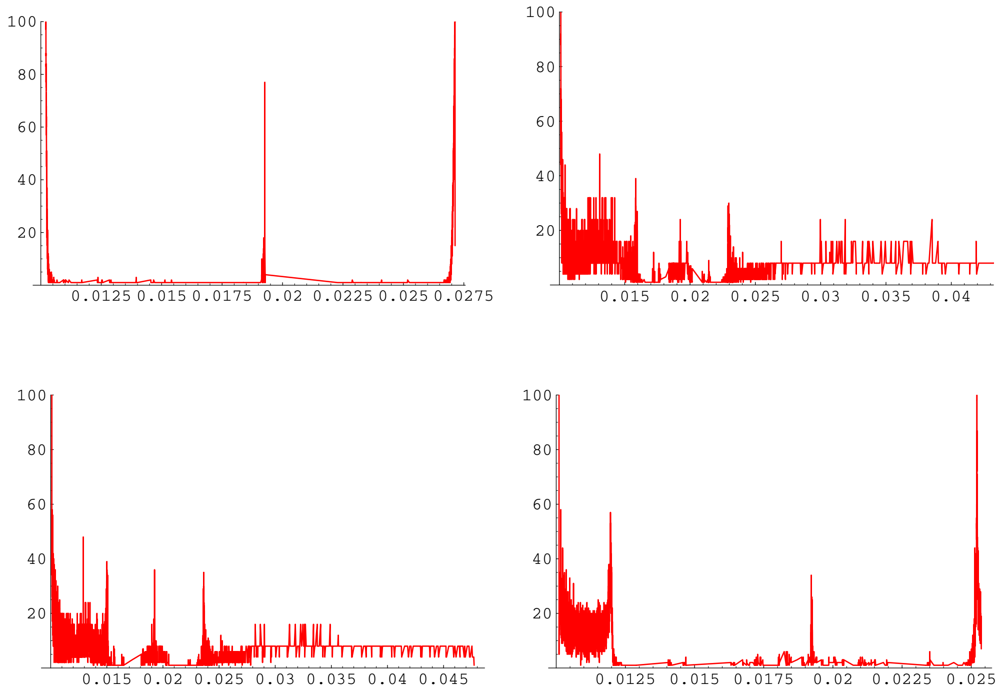


Figure 19: Histogram of the segmentation result given by semi-implicit scheme after 20 time steps (top left). Histograms of the segmentation function given by the explicit scheme after 500 (top right), 1000 (bottom left) and 5000 (bottom right) time steps. (Color Slide).

Now, let us look to behaviour of the explicit scheme in this example. We use the explicit version of the scheme (23) where also the whole second term on the left hand side is taken from the $(n-1)$st time step. Then, due to stability reasons, we have to choose $\tau = 5 * 10^{-6}$. Altough one explicit time step takes just 0.05 sec. (including construction of coefficients and explicit time update of the solution), to get a segmentation result comparable with the semi-implicit scheme we need about 10000 time steps. In Figure 19 we present histograms of the segmentation function, where plot range $[0, 100]$ in vertical direction has been chosen for visualization. We compare histograms, because one cannot use the same treshold δ for explicit and semi-implicit schemes due to very small change on solution between time steps in explicit scheme. In top left there is histogram of the segmentation result given by semi-implicit scheme after 20 time steps. The shocks in solution (corresponding to outer and inner edge of the circle) are given by two large gaps in histogram. In the top right there is a histogram of the segmentation function given by the explicit scheme after 500 time steps, and then after 1000 (bottom left) and 5000 (bottom right) time steps. We see that, due to necessity of small time step, the formation of the piecewise flat solution is very slow for explicit scheme. Altough after 1000 time steps one can see formation of two gaps which could be already used for detection of "final" segmentation result, the CPU time for 1000 steps of explicit scheme is 49.5 sec, which is ten times longer than for semi-implicit scheme. If we would like to obtain the similar histogram as plotted in top left using explicit scheme, we would need 100 times longer CPU time as in the case of semi-implicit scheme.

In all computations presented above, we have used $g(s) = \frac{1}{1+Ks^2}, K = 1$. In experiments without noise there is no significant difference by changing K. We get the same behaviour of the method changing K from 0.1 to 10. It is understandable, because the function g plays a role only along edges and its more ($K > 1$) or less ($K < 1$) fastly decreasing profile governs only speed by which level sets of solution are attracted to the edge from a small neighbourhood. Everywhere else only pure mean curvature motion is considered ($g = 1$).

The situation is different for noisy images like, e.g., depicted in Figure 12 right and Figures 16-17. The extraction of the circle in noisy environment takes a longer time (200 steps with $\tau = 0.01$ and $K = 1$) and it is even worse for $K = 10$. However, decreasing the parameter K gives stronger weight to mean curvature flow in noisy regions, so we can fastly extract the circle, in only 20 steps with the same $\tau = 0.01$. In case of noisy images also the convolution plays a role. E.g., if we switch off the convolution, the process is slower. But decreasing K can again improve the speed of segmentation process. In our computations we either do not apply convolution to I^0 or we use image pre-smoothing by $m \times m$ pixel mask with weights given by the

Gauss function normalized to unit sum.

We start all computations with initial function given as a peak centered in a "focus point" inside the segmented object, as plotted, e.g., in Figure 10 top left. Such function can be described at a circle with center s and radius R by $u^0(x) = \frac{1}{|x-s|+v}$, where s is the focus point and $\frac{1}{v}$ gives maximum of u^0. Outside the circle we take value u^0 equal to $\frac{1}{R+v}$. If one needs zero Dirichlet boundary data, e.g. due to some theoretical reasons (cf. [49, 11]), the value $\frac{1}{R+v}$ can be subtracted from the peak-like profile. If the computational domain Ω corresponds to image domain, we use $R = 0.5$. For small objects a smaller R can be used to speed up computations. Our choice of peak-like initial function is motivated by its nearly flat profile near the boundary of computational domain. However, other choices, e.g., $u^0(x) = 1 - \frac{|x-s|}{R}$, are also possible. If we put the focus point s not too far from a center of the mass of the segmented object, then we get only slightly different evolution of the segmentation function and same segmentation result.

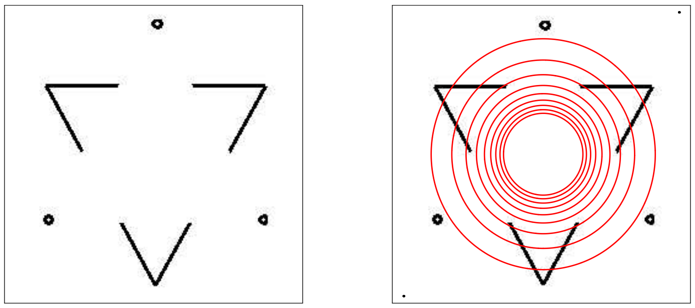


Figure 20: Image with subjective contours - double-Kanizsa triangle (left), image together with isolines of initial segmentation function (right). (Color Slide).

Now we will discuss some further segmentation examples. In Figure 20 we present image (234×227 pixels) with subjective contours of the classic triangle of Kanizsa. The phenomenon of contours that appear in the absence of physical gradients has attracted considerable interest among psychologists and computer vision scientists. Psychologists suggested a number of images that strongly requires image completion to detect the objects. In Figure 20 left, two solid triangles appear to have well defined contours, even in completely homogeneous areas. Kanizsa called the contours without gradient "subjective contours" [29], because the missed boundaries are provided by the visual system of the subject. We apply our algorithm in order to extract the solid triangle and complete the boundaries. In the next Figures 21-22 we present evolution of the segmentation function together with plots of level lines accumulating along edges and closing subjective contours areas. In this experiment we used $\varepsilon = 10^{-5}$, $K = 1$, $v = 0.5$, $\tau = 0.001$, TOL$= 10^{-3}$. For long time period (from 60th to 300th time step) we can easily detect also subjective contours of the second triangle. The first one, given by closing of the solid interrupted lines, is presented in Figure 22 bottom, visualizing level line $(\min(u) + \max(u))/2$. Interestingly, for bigger ε the second triangle has not been detected.

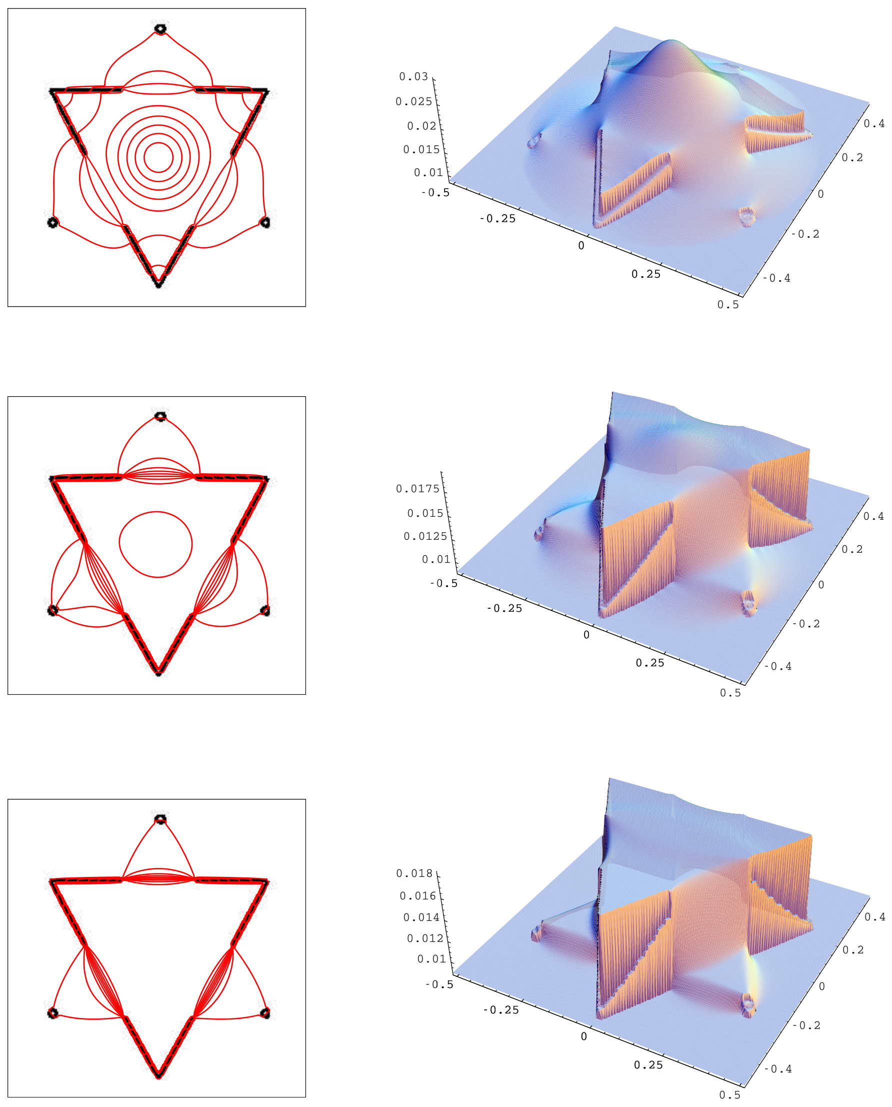


Figure 21: Level lines (left) and graphs of the segmentation function (right) in time steps 10, 30 and 60. (Color Slide).

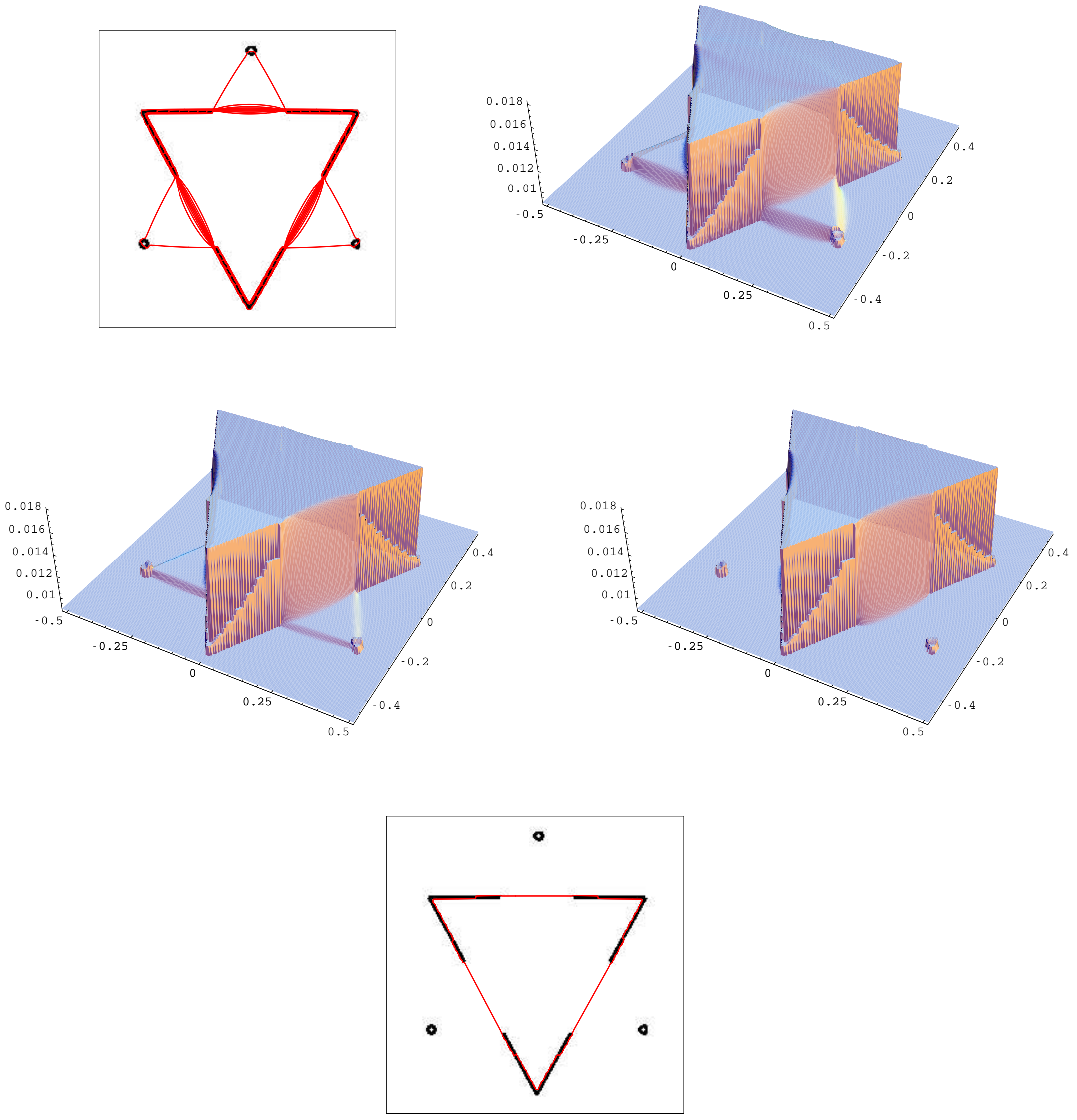


Figure 22: Level lines and graph of the segmentation function in time step 100 (top row). Then we show graphs of segmentation function after 300 and 800 steps (middle row). In the bottom we plot the segmented Kanizsa triangle. (Color Slide).

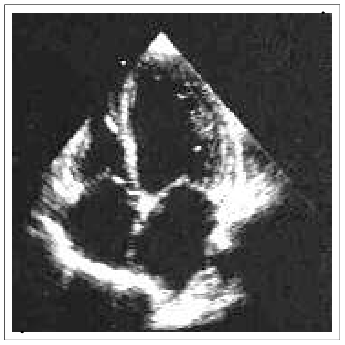


Figure 23: Echocardiographic image with high level of noise and gaps.

The next examples are related to medical image segmentation. First we process a 2D echocardiography (165×175 pixels) with high level of noise and gaps in ventricular and atrium boundaries (see Figure 23).

In Figure 24 we present segmentation of the left atrium. We start with peak-like segmentation function, $v = 1$, and we use $\varepsilon = 10^{-2}$, $K = 0.1$, $\tau = 0.001$, TOL$= 10^{-3}$ and $\delta = 10^{-5}$. In top row of the Figure we present result of segmentation with no pre-smoothing of the given echocardiography. In such case 68 time steps, with overall CPU time 6.54 sec., were needed for threshold δ. In top right we see graph of the final segmentation function. In the middle row we see its histogram (left) and zoom of the histogram around $\max(u)$ (right). By that we take level 0.057 for visualization of the boundary of segmented object (top left). In the bottom row we present the result of segmentation using 5×5 convolution mask. Such result is a bit smoother and 59 time steps (CPU time=5.65 sec.) were used.

For visualization of the segmentation level line in further figures we use the same strategy as above, i.e. the value of u just below the last peak of histogram (corresponding to upper "flat region") is chosen. In segmentation of the right atrium, presented in Figure 25, we took the same parameters as above and no pre-smoothing was applied. CPU time for 79 time steps was 7.59 sec. In segmentation of the left and right ventricles, with more destroyed boundaries, we use $K = 0.5$ and we apply 5×5 convolution mask (other parameters were same as above). Moreover, for the left ventricle we use double-peak-like initial function (see Figure 26 top) to speed up the process for such highly irregular object. In that case 150 time steps (CPU time = 14.5 sec) were used. For the right ventricle 67 time steps (CPU time = 6.57 sec.) was necessary to get segmentation result, see Figure 27.

In the last example given in Figure 28 we present segmentation of the mammography (165×307 pixels). Without pre-smoothing of the given image and with parameters $\varepsilon = 10^{-1}$, $K = 0.1$, $\tau = 0.0001$, $v = 1$, TOL$= 10^{-3}$ and $\delta = 10^{-5}$ we get the segmentation after 72 time steps. Since there are no big gaps we take larger ε and since the object is small (found in a shorter time) we use smaller time step τ.

5 Conclusions

In this chapter we introduced the semi-implicit co-volume level set method for solving the segmentation equation given by the Riemannian mean curvature flow of graphs. We discussed basic properties of the model, the role of model parameters and gave all details for computer implementation of the numerical algorithm. We also showed unconditional stability of our method and its high efficiency for these type of problems. The computational results related to medical image segmentation with partly missing boundaries and subjective contour extraction were discussed. The method was presented for 2D image segmentation. However, as it is common in level set methods, the extension to 3D case is straightforward and can be done easily using ideas of this chapter.

Acknowledgement. This work was supported by NATO Collaborative Linkage Grant No. PST.CLG.979123. The work of first author was supported by grant VEGA 1/0313/03, and by European project "Visual Modelling" in Stefan Banach International Mathematical Centre PAN and ICM, Warsaw University. This work was partially supported by MIUR, grant number 2002013422-005.

References

[1] Alvarez, L., Guichard, F., Lions, P.L. and Morel, J.M., Axioms and Fundamental Equations of Image Processing, Arch. Rat. Mech. Anal., Vol. 123, pp. 200-257, 1993.

[2] Alvarez, L., Lions, P.L. and Morel, J.M., Image selective smoothing and edge detection by nonlinear diffusion II, SIAM J. Numer. Anal., Vol. 29, pp. 845-866, 1992.

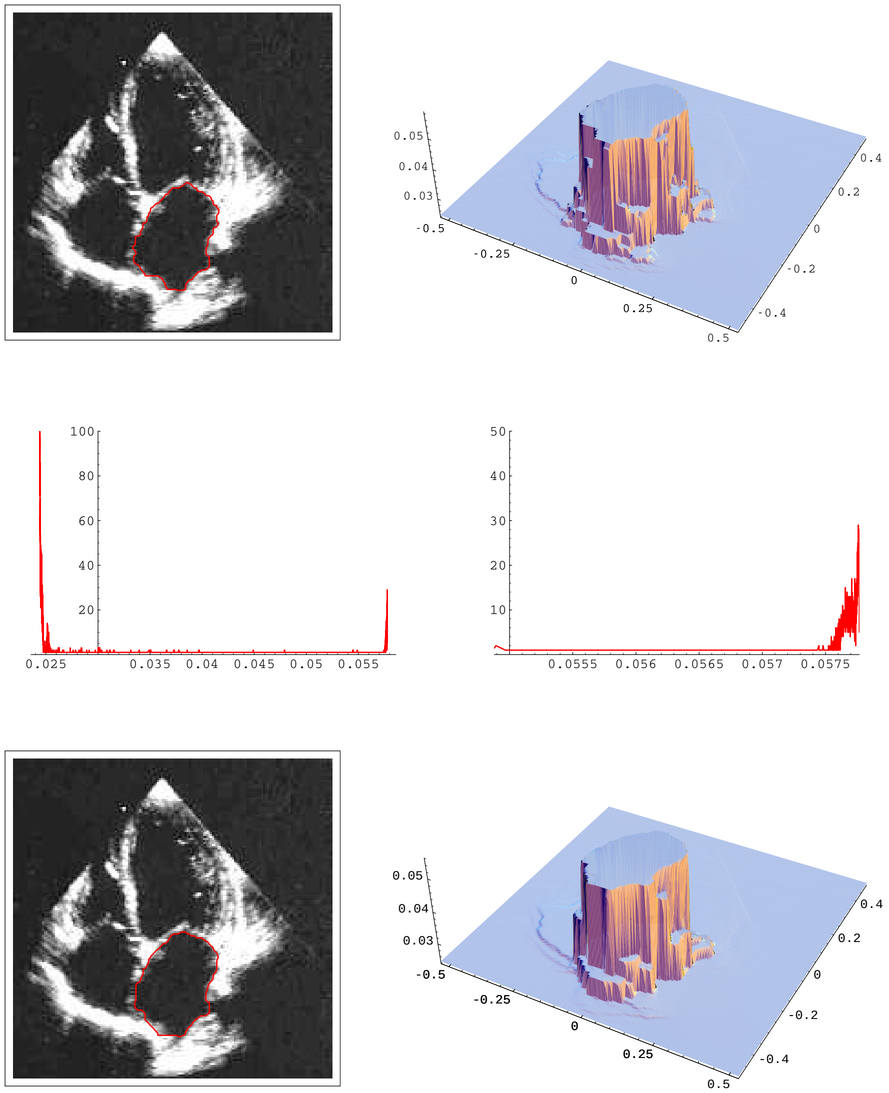


Figure 24: Segmentation level line and graph of the segmentation function for computation without convolution (top row); histogram of the segmentation function and its zoom (middle row). Segmentation level line and graph of the segmentation function for computation with convolution (bottom row). (Color Slide).

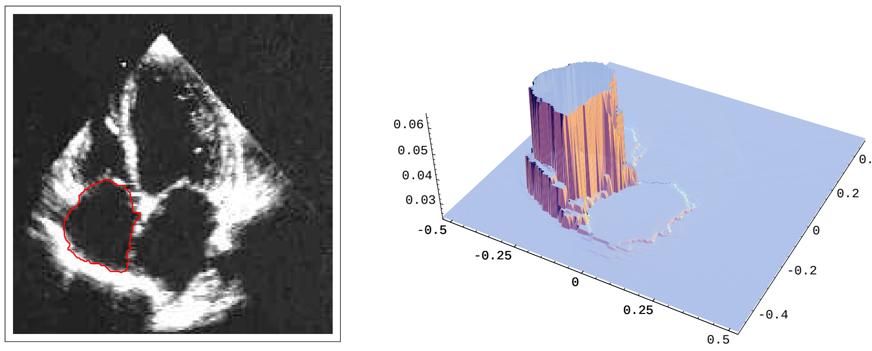


Figure 25: Segmentation level line and graph of the segmentation function for the right atrium. (Color Slide).

[3] Bänsch, E. and Mikula, K., A coarsening finite element strategy in image selective smoothing, Computing and Visualization in Science, Vol. 1, No. 1, pp. 53-61, 1997.

[4] Bänsch, E. and Mikula, K., Adaptivity in 3D image processing, Computing and Visualization in Science, Vol. 4, No. 1, pp. 21-30, 2001.

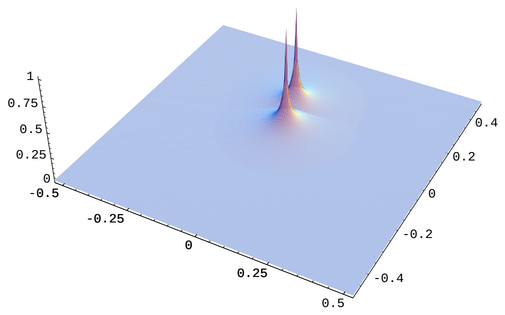
[5] Catté, F., Lions, P.L., Morel, J.M. and Coll, T., Image selective smoothing and edge detection by nonlinear diffusion, SIAM J. Numer. Anal., Vol. 29, pp. 182-193, 1992.

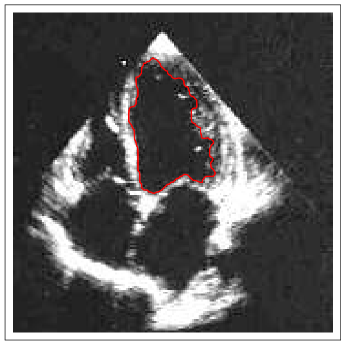
[6] Caselles, V., Catté, F., Coll, T. and Dibos, F., A geometric model for active contours in image processing, Numerische Matematik, Vol. 66, pp. 1-31, 1993.

[7] Caselles, V., Kimmel, R. and Sapiro, G., Geodesic active contours, in: Proceedings International Conference on Computer Vision'95, pp. 694-699, Boston, 1995.

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

[9] Caselles, V., Kimmel, R. and Sapiro, G. and Sbert, C., Minimal surfaces: a geometric three dimensional segmentation approach, Numer. Math., Vol. 77, pp. 423-451, 1997.





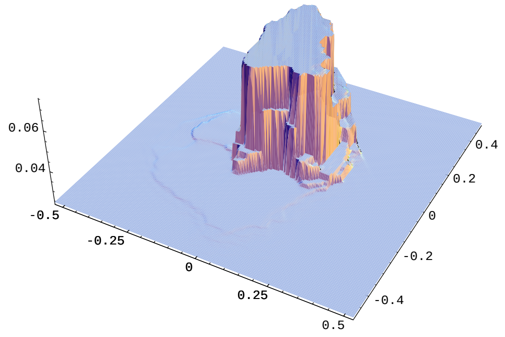


Figure 26: Initial double-peak segmentation function (top), segmentation level line and graph of the segmentation function for the left ventricle. (Color Slide).

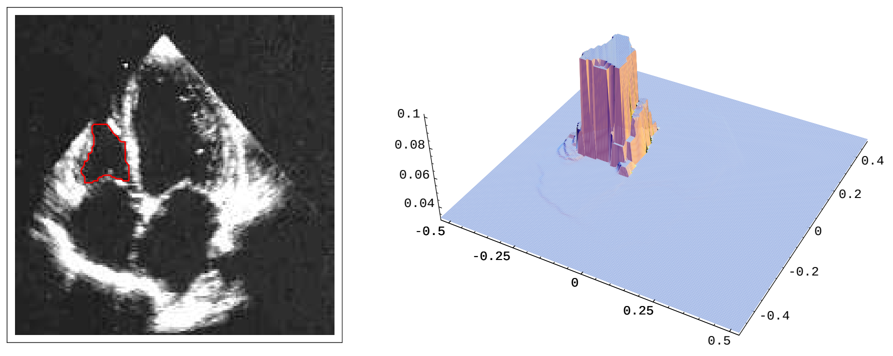


Figure 27: Segmentation level line and graph of the segmentation function for the right ventricle. (Color Slide).

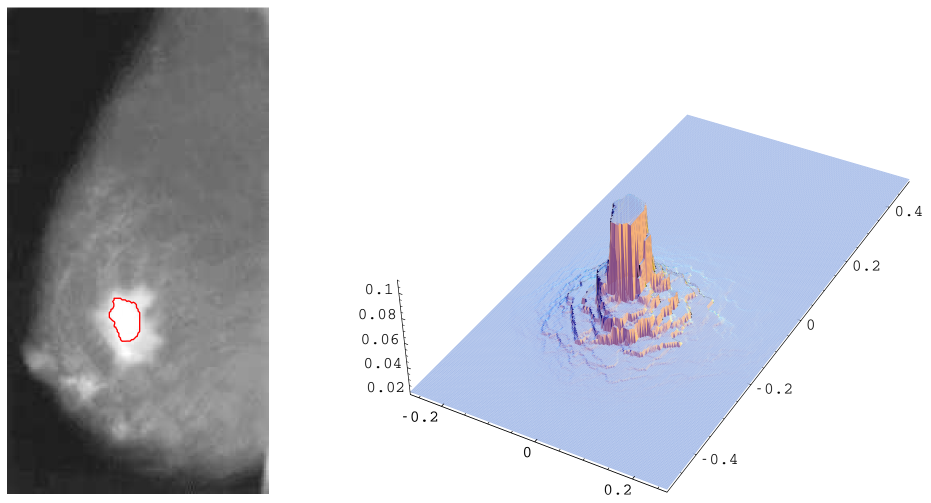


Figure 28: Segmentation level line and graph of the segmentation function for extraction of tumor in mammography. (Color Slide).

[10] Chen, Y.-G., Giga, Y. and Goto, S., Uniqueness and existence of viscosity solutions of generalized mean curvature flow equation, J. Diff. Geom., Vol. 33, pp. 749-786, 1991.

[11] Citti, G. and Manfredini, M., Long time behavior of Riemannian mean curvature flow of graphs, J. Math. Anal. Appl., Vol. 273, No. 2, pp. 353-369, 2002.

[12] Crandall, M.G., Ishii, H. and Lions, P.L., User's guide to viscosity solutions of second order partial differential equations, Bull.(NS) Amer. Math. Soc., Vol. 27, pp. 1-67, 1992.

[13] Deckelnick, K. and Dziuk, G., Convergence of a finite element method for non-parametric mean curvature flow, Numer. Math., Vol. 72, pp. 197-222, 1995.

[14] Deckelnick, K. and Dziuk, G., Error estimates for a semi implicit fully discrete finite element scheme for the mean curvature flow of graphs, Interfaces and Free Boundaries, Vol. 2, No. 4, pp. 341-359, 2000.

[15] Deckelnick, K. and Dziuk, G., A fully discrete numerical scheme for weighted mean curvature flow, Numer. Math., Vol. 91, pp. 423-452, 2002.

[16] Dziuk, G., Algorithm for evolutionary surfaces, Numer. Math., Vol. 58, pp. 603-611, 1991.

[17] Dziuk, G., Convergence of a semi discrete scheme for the curve shortening flow, Mathematical Models and Methods in Applied Sciences, Vol. 4, pp. 589-606, 1994.

[18] Dziuk, G., Discrete anisotropic curve shortening flow, SIAM J. Numer. Anal., Vol. 36, pp. 1808-1830, 1999.

[19] Evans, L.C. and Spruck, J., Motion of level sets by mean curvature I, J. Diff. Geom., Vol. 33, pp. 635-681, 1991.

[20] Eymard, R., Gallouet, T. and Herbin, R., The finite volume method, in: Handbook for Numerical Analysis, Vol. 7 (Ph. Ciarlet, J. L. Lions, eds.), Elsevier, 2000.

[21] Frolkovič, P. and Mikula, K., Flux-based level set method: a finite volume method for evolving interfaces, Preprint IWR/SFB 2003-15, Interdisciplinary Center for Scientific Computing, University of Heidelberg, 2003.

[22] Gage, M. and Hamilton, R.S., The heat equation shrinking convex plane curves, J. Diff. Geom., Vol. 23, pp. 69-96, 1986.

[23] Grayson, M., The heat equation shrinks embedded plane curves to round points, J. Diff. Geom., Vol. 26, pp. 285-314, 1987.

[24] Handlovičová, A., Mikula, K. and Sarti, A., Numerical solution of parabolic equations related to level set formulation of mean curvature flow, Computing and Visualization in Science, Vol 1., No. 2, pp. 179-182, 1999.

[25] Handlovičová, A., Mikula, K. and Sgallari, F., Semi–implicit complementary volume scheme for solving level set like equations in image processing and curve evolution, Numer. Math., Vol. 93, pp. 675-695, 2003.

[26] Handlovičová, A., Mikula, K. and Sgallari, F., Variational numerical methods for solving nonlinear diffusion equations arising in image processing, J. Visual Communication and Image Representation, Vol. 13, pp. 217-237, 2002.

[27] Kačur, J. and Mikula, K., Solution of nonlinear diffusion appearing in image smoothing and edge detection, Applied Numerical Mathematics, Vol. 17, pp. 47-59, 1995.

[28] Kačur, J. and Mikula, K., Slow and fast diffusion effects in image processing, Computing and Visualization in Science, Vol. 3, No. 4, pp. 185-195, 2001.

[29] Kanizsa, G., Organization in Vision, Hardcover, 1979.

[30] Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A. and Yezzi, A., Gradient flows and geometric active contours models, in Proceedings International Conference on Computer Vision'95, Boston, 1995.

[31] Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A. and Yezzi, A., Conformal curvature flows: from phase transitions to active vision, Arch. Rat. Mech. Anal., Vol. 134, pp. 275-301, 1996.

[32] Kass, M., Witkin, A. and Terzopulos, D., Snakes: active contour models, International Journal of Computer Vision, Vol. 1, pp. 321-331, 1987.

[33] Krivá, Z. and Mikula, K., An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing, J. Visual Communication and Image Representation, Vol. 13, pp. 22-35, 2002.

[34] Le Veque, R., Finite volume methods for hyperbolic problems, Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002.

[35] Lin C.J. and Moré, J.J., Incomplete Cholesky factorizations with limited memory, SIAM. J. Sci. Comput., Vol. 21, pp. 24-45, 1999.

[36] Malladi, R., Sethian, J.A. and Vemuri, B., Shape modeling with front propagation: a level set approach, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, pp. 158-174, 1995.

[37] Mikula, K. and Ramarosy, N., Semi–implicit finite volume scheme for solving nonlinear diffusion equations in image processing, Numer. Math., Vol. 89, No. 3, pp. 561-590, 2001.

[38] Mikula, K., Sarti, A. and Lamberti, C., Geometrical diffusion in 3D-echocardiography, Proceedings of ALGORITMY'97 - Conference on Scientific Computing, West Tatra Mountains - Zuberec (1997) pp. 167-181

[39] Mikula, K. and Ševčovič, D., Evolution of plane curves driven by a nonlinear function of curvature and anisotropy, SIAM J. Appl. Math., Vol. 61, pp. 1473-1501, 2001.

[40] Mikula, K. and Ševčovič, D., Computational and qualitative aspects of evolution of curves driven by curvature and external force, to appear in Computing and Visualization in Science, 2003.

[41] Mikula, K. and Sgallari, F., Semi-implicit finite volume scheme for image processing in 3D cylindrical geometry, to appear in Journal of Computational and Applied Mathematics, 2003.

[42] Osher, S. and Fedkiw, R., Level set methods and dynamic implicit surfaces, Springer-Verlag, 2003.

[43] Osher, S. and Sethian, J.A., Front propagating with curvature dependent speed: algorithms based on the Hamilton–Jacobi formulation, J. Comput. Phys., Vol. 79, pp. 12-49, 1988.

[44] Patankar, S., Numerical heat transfer and fluid flow, Hemisphere Publ. Corp., New York, 1980.

[45] Perona, P. and Malik, J., Scale space and edge detection using anisotropic diffusion, in Proc. IEEE Computer Society Workshop on Computer Vision, 1987.

[46] Saad, Y., Iterative methods for sparse linear systems, PWS Publ. Comp., 1996.

[47] Sapiro, G., Geometric Partial Differential Equations and Image Analysis, Cambridge University Press, 2001.

[48] Sarti, A., Malladi, R. and Sethian, J.A., Subjective Surfaces: A Method for Completing Missing Boundaries, Proceedings of the National Academy of Sciences of the United States of America, Vol. 12, No. 97, pp. 6258-6263, 2000.

[49] Sarti, A. and Citti, G., Subjective Surfaces and Riemannian Mean Curvature Flow Graphs, Acta Math. Univ. Comenianae, Vol. 70, No. 1, pp. 85-104, 2001.

[50] Sarti, A., Malladi, R. and Sethian, J.A., Subjective Surfaces: A Geometric Model for Boundary Completion, International Journal of Computer Vision, Vol. 46, No. 3, pp. 201-221, 2002.

[51] Sarti, A., Mikula, K. and Sgallari, F., Nonlinear multiscale analysis of three-dimensional echocardiographic sequences, IEEE Trans. on Medical Imaging, Vol. 18, pp. 453-466, 1999.

[52] Sarti, A., Mikula, K., Sgallari, F. and Lamberti, C., Nonlinear multiscale analysis models for filtering of 3D + time biomedical images, in: Geometric methods in bio-medical image processing (Ed. R.Malladi), Springer, pp. 107-128, 2002.

[53] Sarti, A., Mikula, K., Sgallari, F. and Lamberti, C., Evolutionary partial differential equations for biomedical image processing, Journal of Biomedical Informatics, Vol. 35, pp. 77-91, 2002.

[54] Sethian, J.A., Numerical algorithm for propagating interfaces: Hamilton–Jacobi equations and conservation laws, J. Diff. Geom., Vol. 31, pp. 131-161, 1990.

[55] Sethian, J.A., Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science, Cambridge University Press, 1999.

[56] Walkington, N.J., Algorithms for computing motion by mean curvature, SIAM J. Numer. Anal. Vol. 33, No. 6, pp. 2215-2238, 1996.

[57] Weickert, J., Anisotropic Diffusion in Computer Vision, Teubner-Stuttgart, 1998.

[58] Weickert, J., Romeny, B.M.t.H. and Viergever, M.A., Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Trans. Image Processing, Vol. 7, No.3, pp. 398-410, 1998.

6 Questions and Answers

(Q1) Outline the level set segmentation models used in the last decade. What is an advection-diffusion mechanism in such models?
(Q2) What is a difference between previous level set segmentation models and Riemannian mean curvature flow of graphs discussed in this chapter?
(Q3) What are the main principles and advantages of the semi-implicit time discretization?
(Q4) How the segmentation partial differential equation (2) is discretized by the co-volume method?
(Q5) What are differences between semi-implicit co-volume method and explicit finite difference method?
(Q6) What are the properties of the system matrix given by the semi-implicit co-volume scheme?
(Q7) How can you get the unconditional stability of the semi-implicit co-volume level set method?
(Q8) What are the efficient methods for solving linear systems arising in the semi-implicit co-volume level set method?

(A1) First level set model with the speed of segmentation curve modulated by $g(|\nabla G_\sigma * I^0|)$, where g is a smooth edge detector function, e.g. $g(s) = 1/(1+s^2)$, has been given by Caselles et al. and Malladi et al. in [6, 36]. Due to shape of the Perona-Malik function g, the moving segmentation curve is strongly slowed down in a neighbourhood of an edge leading to a segmentation result. Later on, the curve evolution and the level set models for segmentation have been significantly improved by finding a driving force in the form $-\nabla g(|\nabla G_\sigma * I^0|)$ by Caselles et al. and Kichenassamy et al. in [8, 31]. The vector field $-\nabla g(|\nabla G_\sigma * I^0|)$ points towards regions where the norm of the smoothed gradient of I^0 is large. Thus if an initial curve belongs to a neighborhood of an edge, then it is driven towards this edge by this proper velocity field. However, in a noisy environment, the evolving level set can behave very irregularly, it can be attracted to spurious edges and no reasonably convergent process can be observed. To prevent such situation one has to regularize the evolution. If evolution of a curve in the normal direction depends on its curvature k, then the sharp irregularities are smoothed. Such motion can be interpreted as an intrinsic diffusion of the curve. A reasonable regularization term is given by $g(|\nabla G_\sigma * I^0|)k$, where the amount of curve intrinsic diffusion is small in the vicinity of un-spurious

edges. Such regularization by the intrinsic diffusion is used in all above mentioned level set segmentation models.
(A2) The main difference is that not the evolution of one particular level set (segmentation curve) [6, 36, 8, 31] but the evolution of the graph of segmentation function [48, 49, 50] is used in order to segment an image object. In the segmentation method presented in this chapter, a "point-of-view" surface, given by an observer (user) chosen fixation point inside the image, is taken as u^0 and this initial state of the segmentation function is evolved by equation (2), until the piecewise flat subjective surface arises. For small ε, the subjective surface closes gaps in image object boundary and is stabilized, i.e. is almost unchanging by further evolution, so it is easy to stop the segmentation process. In standard level set approach following the evolution of one level set, the redistancing is used to keep unit slope along the level set of interest (e.g. along segmentation curve). In such approach the evolution of u itself is forgotten at every redistancing step. Such solution prevents steepening of u and one cannot obtain the subjective surfaces. In the co-volume level set method numerically computed segmentation function can naturally evolve to a "piecewise constant steady state" result. The equation (2) is the ε-regularization of equation (8) and they have the same advection term, $\nabla g^0.\nabla u$, which accumulates segmentation function along the edge and forms a shock. However, diffusion mechanisms are different. In equation (8), the diffusion mechanism is given by the mean curvature flow of level sets of u. In spite of that, the equation (2), e.g. in case $\varepsilon = 1$, gives diffusion which is known as mean curvature flow of graphs. It means that, not level sets of u move in normal direction proportionally to curvature, but the graph of segmentation function moves (as 2D surface in 3D space) in the normal direction proportionally to the mean curvature. Thus, the large variations in the graph of segmentation function are smoothed due to large mean curvature. However, such smoothing is applied only outside edges where advection term dominates. In case $\varepsilon = 1$, the equation (2) can be interpreted as a time relaxation for the minimazion of the weighted area functional or as the mean curvature motion of a graph in Riemann space with metric $g^0\delta_{ij}$. By decreasing ε the metric is stretched in vertical z direction and one gets the model presented in this chapter.
(A3) For time discretization of nonlinear diffusion equations there are basically three posibilities – implicit, semi-implicit or explicit schemes. In all approaches the time derivative is replaced by time difference. In the explicit schemes all further differential terms are taken at the old time level, in implicit schemes all such terms are taken at the new time level, while in

the semi-implicit method the nonlinear terms are treated from the previous time step and the linear ones are considered on the current time level. The semi-implicit discretization in time yields un-conditional stability of the numerical solution. This is the main advantage in comparison with explicit time stepping, where the stability is often achieved only under severe time step restriction. Since in nonlinear diffusion problems (like the level set equation) the coefficients depend on the solution itself and thus they must be recomputed in every discrete time update, an overall CPU time for explicit scheme can be tremendous. On the other hand, the implicit time stepping, altough it is unconditionally stable, leads to solution of nonlinear systems in every discrete time update. For the level set like problems there is no guarantee for convergence of a fast Newton solver and fixed point like iterations are very slow. From this point of view, the semi-implicit method seems to be optimal regarding stability and efficiency. In every time update we solve linear system of equations which can be done efficiently using, e.g., suitable preconditioned iterative linear solvers.
(A4) A digital image I^0 is given on a structure of pixels. Since discrete values of I^0 influence the model, in the co-volume scheme an approximation of the segmentation function u is related to a triangulation which is derived from the pixel structure. A piecewise linear approximation of the segmentation function on triangulation gives constant value of its gradient per triangles, which allow simple and clear construction of fully-discrete system of equations. The co-volume mesh is constructed in such a way that it corresponds exactly to the pixel structure of the image inside the computational domain Ω. Then the equation (2) is integrated in every co-volume to get the integral formulation (14). Then the exact "fluxes" and "capacity function" are approximated numerically and finally one gets a linear system of equations to compute the segmentation function at the new time level.
(A5) The explicit scheme combined with finite differences in space is usually based on formulations like (9) where all derivatives are expanded to get curvature and advection terms. In such case the equation (2) for $\varepsilon = 1$ is written in the form

$$u_t = g^0 \frac{(1+u_{x_2}^2)u_{x_1x_1} - 2u_{x_1}u_{x_2}u_{x_1x_2} + (1+u_{x_1}^2)u_{x_2x_2}}{1+u_{x_1}^2+u_{x_2}^2} + g_{x_1}^0 u_{x_1} + g_{x_2}^0 u_{x_2}.$$

In this form, it is not clear which terms to take from previous and which on the current time level, having in mind the unconditional stability of the method. In spite of that, the basic formulation (2) leads naturally to convenient semi-implicit time discretization, and the co-volume technique relies on its integral formulation. In such a way, the discretization scheme naturally respects a variational structure of the problem, it gives clear discrete form of local mass balance, and it naturally fulfills discrete minimum-maximum principle.

(A6) The matrix of the system is symmetric and diagonally dominant M-matrix. These properties imply that the linear system has unique solution and it fulfills the minimum-maximum principle (i.e., no spurious oscillations appear).

(A7) In fact, there exists unique solution $(u_1^n, \dots, u_M^n)$ of the semi-implicit co-volume level set method for any $\tau > 0$, $\varepsilon > 0$ and for every $n = 1, \dots, N$. Moreover, un-conditional stability estimate

$$\min_p \ u_p^{n-1} \le \min_p \ u_p^n \le \max_p \ u_p^n \le \max_p \ u_p^{n-1}$$

holds for any n, $1 \le n \le N$. To prove it, we may rewrite (23) into the equivalent form

$$u_p^n + \frac{\tau}{b_p^{n-1}} \sum_{q \in C_p} a_{pq}^{n-1}(u_p^n - u_q^n) = u_p^{n-1}$$

and let $\max(u_1^n, \dots, u_M^n)$ be achieved in the node p. Then the whole second term on the left hand side is nonnegative and thus $\max(u_1^n, \dots, u_M^n) = u_p^n \le u_p^{n-1} \le \max(u_1^{n-1}, \dots, u_M^{n-1})$. In the same way we can prove the relation for minimum and together we have the desired stability estimate.

(A8) The system matrix has known (penta-diagonal) structure and it is symmetric and diagonally dominant M-matrix. One of the most popular iterative methods, namely Successive Over Relaxation (SOR) method has guaranteed convergence for this type of problems. However, if the convergence is slow due to condition number of the system matrix (which increases with number of unknowns and for increasing τ and decreasing ε), faster iterative methods can be used. E.g., the preconditioned conjugate gradient methods allow fast convergence, altough they need more storage. If the storage requirements are reduced, then they can be very efficient and robust [25].

7 Key Words

```
level set equation
mean curvature flow
```

image segmentation
convolution
regularization
missing boundaries
subjective contours
subjective surfaces
medical images
Riemannian mean curvature flow
metric
co-volume method
variational structure
semi-implicit scheme
nonlinear diffusion
unconditionally stable
image smoothing
vector field
advection
minimal surface
triangulation
M-matrix
SOR method
preconditioned conjugate gradient method
CPU time