Semi-implicit co-volume method in 3D image segmentation

Stefania Corsaro*, Karol Mikula†, Alessandro Sarti‡, Fiorella Sgallari§

Abstract

We introduce three-dimensional semi-implicit complementary volume numerical scheme for solving the level set formulation of (Riemannian) mean curvature flow problem. We apply the scheme to segmentation of objects (with interrupted edges) in 3D images. The method is unconditionally stable, the study of its experimental order of convergence on 3D examples shows its accuracy and it is efficient regarding computational times.

1 Introduction

In this paper we introduce an efficient algorithm for solving the Riemannian mean curvature flow equation

$$u_t = \sqrt{\varepsilon^2 + |\nabla u|^2}\nabla . \left(g(|\nabla G_\sigma * I^0|) \frac{\nabla u}{\sqrt{\varepsilon^2 + |\nabla u|^2}} \right) \tag{1}$$

in the context of 3D image segmentation. Here, $u(t,x)$ is an unknown, we call it *segmentation function* or *subjective surface* [38, 39, 40], defined in $Q_T \equiv [0,T] \times \Omega$. $\Omega \subset I\!R^d$ is a bounded domain with a Lipschitz continuous boundary $\partial\Omega$, $d = 3$ in our case of 3D segmentation, $[0,T]$ is a time interval, I^0 is a given image and $\varepsilon > 0$ is a parameter. The equation is accompanied either with Dirichlet boundary conditions

$$u(t,x) = u^D \quad \text{in } [0,T] \times \partial\Omega, \tag{2}$$

or Neumann boundary conditions

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

where ν is unit normal to $\partial\Omega$, and with the initial condition

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

In image segmentation we use Dirichlet boundary conditions and without lost of generality we may assume $u^D = 0$. The zero Neumann boundary conditions are often used in

*CNR, Institute for High-Performance Computing and Networking (ICAR), Via P. Castellino 111, 80131 Napoli, Italy (e-mail: stefania.corsaro@dma.unina.it)

†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)

computations of interface motions in free boundary problems (see e.g. [41, 33]) or in the morphological image smoothing (see e.g. [2, 22]). We use them in testing the accuracy of our numerical solution when dealing with motion of a particular level set of function u.

The Perona-Malik function $g : \mathbb{R}_0^+ \to \mathbb{R}^+$ is nonincreasing, $g(0) = 1$, admitting $g(s) \to 0$ for $s \to \infty$ [36]. Usually we use the function $g(s) = 1/(1 + Ks^2)$, $K \geq 0$. $G_\sigma \in C^\infty(\mathbb{R}^d)$ 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

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

with $\tilde{I}^0$ the extension of I^0 to $\mathbb{R}^d$ given by periodic reflection through the boundary of image domain. The computational domain $\Omega \subset \mathbb{R}^d$ 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 initial state of the segmentation function is bounded, i.e. $u^0 \in L_\infty(\Omega)$. In our approach, the segmentation is an evolutionary process given by the solution of equation (1) and T represents a time when a segmentation result is achieved. For shortening notations, we will use abbreviation

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

Due to properties of function g and smoothing effect of the convolution we always have $1 \geq g^0 \geq \nu_\sigma > 0$ [6, 23].

The equation (1) is the ε-regularization, in the sense of Evans and Spruck,

$$|\nabla u| \approx |\nabla u|_\varepsilon = \sqrt{\varepsilon^2 + |\nabla u|^2} \tag{8}$$

of the segmentation equation suggested in [8, 9, 25]

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

In [17, 10], the existence of viscosity solution [12] of the curvature driven level set equation [34], i.e. equation (9) with $g^0 \equiv 1$, was proven. Analysis of equation (9) and (1), respectively, were done in [25, 9] and [39, 11]. In [17], the ε-regularization (8) was used as a tool to prove existence of a viscosity solution. Rescaling the motion of graph by mean curvature by a factor $\frac{1}{\varepsilon}$, and letting $\varepsilon \to 0$, gives the level set evolution. Using the ε-regularization in equation (9) leads to a mean curvature flow of graphs with respect to a specific Riemann metric given by the image features, and, ε is a modelling parameter - it can help in completing of interrupted edges, e.g. in case of noisy images [39]. The idea to use such Riemannian mean curvature flow of graphs to extract the so-called subjective contours [24] and to segment images with interrupted edges originates in [38, 39, 40], see also [16, 45]. It is called *subjective surfaces method*. In spite of other level set techniques used for segmentation, subjective surfaces method does not move one particular (e.g. zero) level set to boundary of segmented object, but it moves there all the level sets. The standard level set methods prevent discontinuity formations in level set function, e.g., by the

reinitialization of front. In spite of that, in the subjective surfaces method, discontinuities are allowed and are the most important part of the solution. The forming discontinuity (shock) in evolving graph of the solution is used to detect object boundaries. We will discuss and illustrate the corresponding ideas in the next section.

In the computational method for solving (1) suggested in this paper we use an efficient, unconditionally stable, semi-implicit time discretization, and a new 3D co-volume spatial discretization suitable for image processing applications. Since the motion of graph given by equation (1) is regularization converging with $\varepsilon \to 0$ to the level set flow (9), our method can also be used for efficient solution of equation (9) and for any other 3D curvature driven level set application, too.

For time discretization of nonlinear diffusion equations there are basically three possibilities – implicit, semi-implicit or explicit schemes. For spatial discretization usually finite differences [41, 33], finite volumes [35, 18, 27, 30] or finite element methods [5, 42, 13, 14, 15, 4, 23] are used. The co-volume technique (called also complementary volume or finite volume-element method) is a combination of finite element and finite volume methods. The discrete equations are derived using the *finite volume methodology*, i.e. integrating equation in the so-called control (complementary, finite) volume. Very often the control volumes are constructed as elements of a dual (complementary) grid to a *finite element triangulation* (tetrahedral grid in 3D case). Then nonlinear quantities, as diffusion coefficients on faces of co-volumes and/or a capacity function inside every co-volume, are evaluated using piecewise linear approximation of solution on triangulation employing thus the methodology of the linear finite element method. Finite volume methodology brings naturally discrete minimum-maximum principle. The piecewise linear representation (reconstruction) of segmentation function on the finite element grid yields a fast and simple evaluation of nonlinearities usually depending on the absolute value of gradients.

Implicit, i.e. nonlinear time discretization and co-volume technique for solution of the level set equations was first introduced in [43]. The implicit time stepping as in [43], 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 efficient nonlinear solver known so far and simple fixed point like iterations are very slow [43]. The efficient 2D co-volume level set method based on semi-implicit, i.e. linear, time discretization was given and studied in [22]. In [22], the method was applied to 2D image smoothing nonlinear diffusion level set equation [2].

The equation (9) can be rewritten into the advection-diffusion form

$$u_t = g^0 |\nabla u| \nabla . \left(\frac{\nabla u}{|\nabla u|} \right) + \nabla g^0 . \nabla u. \tag{10}$$

Various finite difference schemes [8, 9, 25, 38, 39, 40] are usually based on this form using up-winding in advection term and explicit time stepping. In spite of that, our co-volume technique relies on discretization of the basic form (9), or more precisely on its regularization (1), and we use its integral formulation. In such way, the discretization scheme naturally respects variational structure of the problem and it gives clear and simple derivation of difference equations. Our semi-implicit discretization in time fulfills unconditionally discrete minimum-maximum principle (L_∞-stability), i.e., no spurious oscillations appear 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 computational time for explicit scheme can be tremendous. The explicit scheme combined with finite differences in space is usually based on formulations like (10) where moreover all derivatives are expanded to get curvature and advection terms. Then, e.g. in 2D, equation (1) 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. In this form, it is not clear (reader may try) which term to take from previous and which one on the current time level, having in mind the unconditional stability of the method. In spite of that, the basic formulation (1) leads naturally to convenient semi-implicit time discretization. Since the implicit time stepping as in [43] is very slow due to necessity to solve nonlinear systems, the semi-implicit method seems to be optimal regarding stability and computational efficiency. Let us also 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 the next section we discuss some related models leading to equation (1). In section 3 we introduce in details our 3D semi-implicit co-volume method. In section 4 we study its experimental orders of convergence and discuss segmentation experiments. The paper is finished by conclusions.

2 Related models

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 noise, which is intrinsically linked to modern non-invasive acquisition techniques (as ultrasound), or in images with occlusions or subjective contours (in some psychologically motivated examples), these edges can be very irregular or even interrupted. Then the image analysis and segmentation of objects become a difficult task.

In the so-called active contour models [26] an evolving family of curves converging to edges is constructed. A simple approach (similar to various discrete region growing algorithms) is to put a small seed, e.g. a small circle in 2D case, or a small ball in 3D case, inside the object and then evolve this, let us call it segmentation curve or surface, to find automatically the object boundary. For such moving curves and surfaces, the level set models have been introduced in the last years. A basic idea in the level set methods [34] is that the moving curve or surface corresponds to the evolution of a particular level-line or level-surface of the so called level-set function u which solves some form of the following general level set equation $u_t = F|\nabla u|$ in which F represents the normal component of the velocity of this motion. The level set approach to moving curves or surfaces is robust in the sense that it can handle changes of topology and track various singularities like corners etc. On the other hand it is computationally more expensive, because it adds one more dimension to the original problem.

The 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 smooth edge detector

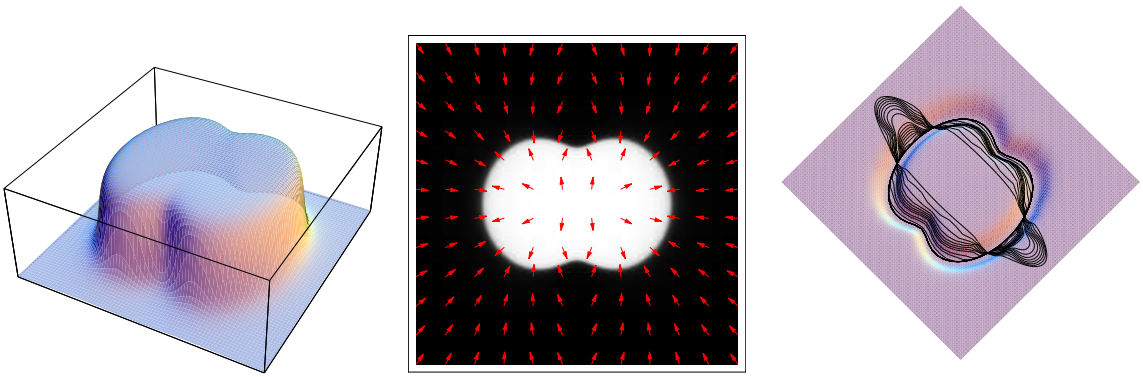

Figure 1: Left: A graph of the image intensity function $I^0(x)$; Middle: Image given by the intensity $I^0(x)$ plotted together with arrows representing the vector field $-\nabla g(|\nabla I^0(x)|)$; Right: An initial ellipse driven by the vector field $-\nabla g(|\nabla I^0(x)|)$ down to the valley in the graph $g(|\nabla I^0(x)|)$ to find the edge in the image.

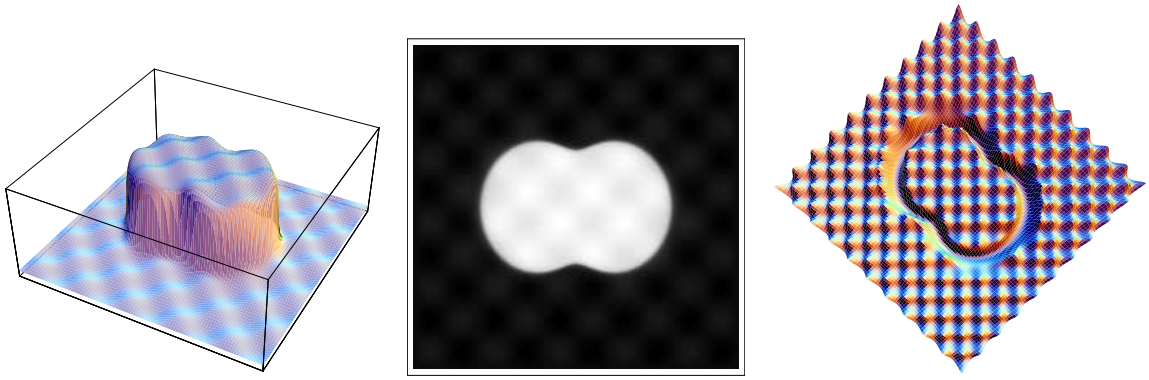

Figure 2: The situation is more complicated in case of a "noisy" image (middle); we plot also a graph of its intensity $I^0(x)$ (left) and corresponding surface $g(|\nabla I^0(x)|)$ (right).

function, e.g. $g(s) = 1/(1+Ks^2)$, has been given in [7] and [28]. In such model, a "steady state" of the segmentation curve corresponds to boundary of segmented object. Due to the shape of the Perona-Malik function g, the moving 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 introducing a driving force in the form $-\nabla g(|\nabla I^0(x)|)$ [8, 9, 25]. 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 Figure 1, illustrating 2D situation). 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, as one can see from Figures 2 and 3, the situation is much more complicated in the case of noisy images. The advection 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 the left

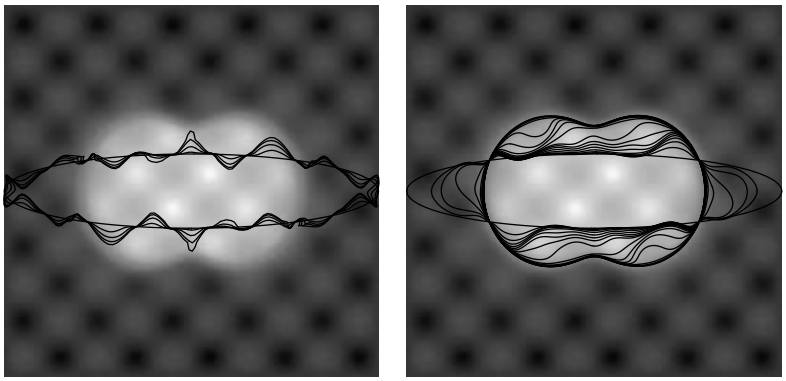

Figure 3: The evolution only by advection leads to attracting a curve (initial ellipse) to spurios edges, the evolution must be stopped without any reasonable segmentation result (left); by adding regularization term related to curvature of evolving curve the edge is found smoothly (right).

part of Figure 3. To prevent such situations, one has to regularize the evolution. A helpful regularization is to add a curvature dependence to the level set flow. If normal velocity of evolution depends on curvature k, then the sharp curve irregularities are smoothed. Such motion represents the so-called intrinsic diffusion of the curve [20, 21]. An appropriate regularization term is given by $g^0 k$, where the amount of curve intrinsic diffusion is small in the vicinity of an un-spurious edge. In the right part of Figure 3, we present initial ellipse evolution to successfull segmentation result using such combined advection-intrinsic-diffusion model computed by the Lagrangean method from [31]. The level set formulation of such curve evolution is given by the equation (10) which is of course only another form of equation (9). The same considerations as above, described for curves, hold also for evolving surfaces in 3D, with the only difference that regularization depends on the mean curvature of surface.

However, there is still a practical problem with the previous approach. It behaves very well, if the initial segmentation curve is in a vicinity of the edge. But, if it is not a case, in general, it is difficult to drive an arbitrary initial segmentation curve there. E.g., 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 [20, 21]). 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 (9).

An important observation now is, that equation (9) moves not only one particular level set (segmentation curve or surface) but all the level sets by the above mentioned advection-intrinsic-diffusion mechanism. So, in spite of the previously mentioned segmentation approaches, one may begin to think not about the evolution of one particular level set but on evolution of the whole surface composed by those level sets. The idea to look on the graph evolution is a basis of the subjective surfaces method. Initially, a "point-of-view" surface, given by the user chosen fixation point inside the image, is taken as u^0 (see Figure 4 top right for illustration in 2D situation). Then this initial state of the segmentation function is evolved by equation (1), until the so-called subjective surface arises (see e.g. Figure 4 bottom right). During the evolution, first, the isolines which are close to the edge, where the advection term dominates, are attracted from both sides to this edge. A shock (steep gradient) is formed due to accumulation of these level sets. In the regions

outside edges the advection term is vanishing and $g^0 \equiv 1$, so only intrinsic diffusion (if we think about ε close to zero) 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 the segmentation function until the upper level of the shock is achieved and in such way the flat region in the profile of segmentation function inside the object is formed. Outside of the object, the level sets are also shrinking, until they are attracted by nonzero velocity field and then they contribute to the shock. In the bottom right of Figure 4 we see the shape of segmentation function u after such evolution, and it is very easy to use one of the level lines, e.g. $(\max(u) + \min(u))/2$, to get the boundary of segmented object. Increasing the value of ε can even speed-up diffusion and flattening of this profile inside and outside the edges. In case of objects with edge interruptions, however, we need small ε to stretch the Riemannian metric and to close the interrupted shocks in the graph of segmentation function by minimal surfaces.

3 Computational method

We present our method formally in discretization of equation (9), although we always use its ε-regularization (1) with a specific $\varepsilon > 0$. The notation is simpler in case of (9) and it will be clear where regularization appears in the numerical scheme.

3.1 Semi-implicit time discretization

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 (9) 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. By such approach we get our semi-discrete in time scheme:

Let τ and σ be fixed numbers, I^0 be a given image and u^0 be a given initial segmentation function. Then, for every discrete time moment $t_n = n\tau$, $n = 1, \dots N$, we look for a function u^n, solution of the equation

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

3.2 Co-volume spatial discretization in 3D

A 3D digital image is given on a structure of voxels with cubic shape, in general. Since discrete values of image intensity I^0 are given in voxels and they influence the model, we will relate spatially discrete approximations of the segmentation function u also to the voxel structure, more precisely, to voxel centers. In every discrete time step t_n of the method (11) we have to evaluate gradient of the segmentation function at the previous step $|\nabla u^{n-1}|$. To that goal we put 3D tetrahedral grid into the voxel structure and take a piecewise linear approximation of the segmentation function on such a grid. Such approach will give a constant value of gradient in tetrahedras (which is the main feature of the co-volume [43, 22] and linear finite element [13, 14, 15] methods in solving mean curvature flow in the level set formulation) allowing simple, clear and fast construction of fully-discrete system of equations.

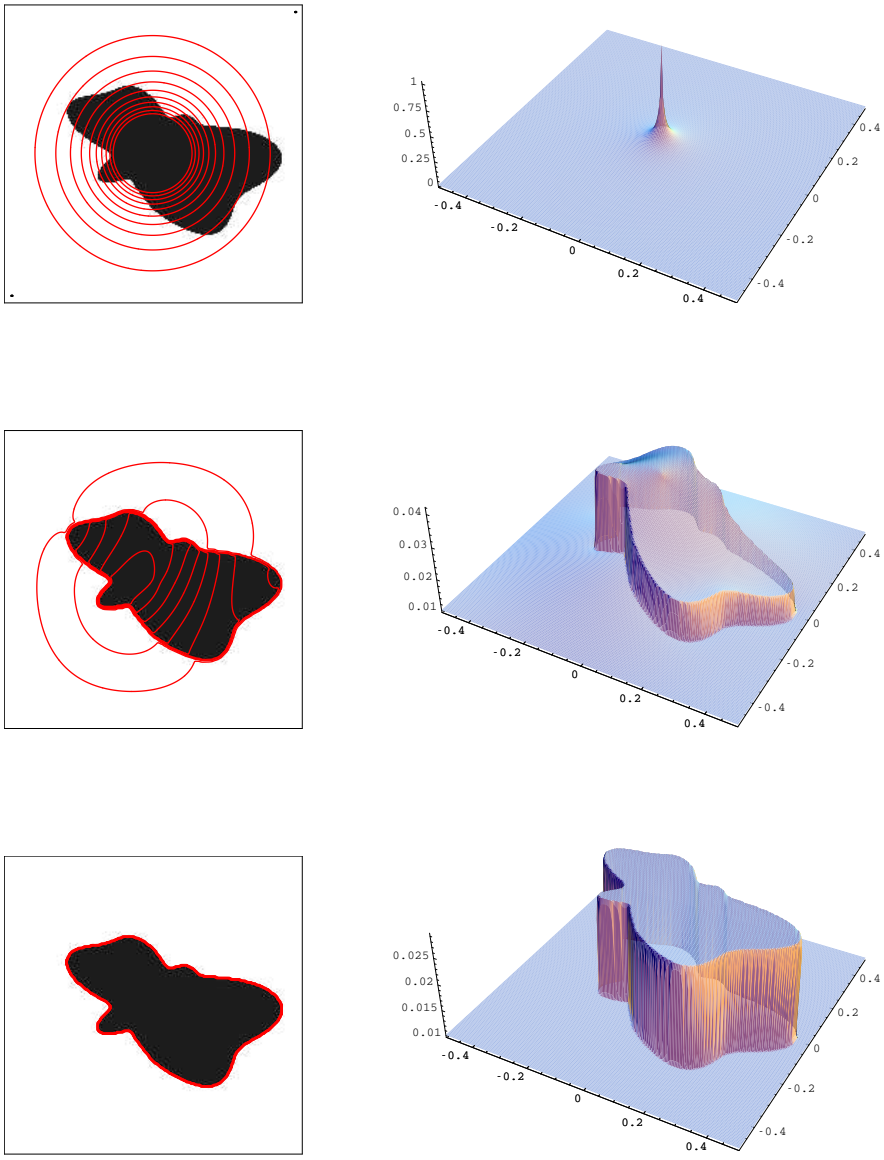

Figure 4: Subjective surface based segmentation of a "batman" image. In the left column we plot the black and white image to be segmented together with isolines of the segmentation function. In the right column there is a shape of the segmentation function. The rows correspond to time steps 0,1 and 10 which gives the final result.

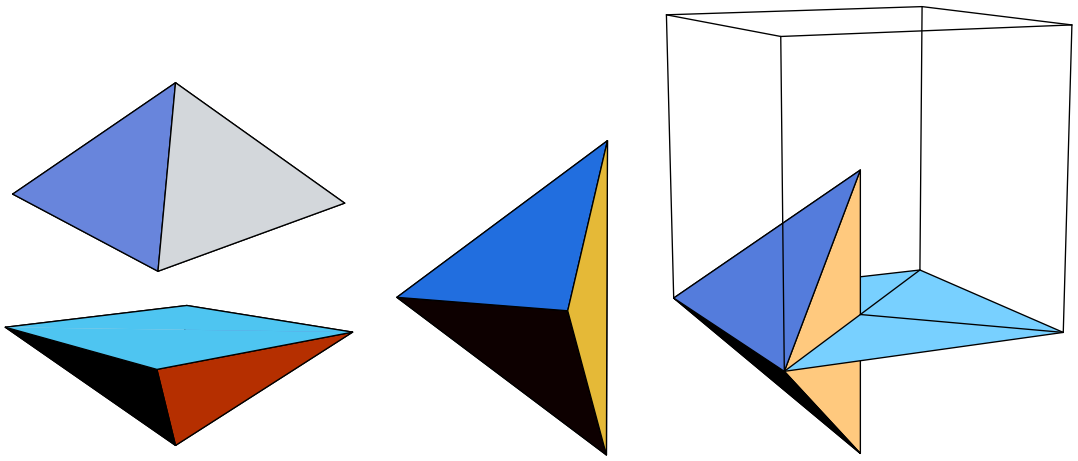

Figure 5: Neighbouring pyramids which are joined together (left); joining these pyramids and then splitting into four parts give tetrahedron of our 3D grid (middle); intersection of the tetrahedron with the bottom face of voxel–co-volume (right).

The formal construction of our co-volumes will be given in the next paragraph and we will see that the co-volume mesh corresponds back to the image voxel structure, what is reasonable in image processing applications. On the other hand, the construction of co-volume mesh has to use 3D tetrahedral finite element grid to which it is complementary. This will be possible using following approach. First, every cubic voxel is splitted into 6 pyramids with vertex given by the voxel center and base surfaces given by the voxel boundary faces. The neighbouring pyramids of neighbouring voxels are joined together to form octahedron which is then splitted into 4 tetrahedras using diagonals of the voxel boundary face - see Figure 5. In such way we get our *3D tetrahedral grid*. Two nodes of every tetrahedron correspond to centers of neighbouring voxels, and further two nodes correspond to voxel boundary vertices; every tetrahedron intersects common face of neighbouring voxels. Let us note that due to our image processing application in mind, we restrict all further considerations only to this type of grids. In our method, only the centers of voxels will represent degree of freedom nodes (DF nodes), i.e. solving the equation at a new time step we update the segmentation function only in these DF nodes. Additional nodes of tetrahedras will not represent degrees of freedom, we will call them non-degree of freedom nodes (NDF nodes), and they will be used in piecewise linear representation of segmentation function. Let a function u be given by discrete values in the voxel centers, i.e. in DF nodes. Then in 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 tetrahedral grid is built.

For the tetrahedral grid $\mathcal{T}_h$, given by the previous construction, we construct a co-volume (dual) mesh. We modify the approach given in [43, 22] 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 $p = 1, \ldots, M$. Since there will be one-to-one correspondence between co-volumes and DF nodes, without any confusion, we use the same notation for them. 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 planes e_{pq} that bisect and are perpendicular to the edges $\sigma_{pq}, q \in C_p$. By this construction, if e_{pq}

intersects σ_{pq} in its center, the co-volume mesh corresponds exactly to the voxel structure of the image inside the computational domain Ω where the segmentation is provided. Then the co-volume boundary faces do cross in NDF nodes. So we can also say that NDF nodes corresponds to zero-measure co-volumes and thus they do not add additional equations to discrete model (cf. (13)), they do not represent degrees of freedom in the co-volume method. We denote by $\mathcal{E}_{pq}$ the set of tetrahedras having σ_{pq} as an edge. In our situation (see Figure 5) every $\mathcal{E}_{pq}$ cosists of 4 tetrahedras. For each $T \in \mathcal{E}_{pq}$ let c_{pq}^T be the area 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 $\mathbb{R}^{d-1}$. Let $\mathcal{N}_p$ be the set of all tetrahedras that have DF node p as a vertex. Let u_h be a piecewise linear function on $\mathcal{T}_h$. 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 DF node p of $\mathcal{T}_h$.

With these notations, we are ready to derive co-volume spatial discretization. As it is usual in finite volume methods [27, 18, 35], 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" in the scheme. Now the exact "fluxes" $\int_{e_{pq}} \frac{g^0}{|\nabla u^{n-1}|} \frac{\partial u^n}{\partial \nu} ds$ on the right hand side and "capacity function" $\frac{1}{|\nabla u^{n-1}|}$ in the left hand side (see e.g. [27]) 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 tetrahedron $T \in \mathcal{T}_h$, 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 measure in $\mathbb{R}^d$ of co-volume p and M_p is an approximation of the capacity function inside the finite volume p. For that goal we use either the averaging of the gradients proposed by Walkington [43], i.e.

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

or

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

which is close to finite element approximation with the mass lumping. Then the regularization of both approximations of the capacity function is given either by

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

or by

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

and if we define coefficients (where the ε-regularization is taken into account)

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

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

we get from (15)-(16) our

Fully-discrete semi-implicit co-volume scheme: *Let u_p^0, $p = 1, \dots, M$ be given discrete initial values of the segmentation function. Then, for $n = 1, \dots, N$ we look for $u_p^n, p = 1, \dots, M$, 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 $n = 1, \dots, N$. The system matrix is symmetric and diagonally dominant M-matrix. For any $\tau > 0$, $\varepsilon > 0$ the following L_∞ 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

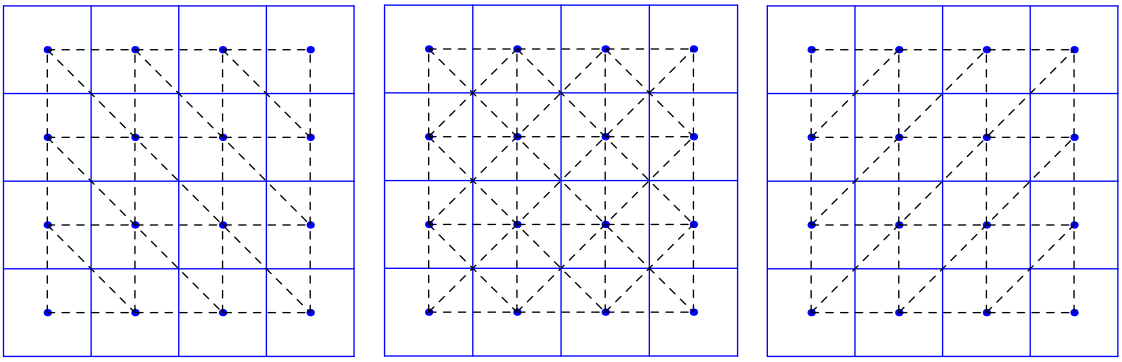

Figure 6: By dashed lines we plot the "left oriented" triangulation (left), "right oriented" triangulation (right) and "symmetric" triangulation corresponding to our method in 2D. We plot also the image pixels (blue solid lines) corresponding to co-volume mesh, and DF nodes (blue round points) corresponding to centers of pixels.

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 $\max(u_1^n, \ldots, u_M^n)$ be achieved in the node p. Then the whole second term on the left hand side is nonnegative and thus $\max(u_1^n, \ldots, u_M^n) = u_p^n \le u_p^{n-1} \le \max(u_1^{n-1}, \ldots, u_M^{n-1})$. 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).

The evaluation of g_T^0 included in coefficients (22) can be done in several ways. First, we may replace the convolution by the weighted average to get $I_\sigma^0 := G_\sigma * I^0$ (see e.g. [30]) and then relate discrete values of I_σ^0 to voxel centers. Then, as above, we may construct its piecewise linear representation on grid and get constant value of $g_T^0 \equiv g(|\nabla I_\sigma^0|)$ on every tetrahedron $T \in \mathcal{T}_h$. Another possibility is to solve numerically linear heat equation for time t corresponding to variance σ with initial datum given by I^0 (see e.g. [4]) by the same method as above. 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 fixed discrete grid (e.g. described above) gives always bounded values. So, working on fixed 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_T^0.

Remark on corresponding discretization in 2D. Previously studied co-volume algorithms [43, 22] for the level-set-like problems have used either "left oriented" or "right oriented" triangulations and no NDF nodes (see Figure 6). But, then the level set curve or surface evolution is influenced by the grid effect. Of course this effect is satisfactory

weakened by refining the grid (e.g. in interface motion computations, cf. [22]). In image processing we work with fixed given pixel/voxel structure, and we do not refine this structure, so we want to remove this "non-symmetry" of the method. This can be done by averaging of two, "left" and "right" solutions, or implicitly by taking the combination of triangulations as plotted in the middle part of Figure 6. Our 3D tetrahedral and co-volume grid constructions transfer such "symetry" of the method into 3D case. In this sense we improve co-volume techniques used for the level set computations.

Usage of such "symmetric" triangulation can be accompanied also by the linear finite element method of Deckelnick and Dziuk [14, 15], considering also NDF nodes as degrees of freedom. But this would increase the number of unknowns by factor 2, which can be critical in case of image processing applications, usually with huge number of pixels/voxels given. Moreover, NDF nodes are not included in the given image intensity information.

Without any construction of 3D tetrahedral grid, we could also use a tri-linear representation of the segmentation function on finite elements corresponding to image voxels and build tensor-product finite element method. But then we would face a problem of non-constant gradients in evaluation of nonlinearities. The same problem would arise considering co-volume method given by mesh corresponding to voxels and by tri-linear representation of the segmentation function on the cubic grid formed by centers of voxels. Again, such technique would require the evaluation and integration of absolute value of gradient of bi-linear functions on the voxel faces. From the above points of view, *our method gives the smallest possible number of unknowns and the most simple (piecewise constant) nonlinear coefficients evaluation.* In this paper we concentrate mainly on algorithmic and computational aspects of the method. The experiments in the next section show the experimental orders of convergence for the method on nontrivial examples of the level set evolution and thus its reliability in simulations. A full theoretical convergence analysis is not a simple task, and it is out of scope of this paper; however, it is a subject of our current research.

Scheme in the finite difference notation. The co-volume scheme introduced in this paper is designed for the specific mesh given by the cubic voxel structure of 3D image. For simplicity of implementation and for reader convenience we will write the co-volume scheme (23) in a "finite-difference notation". As it is usual for 3D rectangular grids, we associate co-volume p and its center (DF node) with a triple (i,j,k), i representing index in x-direction, j in y-direction and k in z-direction. If Ω is a rectangular subdomain of the image domain where n_1, n_2, n_3 are numbers of voxels of Ω in x,y,z-directions and if we consider Dirichlet boundary conditions then $i = 1,\ldots,m_1$, $j = 1,\ldots,m_2$, $k = 1,\ldots,m_3$, $m_1 \le n_1 - 2$, $m_2 \le n_2 - 2$, $m_3 \le n_3 - 2$ and $M = m_1\, m_2\, m_3$. The unknown value u_p^n can then be denoted by $u_{i,j,k}^n$. For every co-volume p, the set $C_p = \{w, e, s, n, b, t\}$ consisting of 6 neighbours, west $u_{i-1,j,k}$, east $u_{i+1,j,k}$, south $u_{i,j-1,k}$, north $u_{i,j+1,k}$, bottom $u_{i,j,k-1}$, top $u_{i,j,k+1}$, and the set $\mathcal{N}_p$ consists of 24 tetrahedras.

In every discrete time step $n = 1,\ldots,N$ and for every i,j,k we compute absolute value of gradient $|\nabla u_T^{n-1}|$ on these 24 tetrahedras. We denote by $G_{i,j,k}^{z,l}, l = 1,\ldots,4, z \in C_p$ the square of the gradient on tetrahedras crossing the west, east, south, north, bottom and top co-volume faces. If we define (omitting upper index $n-1$)

$$\begin{aligned} s_{i,j,k} &= (u_{i,j,k} + u_{i-1,j,k} + u_{i,j-1,k} + u_{i-1,j-1,k} + \\ &+ u_{i,j,k-1} + u_{i-1,j,k-1} + u_{i,j-1,k-1} + u_{i-1,j-1,k-1})/8, \end{aligned}$$

the value at left-south-bottom NDF node of co-volume, then for the west face

$$
\begin{aligned}
G^{w,1}_{i,j,k} &= \left(\frac{u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2+\left(\frac{s_{i,j,k+1}-s_{i,j,k}}{h}\right)^2+\\
&\quad \left(\frac{u_{i,j,k}+u_{i-1,j,k}-s_{i,j,k+1}-s_{i,j,k}}{h}\right)^2\\
G^{w,2}_{i,j,k} &= \left(\frac{u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2+\left(\frac{s_{i,j+1,k+1}-s_{i,j,k+1}}{h}\right)^2+\\
&\quad \left(\frac{s_{i,j+1,k+1}+s_{i,j,k+1}-u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2\\
G^{w,3}_{i,j,k} &= \left(\frac{u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2+\left(\frac{s_{i,j+1,k+1}-s_{i,j+1,k}}{h}\right)^2+\\
&\quad \left(\frac{s_{i,j+1,k+1}+s_{i,j+1,k}-u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2\\
G^{w,4}_{i,j,k} &= \left(\frac{u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2+\left(\frac{s_{i,j+1,k}-s_{i,j,k}}{h}\right)^2+\\
&\quad \left(\frac{u_{i,j,k}+u_{i-1,j,k}-s_{i,j+1,k}-s_{i,j,k}}{h}\right)^2
\end{aligned}
$$

and correspondingly for further co-volume faces.

In the same way, but only once, in the beginning of algorithm we compute values $G^{\sigma,z,l}_{i,j,k}, l=1,\ldots,4, z\in C_p$, changing u by I^0_σ in the previous expressions. Then for every i,j,k we construct (west, east, south, north, bottom and top) coefficients

$$
\begin{aligned}
w_{i,j,k} &= \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,w,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{w,l}_{i,j,k}}}, \quad e_{i,j,k} = \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,e,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{e,l}_{i,j,k}}},\\
s_{i,j,k} &= \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,s,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{s,l}_{i,j,k}}}, \quad n_{i,j,k} = \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,n,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{n,l}_{i,j,k}}},\\
b_{i,j,k} &= \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,b,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{b,l}_{i,j,k}}}, \quad t_{i,j,k} = \tau\frac{1}{4}\sum_{l=1}^{4}\frac{g(\sqrt{G^{\sigma,t,l}_{i,j,k}})}{\sqrt{\varepsilon^2+G^{t,l}_{i,j,k}}},
\end{aligned}
$$

and we use either (cf. (17))

$$
m_{i,j,k} = \frac{1}{\sqrt{\varepsilon^2+\left(\frac{1}{24}\sum\limits_{z\in C_p}\sum\limits_{l=1}^{4}\sqrt{G^{z,l}_{i,j,k}}\right)^2}}
$$

or (cf. (18))

$$
m_{i,j,k} = \frac{1}{24}\sum_{z\in C_p}\sum_{l=1}^{4}\frac{1}{\sqrt{\varepsilon^2+G^{z,l}_{i,j,k}}}
$$

to define diagonal coefficients

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

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

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

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

$$\begin{aligned} c_{i,j,k}u^n_{i,j,k} - w_{i,j,k}u^n_{i-1,j,k} - e_{i,j,k}u^n_{i+1,j,k} - s_{i,j,k}u^n_{i,j-1,k} - \\ n_{i,j,k}u^n_{i,j+1,k} - b_{i,j,k}u^n_{i,j,k-1} - t_{i,j,k}u^n_{i,j,k+1} = r_{i,j,k}. \end{aligned} \tag{28}$$

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

Solution of the linear systems. We can solve the system (28) by any efficient preconditioned linear iterative solver suitable for sparse, symmetric, diagonally dominant M-matrices. If the spectral properties of the matrix are good (they are influenced by a time step τ and regularization parameter ε), then a choice of a stationary iterative method is probably an optimal approach (see e.g. [37]). E.g., if the so-called SOR (Successive Over Relaxation) method is used, then our semi-implicit co-volume method can be implemented in tens of lines. At the nth discrete time step we start the iterations by setting $u^{n(0)}_{i,j,k} = u^{n-1}_{i,j,k}$, $i = 1,\dots,m_1$, $j = 1,\dots,m_2$, $k = 1,\dots,m_3$. Then in every iteration $l = 1,\dots$ and for every $i = 1,\dots,m_1$, $j = 1,\dots,m_2$, $k = 1,\dots,m_3$ the following two step procedure is used:

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

We define squared L_2-norm of residuum at current SOR iteration by

$$\begin{aligned} R^{(l)} = \sum_{i,j,k} (c_{i,j,k}u^{n(l)}_{i,j,k} - w_{i,j,k}u^{n(l)}_{i-1,j,k} - e_{i,j,k}u^{n(l)}_{i+1,j,k} - \\ s_{i,j,k}u^{n(l)}_{i,j-1,k} - n_{i,j,k}u^{n(l)}_{i,j+1,k} b_{i,j,k}u^{n(l)}_{i,j,k-1} - t_{i,j,k}u^{n(l)}_{i,j,k+1} - r_{i,j,k})^2. \end{aligned}$$

The iterative process is stopped if $R^{(l)} < \mathrm{TOL}\ R^{(0)}$. The relaxation parameter ω is chosen by user to improve convergence rate of the method.

Of course, the number of iterations in SOR 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 these dependences, more sofisticated approaches can be recommended. A fast solution of the large linear system (28) can be done by the preconditioned conjugate gradient (PCG) method. Incomplete Cholesky factorization (ICF) is an effective tool to construct efficient preconditioners for symmetric positive definite M-matrices. Satisfactory results have been documented in [22] for solving level-set-like PDEs in 2D image processing and curve evolution. A key issue in ICF is to choose a sparsity pattern S. The first ICF, proposed by Meijerink and van der Vorst [29], kept the sparsity pattern of the original matrix. Another popular ICF method is based on the drop tolerance approach, in which nonzeros are included in the incomplete factor, when they are larger than some threshold parameter. Therefore, the memory requirements are unpredictable. Many variants and detailed descriptions of the algorithms can be found in Saad's book [37] . In [22], we

n	h	Error, $\varepsilon = h^2$	EOC	Error, $\varepsilon = h$	EOC	Error, $\varepsilon = 10^{-6}$	EOC
10	0.25	2.9208e-2		6.5190e-2		2.4936e-2	
20	0.125	5.5133e-3	2.40	2.0893e-2	1.64	5.2251e-3	2.25
40	0.0625	1.3106e-3	2.07	5.4823e-3	1.93	1.2939e-3	2.01
80	0.03125	3.2371e-4	2.01	1.3679e-3	2.00	3.2270e-4	2.00
160	0.015625	8.0548e-5	2.00	3.4086e-4	2.00	8.0485e-5	2.00

Table 1: Errors in $L_\infty((0,T), L_2(\Omega))$-norm, and EOC comparing numerical and exact solution (29).

have implemented the idea of Lin and Moré, that allows memory saving. The incomplete factor is calculated column by column. After a column is obtained, only the p largest (in magnitude) elements are stored back to the factor. The same idea is applied here for solving the system (28). Our experience is that, with the same TOL, the PCG and SOR are similarly fast for time steps in a range $h^2 - 10 \times h^2$, while for a larger time steps the PCG is faster.

The so-called AOS schemes [44] are also often used in image processing applications. They are semi-implicit, and a basic idea is to solve the system (28) not exactly, but by splitting its solution into three substeps and then collecting the partial results in an additive way. The structure of our scheme allows to use also the AOS approach, but, in order not to deal with a splitting error, we do not apply it in this paper.

4 Discussion On Numerical Results

In this section we study experimental order of convergence of the method in case of mean curvature driven level set flow (i.e. for equation (1) with $g \equiv 1$ and $\varepsilon \to 0$), and then we present some 3D segmentation examples. The comparison with nontrivial exact solutions show the second order accuracy for smooth (or mildly singular) solutions and the first order accuracy for highly singular solutions (i.e. when gradient is vanishing on a large subset of a domain and discontinuity set of the gradient field is nontrivial). It means that the method is accurate in computing interface motions, and it is reliable also for computing evolutions including flat regions as arising in subjective surfaces based segmentation method.

4.1 Experimental order of convergence

In the first example we test the method using the exact solution [32]

$$u(x,y,z,t) = (x^2 + y^2 + z^2 - 1)/4 + t \tag{29}$$

of the level set equation

$$u_t = |\nabla u| \nabla . (\frac{\nabla u}{|\nabla u|}) \tag{30}$$

and we consider Dirichlet boundary conditions given by this exact solution.

This problem, and all further computed examples in this subsection, are solved in the spatial domain $\Omega = [-1.25, 1.25]^3$ and in the time interval $T = 0.16$. We have taken subsequent grid refinement with $M = n^3$ co-volumes (DF nodes), $n = 10, 20, 40, 80, 160$.

n	h	Error, $\varepsilon = h^2$	EOC	Error, $\varepsilon = 10^{-6}$
10	0.25	5.2252e-2		0.9114e-6
20	0.125	1.5088e-3	1.79	1.0309e-6
40	0.0625	4.0474e-3	1.89	1.0648e-6
80	0.03125	1.0118e-3	2.00	1.0473e-6

Table 2: Errors in $L_\infty((0,T),L_2(\Omega))$-norm, and EOC comparing numerical and exact solution (32).

So the grid size is $h = 2.5/n$. The time step τ is chosen proportionally to h^2 what is natural for testing the schemes for solving parabolic problems. For the convergence of the iterative scheme we use TOL=10^{-6}, and we measure errors in $L_\infty((0,T),L_2(\Omega))$-norm.

Let us assume that the error of the scheme in some norm is proportial to some power of the grid size, i.e. $Error(h) = Ch^\alpha$, with a constant C. Then halving the grid size we have $Error(h/2) = C(h/2)^\alpha$ from where we can simply extract

$$\alpha = \log_2(Error(h)/Error(h/2)). \tag{31}$$

The α is called the *experimental order of convergence* (EOC) and it can be determined by comparing numerical solutions and exact solution on subsequently refined grids.

In Table 4.1 we report errors in $L_\infty((0,T),L_2(\Omega))$-norm for refined grids and for several choices of ε. In all cases we observe $\alpha = 2$, the coupling $\varepsilon \approx h^2$ seems optimal (see also other examples), but as one can see, choosing ε even smaller can get even smaller errors on coarse grids.

Next interesting example comes from [15] and is given by (see Figure 7)

$$\begin{aligned} u(x,y,z,t) &= x + 0.5,\ x \le -0.5 \\ &= 0,\ -0.5 \le x \le 0.5 \\ &= x - 0.5,\ x \ge 0.5 \end{aligned} \tag{32}$$

The level sets of u are planes, with zero mean curvature, so the solution remains unaltered by the flow. There is also a large part of the solution where gradient is vanishing, however the set of discontinuity in gradient is relatively simple (two planes) regarding to a possible orientation of our grid (we can make it parallel). Mathematically, it is a trivial example, but it is a good test for a numerical scheme. One can observe (see Table 4.1) that the error of the scheme in this example is proportional just to regularization parameter ε, and thus it can be made as small as desirable. This is a simple consequence of the *consistency* of our scheme in the sense, that it gives exact solution for any linear initial function and for any choice of regularization parameter ε, any grid size h and any size of time step τ. Such property can be checked by inspection of the scheme (23). For the linear function, the gradients in all tetrahedras are the same, so we get the system with the same structure as given by the backward Euler scheme for solving linear diffusion equation with constant coefficients, and such scheme does not alter a steady state.

In the next example we compare our numerical solution with the highly singular solution given by

$$u(x,y,z,t) = \min((x^2+y^2+z^2-1)/4 + t, 0). \tag{33}$$

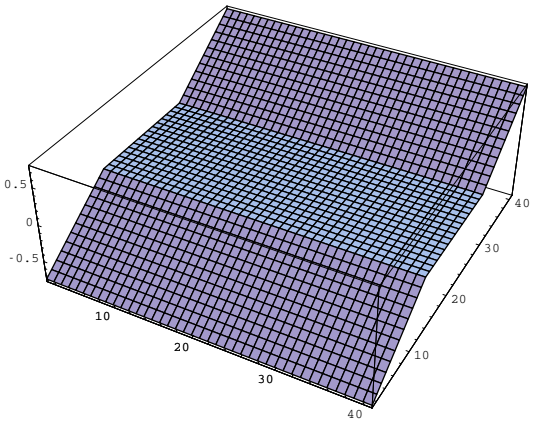

Figure 7: Exact solution (32) remaining unchanged under mean curvature flow (plot at $z = 0$).

n	h	Error, $\varepsilon = h^2$	EOC
10	0.25	6.9571e-2	
20	0.125	4.2686e-2	0.70
40	0.0625	2.2049e-2	0.95
80	0.03125	1.1030e-2	0.99
160	0.015625	5.5544e-3	0.99

Table 3: Errors in $L_\infty((0,T), L_2(\Omega))$-norm, and EOC comparing numerical and exact solution (33).

The initial function and numerical result at $T = 0.16$ are plotted in Figure 8. We see only slight smoothing in numerical solution along singularity, numerical solution converges to viscosity solution with $\alpha = 1$.

In the last example we test experimental order of convergence comparing the numerical and exact evolution of one particular level set. Namely, we use the exact solution given by shrinking sphere with exact radius $r(t) = \sqrt{r(0) - 4t}$, starting with $r(0) = 1$. Our initial level set function is a 3D cone (signed distance function) having zero value on the unit sphere. We consider zero Neumann boundary conditions, so we do not know the exact solution for evolution of the whole level set function. At every time step we evaluate $L_2(S^2)$-norm of error, where S^2 is the unit sphere, comparing the exact shrinking sphere and the numerical zero level set and then we take $L_\infty((0,T), L_2(S^2))$-norm for the overall error in time interval $T = 0.16$. Figure 9 represents the initial sphere and the numerical zero level set at $T = 0.16$. Table 4.1 documents the higher order rate of convergence, where again the coupling $\varepsilon = h^2$ is used. Similarly to the first example, different couplings or choice of a fixed small ε give the similar errors and convergence rates.

4.2 Segmentation examples

Here we present two 3D segmentation examples. The first one is simple, in the image with resolution 40^3 voxels is an object - cube with two holes (see Figure 10 left), the holes are in front and back faces. Figure 10 gives reconstructed surface with the hole filling.

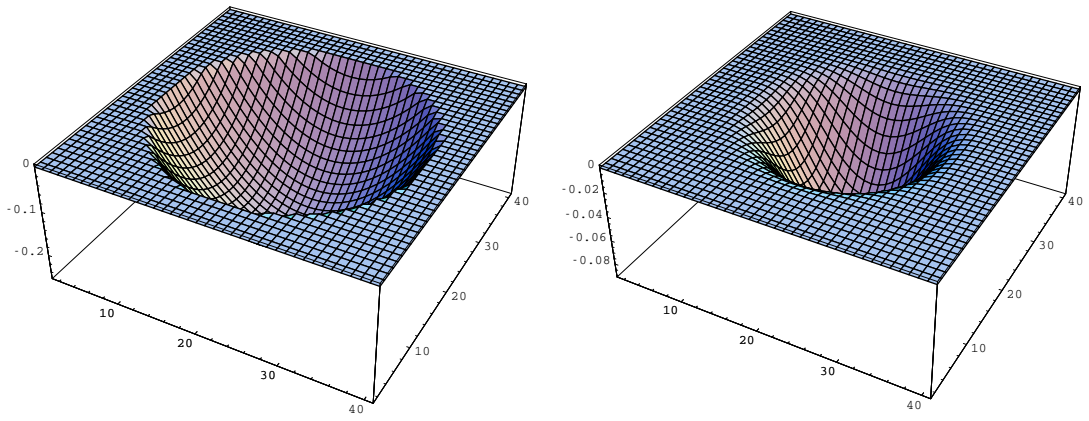

Figure 8: Numerical solution corresponding to exact solution (33) plotted at times $t = 0$ (left) and $t = 0.16$ and for $z = 0$.

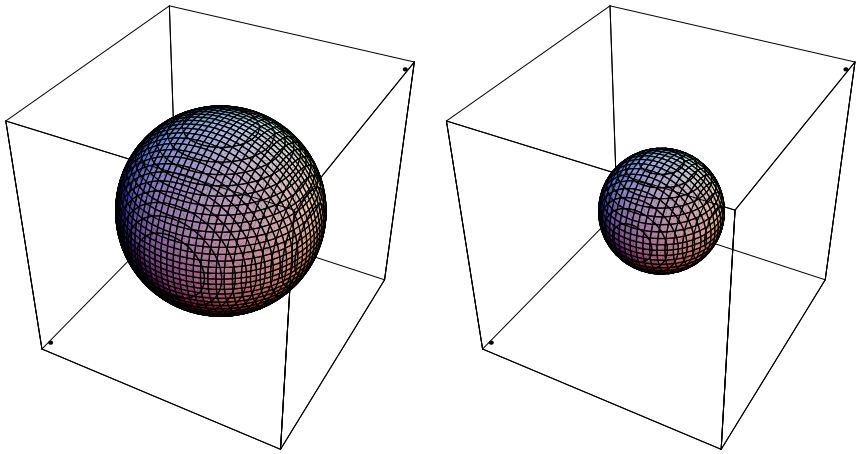

Figure 9: Numerical solution corresponding to exact shrinking sphere with radius $r(t) = \sqrt{r(0) - 4t}$, $r(0) = 1$, plotted at times $t = 0$ (left) and $t = 0.16$.

n	h	Error, $\varepsilon = h^2$	EOC
10	0.25	7.7228e-2	
20	0.125	2.8732e-2	1.42
40	0.0625	7.8269e-3	1.87
80	0.03125	4.7901e-4	4.03
160	0.015625	9.9568e-5	2.26

Table 4: Errors in $L_\infty((0,T), L_2(S^2))$-norm, and EOC comparing numerical zero level sets and the exact shrinking sphere.

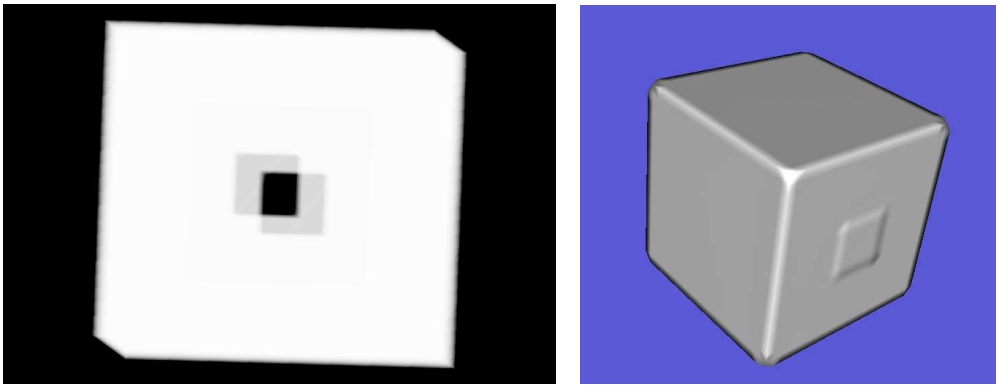

Figure 10: Segmentation (right) of a 3D cube with two holes (left).

Segmentation of the object of this size takes few seconds (on 2.4GHz PC). Usually we take $K = 1$, $TOL = 0.01$ and $h = \frac{1}{n_1}$. Then the optimal time step for semi-implicit scheme is around $10 \times h^2$, and one needs few tens of time steps to find the result (one step is about 0.15 sec). In this example the maximum of the initial segmentation function, cf. setion 2, is taken in the image center.

The second example is given by 3D echocardiography of the size $81 \times 87 \times 166$. As one can see from volume rendering visualization (Figure 11 left), the 3D image is very noisy, however the surface of the ventricle is observable. How noisy is the image intensity, can be seen also from Figure 12 where we plot intensity and its graph in one 2D slice. Now we start the segmentation process with few "points of view" inside the object of interest, i.e. the initial segmentation function has several maxima, and let it evolve until the difference in L_2-norm of two subsequent time steps is less than prescribed threshold. Then we look to a 2D slice with relatively good boundary edges (Figure 13), where we see accumulation of level sets along the inner boundary of the ventricle (Figure 13 left). The largest gap in the histogram (Figure 13 middle) indicates the shock in the segmentation function, so we choose one level inside the gap, and plot it inside the slice (Figure 13 right). We can check how this level set looks like in other noisy slices (Figure 14), and then make a 3D isosurface visualization (Figure 11 right) which gives realistic representation of the left ventricle.

The computation of one time step takes 3 seconds on 2.4GHz one processor PC and it takes 200 steps to finish segmentation, so the overal computing time for this size of images is in a range of few minutes. The MPI parallel implementation [3] of our 3D segmentation scheme has been built under the support of the Project HPC-EUROPA at CINECA SuperComputing Center, Bologna, so the segmentation time is even speeding up in dependence on the number of available processors (e.g. in Linux cluster). The MPI parallelization of our algorithm is straightforward, because it uses classical structures of scientific computing methodology for solving partial differential equations and linear algebra problems (see e.g. [3]).

Acknowledgement. This work was supported by NATO Collaborative Linkage Grant No. PST.CLG.979123, by grants VEGA 1/0313/03 and APVT-20-040902. We thank to Roberto Gori from CINECA for help with 3D visualizations.

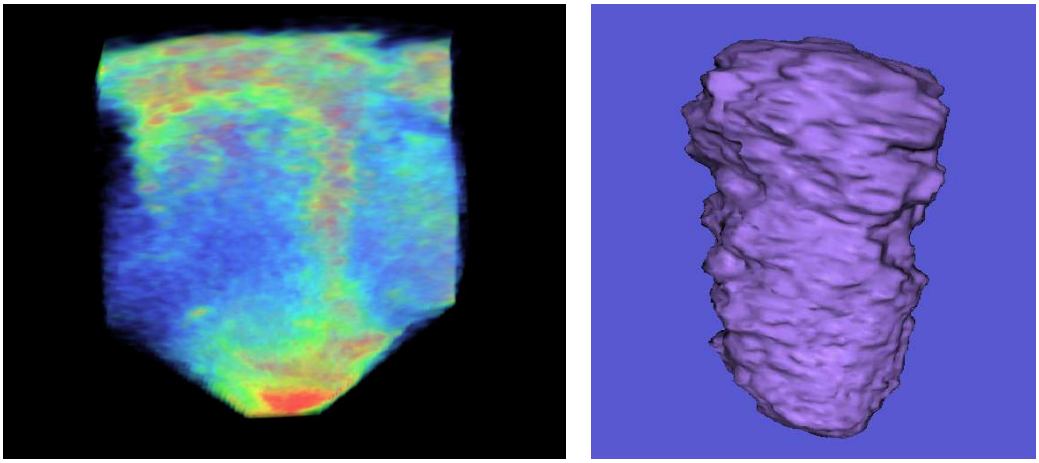

Figure 11: Volume rendering of original 3D data set (left) and segmentation of the ventricle (right).

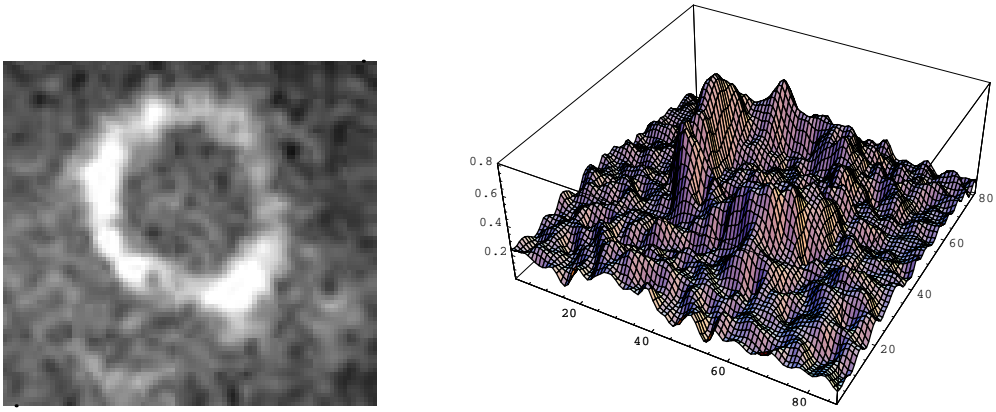

Figure 12: Plot of image intensity in the slice $k = 130$ (left), and its 3D graphical view (right).

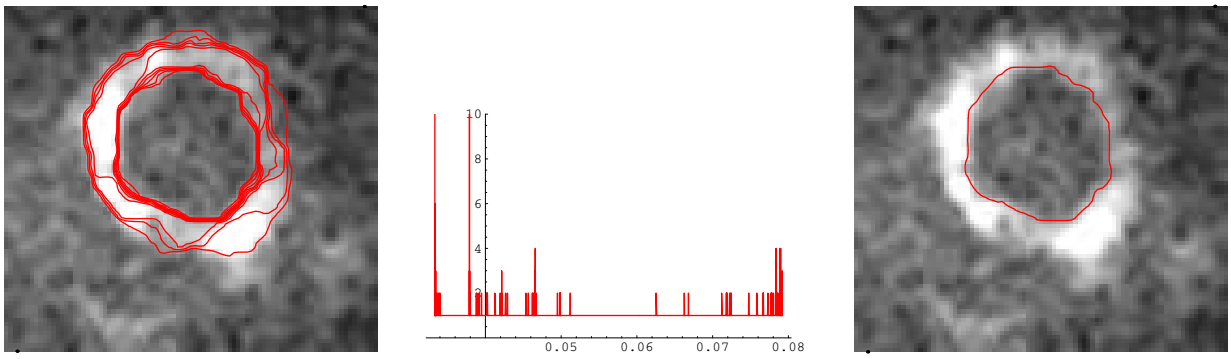

Figure 13: Plot of accumulated level sets in the slice $k = 130$ (left); the histogram of the segmentation function in this slice (middle); image intensity with level set $u = 0.052$ (right). Visualization of 3D surface in Figure 11 is done with the same level set.

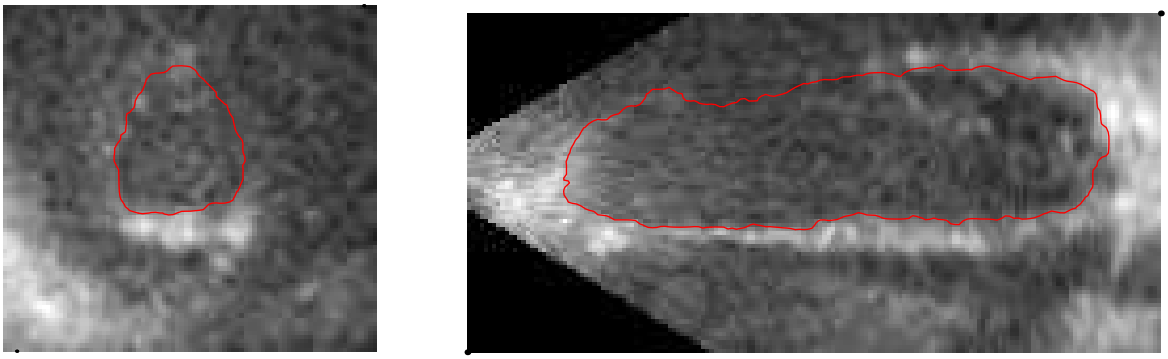

Figure 14: Plot of image intensity together with level line 0.052 in two other slices $k = 100$ (left), and $j = 40$ (right).

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.

[3] Aoyama, Y., Nakano, J., RS/6000 SP: Practical MPI Programming, IBM, 1999, www.redbooks.ibm.com

[4] 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.

[5] Brenner, S.C., Scott, L.R., The mathematical theory of finite element method, Springer, New York, 2002.

[6] 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.

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

[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.

[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., Numerical approximation of mean curvature flow of graphs and level sets, in Mathematical aspects of evolving interfaces. (L. Ambrosio, K. Deckelnick, G. Dziuk, M. Mimura, V. A. Solonnikov, H. M. Soner, eds.), pp 53-87, Springer, Berlin-Heidelberg-New York, 2003.

[16] Deschamps, T., Malladi, R., Rawe, I., Fast evolution of image manifolds and application to filtering and segmentation in 3D medical images, IEEE Transactions on Visulization and Computer Graphics, Vol. 10, No. 5, pp. 525-535, 2004.

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

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] Kanizsa, G., Organization in Vision, Praeger, New York, 1979.

[25] 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.

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

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

[28] 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.

[29] Meijerink, J.A., van der Vorst, H.A., An iterative solution method for systems of which the coefficient matrix is a symmetric M-matrix, Math. Comp., Vol.31, pp.148-162, 1997.

[30] 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.

[31] Mikula, K. and Ševčovič, D., Computational and qualitative aspects of evolution of curves driven by curvature and external force, Computing and Visualization in Science, Vol. 6, No. 4, pp. 211-225, 2004.

[32] Oberman, A.M., A convergent monotone difference scheme for motion of level sets by mean curvature, Numer. Math., DOI: 10.1007/s00211-004-0566-1, 2004.

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

[34] 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.

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

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Thomée, V., Galerkin finite element methods for parabolic problems, Springer, Berlin, 1997.

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

[44] M Weickert, J., Romeny, B.M.t.H., Viergever, M.A., Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Trans. Image Processing, Vol. 7, pp. 398-410, 1998.

[45] Zhu, W., Chan, T., Capture Illusory Contours: A Level Set Based Approach, CAM Report 03-65, UCLA, 2003.