Proceedings of ALGORITMY 2012
pp. 416–426

PARALLEL ALGORITHMS FOR SEGMENTATION OF CELLULAR STRUCTURES IN 2D+TIME AND 3D MORPHOGENESIS DATA*

KAROL MIKULA, MICHAL SMÍŠEK AND RÓBERT ŠPIR †

Abstract. In this paper we present a robust method to segment 3D and 2D+time biological data. Our method uses distance function and the Generalized Subjective Surfaces (GSUBSURF) level-set segmentation algorithm. We focus on memory-efficient implementation of distance function computation, a novel way of finite volume method discretization of GSUBSURF PDE and the optimized parallelization of our code via the MPI library, enabling us to take the advantage of high-performance computing servers. Several experiments, including optimized parallelization test, validation of parallel GSUBSURF implementation and visual check of method performance on real data, are presented as well.

Key words. distance function, level-set segmentation, biological image analysis, parallelization, high-perfomance computing

AMS subject classifications. 35K65, 35L60, 65M06, 65M08, 65Y05

1. Introduction and problem definition. In this paper we present algorithms for solution of two problems: First of them is to segment tubular structures of cell evolution from a 2D video of a fruit fly (*Drosophila*) in pupal state. Our second task is, from a 3D image of cells of a zebrafish (*Danio Rerio*) in embryogenesis, to extract the volume and shape of the whole embryo. Although these tasks are different in nature, technically the solutions arising for both of them tend to be very similar.

The similarity of these problems is caused by treating the temporal dimension in the 2D+t problem as a third, artificial, spatial dimension. Then we can use the same algorithms in 3D domain to solve these problems.

The input to both our problems is the intensity function in defined the domain Ω and the set of points denoting cell positions. We call them the cell identifiers. The input intensity function over the domain is a 2D video of cellular evolution in the first problem and a 3D image of cell membranes in the second problem. Same holds for the dimensionality of cell identifier sets. The visualization of intensity functions is to be seen in fig. 1.1.

The suggested algorithms take into account the intrinsic noise introduced by the confocal laser microscope imaging technology and their goal is to be robust against imaging imperfections. Therefore, we accept that some of the cell identifiers could be missing, and some cells could have more than one cell identifier. We obtain cell identifiers directly from our data using the level-set center detection (LSCD) algorithm [10]. The aim of LSCD algorithm is to yield a set of points identifying the approximate centers of mass of each cell. From the segmentation point of view, the cell identifiers are to be used as seeds to create the initial segmentation profile.

The output to both problems is the segmentation of the domain into logical subdomains, representing cell volumes. In the first case, it is the segmentation of 2D+t

*This work was supported by Grant No.: APVV-0184-10

†Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology, Radlinskeho 11, 813 68 Bratislava, Slovakia (mikula@math.sk, smisek@math.sk, spir@math.sk).

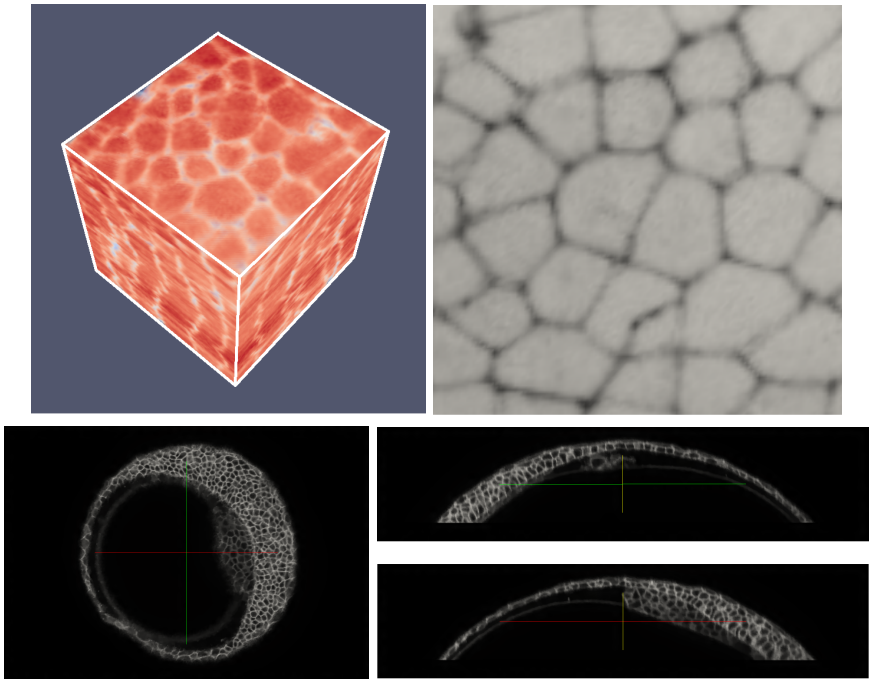

FIG. 1.1. *Data example. Upper left - 100 frames of a 100x100 pixel video of drosophila evolution stacked atop each other, upper right - 50th frame of drosophila evolution video. Lower left - slice of 512x512x190 voxel 3D data of zebrafish embryo, view from top, lower right - slices of zebrafish data, view from sides.*

domain into spatio-temporal pair-of-pants-like structures representing the evolution of cells in time. This segmentation is to be processed further, to obtain 2D+t trajectories of cells by a backward tracking algorithm as in [2, 12], in order to obtain the cell lineage. In the second case, the output is the segmentation of 3D domain representing the shape of the embryo. We use this information further to calculate volume of the embryo and some other characteristics, like the density of cells per unit volume or the density of cell divisions [3, 13].

In comparison with the previous papers [2, 3, 13], we present here the parallel strategy to solve these tasks. Our proposed parallel algorithmic approach consists of these steps:

A. compute distance function (DF) from input set of points by an efficient parallel algorithm,
B. construct in parallel the initial segmentation function (ISF) from DF,
C. use a parallel level-set algorithm, in our case the Generalized Subjective Surfaces (GSUBSURF) to obtain the segmented domain.

The organization of this paper is following: in chapters 2 and 3, we discuss the design and implementation of these algorithmic steps. In chapter 4, we describe parallelization of the code, using MPI. And finally, in chapter 5, we discuss numerical results in the study of the experimental order of convergence, and the experiments performed on phantom and real data.

2. Distance function. To compute the distance function from cell identifiers, we solve the time relaxed Eikonal equation

$$d_t + |\nabla d| = 1\,, \tag{2.1}$$

using the method from [4, 13], based on the so-called Rouy-Tourin scheme [14].

Let Ω be the solution domain and let p be a voxel in this domain: $p \subset \Omega$. Let the set of 6 neighbouring voxels to the voxel p, i.e. voxels sharing common face, be $N_p = \{e, n, t, w, s, b\}$ according to "east - north - top - west - south - bottom" points-of-compass labeling. We denote a particular neighbour of p by q. Let the voxels of our domain be uniform cubes. Let the length of its side be h_D. For each p, let the approximate value of the solution d at time step n, in the center of p, be d_p^n. Let us define M_p^q, $q \in N_p$, as [14]

$$M_p^q = \left(\min(d_q^n - d_p^n, 0)\right)^2 .$$

The scheme for solving the eq. (2.1) in the three-dimensional space then reads as follows:

$$d_p^{n+1} = d_p^n + \tau_D - \frac{\tau_D}{h_D}\left(\max\left(M_p^e, M_p^w\right) + \max\left(M_p^n, M_p^s\right) + \max\left(M_p^t, M_p^b\right)\right)^{1/2} \tag{2.2}$$

where τ_D is the time step size. Scheme is stable for $\tau_D \leq h_D/2$ [13].

To compute the distance function with optimal memory requirements, we calculate new time steps in a chessboard-like manner (often called the "Red-Black Scheme" in numerical mathematics) where all red elements have only black neighbours and all black elements have only red neighbours. In each time step we first update all red elements and then all black elements. We can redefine M_p^q for red and black elements as

$$\begin{aligned} M_{R,p}^q &= \left(\min(d_{B,q}^n - d_{R,p}^n, 0)\right)^2 \\ M_{B,p}^q &= \left(\min(d_{R,q}^{n+1} - d_{B,p}^n, 0)\right)^2 , \end{aligned}$$

and the scheme (2.2) has the form of a two-step procedure

$$\begin{aligned} &d_{R,p}^{n+1} = d_{R,p}^n + \tau_D - \\ &\frac{\tau_D}{h_D}\left(\max\left(M_{R,p}^e, M_{R,p}^w\right) + \max\left(M_{R,p}^n, M_{R,p}^s\right) + \max\left(M_{R,p}^t, M_{R,p}^b\right)\right)^{1/2} , \\ &d_{B,p}^{n+1} = d_{B,p}^n + \tau_D - \\ &\frac{\tau_D}{h_D}\left(\max\left(M_{B,p}^e, M_{B,p}^w\right) + \max\left(M_{B,p}^n, M_{B,p}^s\right) + \max\left(M_{B,p}^t, M_{B,p}^b\right)\right)^{1/2} , \end{aligned}$$

where indices B and R denote black or red elements. Using this approach we can use only one array and overwrite it directly by new iterations as they come, thus reducing memory requirements by half.

3. Generalized Subjective Surfaces. The partial differential equation for GSUBSURF method is

$$u_t - w_a \nabla g \cdot \nabla u - w_c g |\nabla u| \nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) = 0\,, \tag{3.1}$$

and comes from the level set formulation of the geodesic active contour model [5, 11, 15, 6, 13].

In the model (3.1), g is an edge detector function, for which we use $g(s) = \frac{1}{1+Ks^2}$, where K is the edge detection sensibility parameter and $s = |\nabla I|$, where I is the input image intensity function. Parameters w_a and w_c are weights for the advection and curvature terms of the model, respectively. The choice of a boundary condition depends on the problem solved: In our first problem, where we reconstruct the 2D+t structures, we assume the cell image continues even beyond the image domain, so we use zero Neumann boundary condition. For our second problem, where the 3D embryo is well separated from the boundary of the image domain, we choose zero Dirichlet boundary condition. We create the initial segmentation function (ISF) from the DF $d(x)$ computed by the method from section 2, where $d(x)$ represents distance in R^3 to a set of points representing cell identifiers. We can use an isosurface $\delta > 0$ of DF and have a piecewise constant ISF:

$$u_0(x) = \begin{cases} 1 & \text{if } d(x) \leq \delta, \\ 0 & \text{else,} \end{cases}$$

or, we can compute ISF by considering

$$u_0(x) = \frac{1}{1 + d(x)}.$$

The reason to compute ISF from the DF, and thus placing the computational burden on the DF itself, is to minimize amount of inter-process communication. This makes it more suitable for parallel computing. In the former, non-parallel approach[13], the communication had to take place for each voxel and for each cell identifier. Now, when the DF is known, there is no inter-process communication and the formulation of the ISF is a local problem.

In order to derive the discretization of (3.1), we use the semi-implicit finite volume method (FVM) based on [9] for advective part, and on [8] for the curvature part. Image voxel serves as a natural choice for FVM control volume.

Similarly to the previous section, let p be the voxel of our interest and $N_p = \{e, n, t, w, s, b\}$ be the set of neighbours in the points-of-compass labeling. Let u_p^n be the value of the solution u in voxel p at the time step n.

The voxel is a box in R^3 and we assume its uniform edge size h. The voxel faces are denoted by σ and their measure is $m(\sigma) = h^2$. Let ∂p denote the boundary of this domain and let $\mathbf{n}_{\partial p}$ denote the outer normal to this boundary. Volume of the voxel is denoted by $m(p) = h^3$. Let σ_{pq} be the common boundary of p and q and let the line connecting the center of voxel p with the center of the face σ be denoted by $d_{p\sigma}$. Its measure is $h/2$.

First, we approximate the time derivative by the backward difference: $u_t \approx \frac{u^{n+1}-u^n}{\tau}$, where τ is the size of the time step. Then, let us use the semi-implicit approach in time, and integrate (3.1) over the finite volume p:

$$\int_p \frac{u^{n+1} - u^n}{\tau} dx - \int_p w_a \nabla g \cdot \nabla u^{n+1} dx - \int_p w_c g |\nabla u^n| \nabla \cdot \left(\frac{\nabla u^{n+1}}{|\nabla u^n|} \right) dx = 0. \quad (3.2)$$

The approximation of the first term in eq. (3.2) is

$$\int_p \frac{u^{n+1} - u^n}{\tau} dx \approx m(p) \frac{u_p^{n+1} - u_p^n}{\tau}. \quad (3.3)$$

To disretize the second term in eq. (3.2), we first define velocity $\mathbf{v} = -w_a \nabla g$. Then,

$$\int\limits_p -w_a \nabla g \cdot \nabla u^{n+1} dx = \int\limits_p \mathbf{v} \cdot \nabla u^{n+1} dx = \int\limits_p \nabla \cdot (\mathbf{v} u^{n+1}) dx - \int\limits_p u^{n+1} \nabla \cdot \mathbf{v} \; dx \,,$$

and, using Green's theorem in both terms and the constant representation of a solution in the finite volume p in the second term, we obtain

$$\int\limits_p \nabla \cdot (\mathbf{v} u^{n+1}) dx - \int\limits_p u^{n+1} \nabla \cdot \mathbf{v} \; dx \approx \int\limits_{\partial p} (\mathbf{v} u^{n+1}) \cdot \mathbf{n}_{\partial p} dx - u_p^{n+1} \int\limits_{\partial p} \mathbf{v} \cdot \mathbf{n}_{\partial p} dx \,. \quad (3.4)$$

We define an approximate gradient of the edge detector function in finite volume p using central differences: $\nabla g_p = (G_{pe}, G_{pn}, G_{pt}) = (-G_{pw}, -G_{ps}, -G_{pb})$, where

$$\begin{aligned} -G_{pw} = G_{pe} &\approx \frac{g_e - g_w}{2h} \,, \\ -G_{ps} = G_{pn} &\approx \frac{g_n - g_s}{2h} \,, \\ -G_{pb} = G_{pt} &\approx \frac{g_t - g_b}{2h} \,, \end{aligned}$$

and g_q is the value of the edge detector function in $q \in N_p$. Then, we define the integrated flux through a voxel side by

$$v_{pq} = \int\limits_{\sigma_{pq}} \mathbf{v} \cdot \mathbf{n}_{\partial p} \; dS = \int\limits_{\sigma_{pq}} -w_a \nabla g \cdot \mathbf{n}_{\partial p} \; dS \approx -w_a G_{pq} m(\sigma_{pq}) \,.$$

Using the integral fluxes, we can define inflows and outflows through voxel sides as

$$v_{pq}^{in} = \min(v_{pq}, 0) \;\; , \;\; v_{pq}^{out} = \max(v_{pq}, 0) \,.$$

We then approximate (3.4) by using the upwind principle as in [9] and obtain the result:

$$\begin{aligned} &\int\limits_{\partial p} (\mathbf{v} u^{n+1}) \cdot \mathbf{n}_{\partial p} dx - u_p^{n+1} \int\limits_{\partial p} \mathbf{v} \cdot \mathbf{n}_{\partial p} dx \approx \\ &\sum_{q \in N_p} v_{pq}^{in} u_q^{n+1} + \sum_{q \in N_p} v_{pq}^{out} u_p^{n+1} - \sum_{q \in N_p} v_{pq}^{in} u_p^{n+1} - \sum_{q \in N_p} v_{pq}^{out} u_p^{n+1} = \\ &\sum_{q \in N_p} v_{pq}^{in} \left(u_q^{n+1} - u_p^{n+1} \right) \,. \end{aligned}$$

To discretize the third term in eq. (3.2), we use the approach built in [8] for approximation of the mean curvature term and we get

$$\begin{aligned} &- \int\limits_p w_c g_p |\nabla u_p^n|_\varepsilon \nabla \cdot \left(\frac{\nabla u_p^{n+1}}{|\nabla u_p^n|_\varepsilon} \right) dx \approx \\ &\approx - w_c g_p |\nabla u_p^n|_\varepsilon \sum_{q \in N_p} \frac{u_q^{n+1} - u_p^{n+1}}{h} \; \frac{2}{|\nabla u_p^n|_\varepsilon + |\nabla u_q^n|_\varepsilon} \; m(\sigma_{pq}) \,, \end{aligned}$$

where we use the Evans-Spruck regularization [7]. Let f_p^n be the ε-regularized approximation of gradient modulus, $f_p^n \approx |\nabla u_p^n|_\varepsilon$, given by

$$|\nabla u_p|_\varepsilon^n \approx f_p^n = \sqrt{\varepsilon^2 + \frac{1}{m(p)} \sum_{q \in N_p} \frac{m(\sigma_{pq})}{d_{p\sigma}} \left(u_\sigma^n - u_p^n\right)^2}, \tag{3.5}$$

which gives the exact value if u is a linear function and $\varepsilon = 0$ [8]. In (3.5), u_σ^n is the approximate value of u in point x_σ, which is the intersection of σ_{pq} and a line connecting centers of voxels p and q, at the time step n. It is computed as follows [8]:

$$u_\sigma^{n+1} = \frac{u_p^{n+1} f_q^n + u_q^{n+1} f_p^n}{f_p^n + f_q^n}\,. \tag{3.6}$$

The full linear system reads as follows:

$$\begin{aligned} m(p)\frac{u_p^{n+1} - u_p^n}{\tau} + \sum_{q \in N_p} v_{pq}^{in} \left(u_q^{n+1} - u_p^{n+1}\right) - \\ - w_c g_p f_p^n \sum_{q \in N_p} \frac{u_q^{n+1} - u_p^{n+1}}{h} \, \frac{2\, m(\sigma)}{f_p^n + f_q^n} = 0 \end{aligned} \tag{3.7}$$

From the initial condition, one can obtain u_p^0 and u_σ^0. We set $n = 0$ and f_p^n is computed by (3.5). Then, the system (3.7) is solved, and new u_σ^{n+1} are computed by (3.6).

4. Parallelization. To speed up the computation and reduce the process memory consumption we parallelize the algorithms using MPI (Message-passing interface) [1]. We divide the whole volume along the volume edge into N subvolumes, the same number as total count of parallel processes. Of these N subvolumes, N-1 has always the same size and the last subvolume may be the same size or smaller, depending on how the volume can be divided. Each process then performs the calculation only on appropriate subvolume. The only inter-process communication is needed at the following steps of parallel implementation: at the beginning, the cell identifiers are read by first process and then broadcasted to all other processes. The neighbouring slices with updated data are exchanged between processes during the computation of DF and also during the solution of the linear system after each half-iteration of the red-black SOR algorithm. The appropriate part of the input image is read from input file in parallel by each process and communication is not needed.

We tested the parallel computation speedup on a high-performance computing (HPC) server with 4x octo-core AMD Opteron 6134 with 256GB of RAM using real data volume with dimensions 512x512x120 voxels. We achieved nearly linear speedup while running up to eight processes and slightly lower speedup after that (tab. 4.1). During the initial testing we observed a curious phenomenon where there was a massive slowdown when the program was run on certain number of processors. Interestingly, this problem was dependent not on the number of processors, but on the number of the subvolume slices. We verified this fact by running the calculation on fourteen processes with 37, 38 or 39 slices on each subvolume. Then the calculation took 3223, 2815 and 2664 seconds respectively, see also peaks in the upper curve in fig. 4.1. Logically, the calculation should be faster on smaller subvolumes, but on

TABLE 4.1
Parallel calculation speedup.

No. of processes	1	2	4	8	16	32
Calculation time	26421	13420	6667	3417	2078	1474
Speedup	-	1.97	2.01	1.95	1.64	1.40

the contrary, it was slower. Depending on the number of slices, arrays of variables in algorithms stored in memory are not optimally aligned for the processor caches, so with certain sizes there is heavy cache trashing and calculation is slowed down significantly. To overcome this problem, we restructure the program memory organization so when the variables are used together, we are not using separate arrays for each of them, but we store them in a single array of structures. Then they are stored near each other in memory. For example, when we are calculating u_σ for north, south, west, east, bottom and top voxel faces, we store them all in single structure containing 6 double variables for each voxel and we are using single array of these structures for whole image. Likewise we are storing other variables in our programs. With this memory organization the hardware data prefetching on processors works in an optimal way. This change not only eliminates the aforementioned problem, but also speeds up the serial code by 40% as seen on fig. 4.1 (starting points of the curves). The slight slowdowns in optimized code after multiples of eight processes are caused by the fact, that each four cores of the server are sharing memory access. If we run eight processes, each process has dedicated memory access, but when we run nine processes, two processes are sharing memory access, so there is slight slowdown. In the tab. 4.1 we can see that we achieved nearly linear speedup while running up to eight processes. After that, the main bottleneck in the calculation is the speed of memory access, so the speedup is lower. We verified this by running the calculation with sixteen processes on two servers, eight processes on each. With this configuration the memory access by processes is used optimally. The calculation time, despite the need of inter-process communication through local network, was faster than running sixteen processes on a single server, where each two processes share single memory access (1814 seconds versus 2078 seconds).

To minimize the amount of data needed for communication, we first rotate the volume so it is always divided along the longest edge - then the amount of data sent during the communication is minimal. This does not bring a notable speedup when running the calculation on single server, because communication through shared memory inside the server is sufficiently fast even for larger amount of data. On the other hand, in the worst-case scenario, when each process runs on a different server connected together by 1Gbit local area network, the speedup was considerable. With our testing volume, when divided along the edge with the size of 120 voxels, we are sending slice with the size of 512x512 type double variables (2MB of data). The calculation on eight processors took 5520 seconds. When we divided it along the 512 voxels edge, the amount of data for communication was reduced to 0.5MB and whole calculation took 3750 seconds, which is faster by 33%.

5. Numerical experiments.

5.1. Phantom experiments. To perform the basic correctness test of our implementations, we created simple phantom data. For the first problem, we created an artificial image of the evolution of three cells, one of which undergoes a cell division

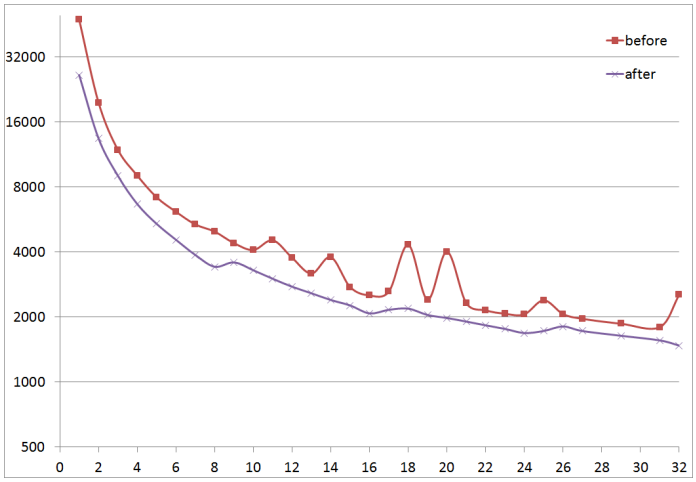

FIG. 4.1. *Comparison of duration of the calculation before and after the optimization depending on the number of used processors. X-axis: number of processors, y-axis: time in seconds, note that y-axis has logarithmic scale. The upper curve represents calculation time before optimizations and the lower curve represents time after the optimization.*

in the middle of the video. Then, using randomly generated cell identifiers, we constructed ISF. Visually we can see that the segmentation algorithm correctly retrieves spatiotemporal shapes of cellular evolution, cf. fig. 5.1.

For the second problem, we created an artificial hemisphere as an approximation of the embryo shape. From randomly placed cell identifiers we computed DF, constructed the ISF and we were able to reconstruct the embryo phantom using the GSUBSURF algorithm, cf. fig. 5.2.

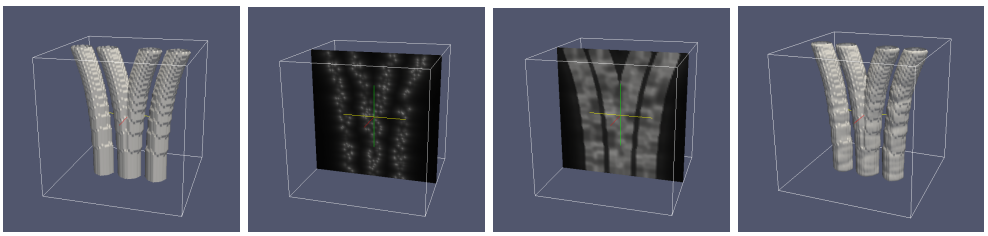

FIG. 5.1. *2D+time phantom image segmentation. From left to right: first - phantom image of three cells, where the central one undergoes a cell division during its evolution, second - hyperbolic initial segmentation function, visualized as a slice, third - segmentation result after 1000 GSUBSURF time steps, visualized as a slice, fourth - the same segmentation result, visualized as an isosurface.*

5.2. Study of the experimental order of convergence. We tested our parallel implementation of (3.5)-(3.7) for the curvature part of GSUBSURF, so we let $w_a = 0$ and $w_c = 1$ and $g = 1$ and we considered the exact solution of the problem in the form [6]

$$u(x, y, z, t) = \left(x^2 + y^2 + z^2 - 1\right)/4 + t\,. \tag{5.1}$$

We solve the problem in the spatial domain $\Omega = [-1.25, 1.25] \times [-1.25, 1.25] \times$

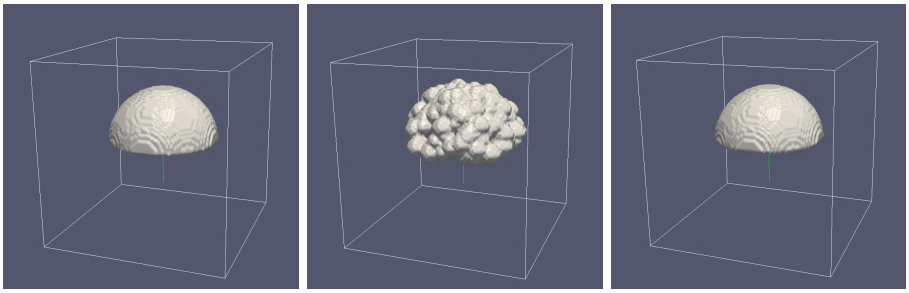

FIG. 5.2. *3D phantom image segmentation. Left - phantom image of an embryo of hemispherical shape, center - initial segmentation function, piecewise-constant thresholding of DF from randomly generated cell identifiers, right - segmentation result*

TABLE 5.1
This table shows EOC analysis of full domain solution. NTS-number of time steps, err - error in $L_2(\Omega)$-norm at the time $T = 0.16$.

n	h	NTS	τ	err	EOC
10	0.25	4	0.04	0.00582	
20	0.125	16	0.01	0.001612	1.85
40	0.06125	64	0.0025	0.000445	1.86
80	0.030625	256	0.000625	0.000118	1.92
160	0.015625	1024	0.00015626	0.00003	1.98

$[-1.25, 1.25]$ over the time interval $T \in [0, 0.16]$. We divide the domain subsequently into n^3 finite volumes, where $n = 10, 20, 40, 80, 160$ with edge size $h = 2.5/n$. The length of time step τ is proportional to h^2.

We measured the experimental order of convergence (EOC) with respect to grid refinement. The results of parallel computations are shown in tab. 5.1. The obtained results are independent on the number of processors used for calculations.

5.3. Real data experiment. Finally, we performed a segmentation of real data. In fig. 5.3 we show isosurfaces of the ISF (left) and isosurfaces of the corresponding segmentation result, showing segmented spatio-temporal tubular structures. These represent cellular evolution of approximately 30 cells from the mono-layered epithelium of a *Drosophila* pupa in morphogenesis.

In fig. 5.4 we present segmentation of the overall shape of the zebrafish embryo during the first stages of the development. The segmentation is visualized as one horizontal and two vertical slices, where we plot original data together with a segmentation isosurface.

REFERENCES

[1] Y. AOYAMA, J. NAKANO, *RS/6000 SP: Practical MPI Programming*, IBM, Poughkeepsie, New York, 1999

[2] Y. BELLAÏCHE, F. BOSVELD, F. GRANER, K. MIKULA, M. REMEŠÍKOVA, M. SMÍŠEK, *New Robust Algorithm for Tracking Cells in Videos of Drosophila Morphogenesis Based on Finding an Ideal Path in Segmented Spatio-Temporal Cellular Structures*, Proceeding of the 33rd Annual International IEEE EMBS Conference, Boston Marriott Copley Place, Boston, MA, USA, August 30 - September 3, 2011, IEEE Press, 2011

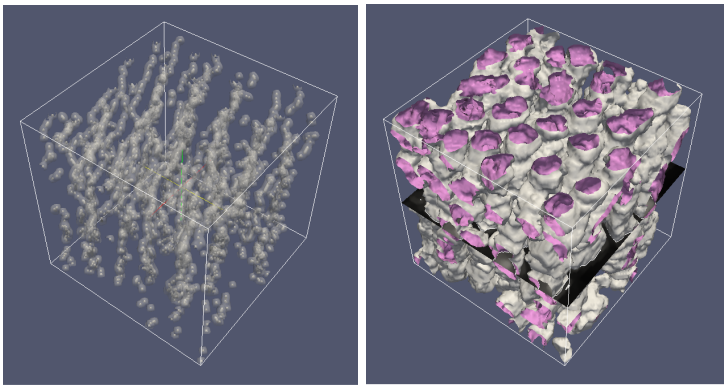

FIG. 5.3. *Drosophila real data analysis. Left - initial segmentation function, right - final segmentation.*

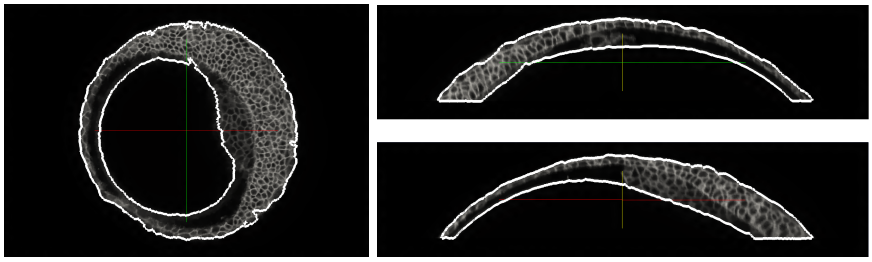

FIG. 5.4. *Zebrafish real data analysis. Left - segmentation slice viewed from top, right - segmentation slices, viewed from sides.*

[3] P. BOURGINE, R. ČUNDERLÍK, O. DRBLÍKOVÁ, K. MIKULA, N. PEYRIÉRAS, M. REMEŠÍKOVÁ, B. RIZZI, A. SARTI, *4D embryogenesis image analysis using PDE methods of image processing*, Kybernetika, Vol. 46, No. 2 (2010) pp. 226-259

[4] P. BOURGINE, P. FROLKOVIČ, K. MIKULA, N. PEYRIÉRAS, M. REMEŠÍKOVÁ, *Extraction of the intercellular skeleton from 2D microscope images of early embryogenesis*, in Lecture Notes in Computer Science 5567 (Proceeding of the 2nd International Conference on Scale Space and Variational Methods in Computer Vision, Voss, Norway, June 1-5,2009), Springer (2009) pp. 38-49

[5] V. CASELLES, R. KIMMEL, G. SAPIRO, *Geodesic active contours*, International Journal of Computer Vision 22 (1997), 67-79

[6] S. CORSARO, K. MIKULA, A. SARTI, F. SGALLARI, *Semi-implicit co-volume method in 3D image segmentation*, SIAM Journal on Scientific Computing, Vol. 28, No. 6 (2006) pp. 2248-2265

[7] L. C. EVANS, J. SPRUCK, *Motion of level sets by mean curvature*, J. Diff. Geom. 33 (1991) pp. 635-681

[8] R. EYMARD, A. HANDLOVIČOVÁ, K. MIKULA, *Study of a finite volume scheme for the regularised mean curvature flow level set equation*, IMA Journal on Numerical Analysis, Vol. 31 (2011) pp. 813-846

[9] P. FROLKOVIČ, K. MIKULA, *Flux-based level set method: a finite volume method for evolving interfaces*, Applied Numerical Mathematics, Vol. 57, No. 4 (2007) pp. 436-454

[10] P. FROLKOVIČ, K. MIKULA, N. PEYRIÉRAS, A. SARTI, *A counting number of cells and cell segmentation using advection-diffusion equations*, Kybernetika, Vol. 43, No. 6(2007) pp. 817-829

[11] S. KICHENASSAMY, A. KUMAR, P. OLVER, A. TANNENBAUM, A. YEZZI, *Conformal curvature ows: from phase transitions to active vision*, Arch. Rational Mech. Anal. 134 (1996), 275-301

[12] K. MIKULA, N. PEYRIÉRAS, M. REMEŠÍKOVÁ, M. SMÍŠEK, *4D numerical schemes for cell image*

segmentation and tracking, in Finite Volumes in Complex Applications VI, Problems & Perspectives, Eds. J. Fořt et al. (Proceedings of the Sixth International Conference on Finite Volumes in Complex Applications, Prague, June 6-10, 2011), Springer Verlag, 2011, pp. 693-702

[13] K. Mikula, N. Peyriéras, M. Remešíková, O. Stašová, *Segmentation of 3D cell membrane images by PDE methods and its applications*, Computers in Biology and Medicine, Vol. 41, No. 6 (2011) pp. 326-339

[14] E. Rouy, A. Tourin, *Viscosity solutions approach to shape-from-shading*, SIAM Journal on Numerical Analysis 29 No. 3 (1992), 867884

[15] A. Sarti, R. Malladi, J. A. Sethian, *Subjective Surfaces: A Method for Completing Missing Boundaries*, Proceedings of the National Academy of Sciences of the United States of America 12 (97) (2000), 62586263