Slovak University Of Technology in Bratislava
Faculty of Civil Engineering

Ing. Balázs Kósa

Dissertation Thesis Abstract

Point cloud surface reconstruction and image segmentation by level set method

to obtain the Academic Title of *philosophiae doctor (PhD.)*
in the doctorate degree study programme
9.1.9 Applied Mathematics
full-time study

Bratislava 2020

Dissertation Thesis has been prepared at Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology in Bratislava.

Submitter:	Ing. Balázs Kósa Department of Mathematics and Descriptive Geometry Faculty of Civil Engineering, STU, Bratislava
Supervisor:	Prof. RNDr. Karol Mikula, DrSc. Department of Mathematics and Descriptive Geometry Faculty of Civil Engineering, STU, Bratislava
Readers:	Doc. RNDr. Zuzana Krivá, PhD. Department of Mathematics and Descriptive Geometry Faculty of Civil Engineering, STU, Bratislava
	Doc. Ing. Gabriel Okša, CSc. Informatics Department of The Mathematical Institute Slovak Academy of Sciences, Bratislava
	Prof. RNDr. Daniel Ševčovič, DrSc. Department of Applied Mathematics and Statistics Faculty of Mathematics, Physics and Informatics, CU, Bratislava

Dissertation Thesis Abstract was sent: ....

Dissertation Thesis Defence will be held on .... at .... am/pm at Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology in Bratislava, Radlinskeho 11.

prof. Ing. Stanislav Unčík, PhD.
Dean of Faculty of Civil Engineering

Abstract

In this work, we deal with the problem of surface reconstruction from 3D point cloud data and the segmentation of 3D images. For these tasks, we apply a mathematical model and numerical method based on the level set algorithm. This method solves the advection equation with a curvature term, which gives the evolution of an initial condition to the final state. For surface reconstruction, the advective velocity can be defined as the gradient of the distance function calculated to the point cloud data. The 3D images contain complicated structures without clear boundaries for which automatic segmentation may not give useful results. We use manually identified point cloud data to distinguish the subvolumes of the image we want to segment. Combining the information of the point cloud data with the 3D image intensity we can define the advective velocity of the level set equation for image segmentation as the weighted sum of distance function and edge detector function gradients. The edge detector function is applied to the presmoothed 3D image. For the calculation of the distance function, we compare the fast sweeping method, the vector distance transform algorithm, the fast marching method, and the Dijkstra-Pythagoras algorithm, and propose how to modify the fast sweeping method and the vector distance transform algorithm for finding the middle surface between different shapes. For the numerical discretization, we use a semi-implicit co-volume scheme in the curvature part and an implicit upwind scheme in the advective part. The method was tested on representative examples and applied to real data. For surface reconstruction, we used data representing archaeological finds and for tissue segmentation microscopic 3D images.

Keywords: point cloud data, microscopic 3D images, surface reconstruction, image segmentation, level set method, distance function calculation

Abstrakt

V rámci tejto práce sa zaoberáme problémom rekonštrukcie plôch z 3D mračna bodov a segmentáciou 3D obrazov. Pre tieto úlohy používame matematický model a numerickú metódu založenú na level set algoritme. Táto metóda rieši advekčnú rovnicu s krivostnou časťou, pomocou ktorej dostávame vývoj počiatočnej podmienky do konečného stavu. Pre rekonštrukciu plôch je advekčná rýchlosť definovaná ako gradient funkcie vzdialenosti k mračnu bodov. V biologických aplikáciach môžu 3D obrazy obsahovať komplikované štruktúry bez jasných hraníc, pre ktoré automatická segmentácia nemusí vždy priniesť užitočné výsledky. Na odlíšenie takýchto štruktúr v obraze, ktorý chceme segmentovať, používame manuálne identifikované mračno bodov. Kombináciou informácií o mračne bodov s intenzitou 3D obrazu môžeme definovať advekčnú rýchlosť v level set rovnici ako vážený súčet gradientov funkcií vzdialenosti a hranového detektora pričom hranový detektor sa aplikuje na vyhladený 3D obraz. Pre výpočet funkcie vzdialenosti porovnávame fast sweeping metódu, algoritmus vector distance transform, fast marching metódu a algoritmus Dijkstra-Pythagorasov a navrhujeme, ako zmeniť fast sweeping metódu a algoritmus vector distance transform na nájdenie strednej plochy medzi rôznymi útvarmi. Pre numerickú diskretizáciu používame semi-implicitnú schému duálnych konečných objemov v krivostnej časti a implicitnú upwind schému v advekčnej časti. Metóda bola testovaná na reprezentatívnych príkladoch a následne použitá na reálne dáta. Pre rekonštrukciu plôch sme použili údaje predstavujúce archeologické nálezy a pre segmentácie tkanív mikroskopické 3D obrazy.

Kľúčové slová: mračno bodov, mikroskopické 3D obrázky, rekonštrukcia plochy, segmentácia obrazu, level set metóda, výpočet funkcie vzdialenosti

Contents

1. Introduction

In the focus of our thesis, there are tasks of image segmentation and surface reconstruction from point cloud data. In the image segmentation, we mainly deal with the analysis of biological samples, represented by 3D microscopic images, where we want to segment different continuous tissues consisting of separate cells or fractions of a specific organ. In such 3D images, we need a suitable way to obtain a digital model of the studied cell populations to be able to analyze their shape, volume, or evolution. For the point cloud surface reconstruction, we used to deal with data obtained by laser scanning, representing, e.g., archaeological finds. Here, the goal is to create a computerized 3D representation of the original object from which the point cloud was created, as accurately as possible. In both cases, the final result is a digital 3D model represented by a triangulated surface as a result of suitable surface evolution. This fact gave us the idea that there are parallels in these two tasks and yield a new mathematical model for image segmentation supported by the point cloud presented in this thesis.

Image segmentation is widely used to divide images into fractions of units with related values in order to allow an easier representation, edge detection, or shape analysis. Such image segments contain data, which are related and have similar features. An image simplified in this manner is easier to analyze and the data it represents is more clear to understand. There are many segmentation methods used by the computer vision community [1, 2, 3, 4], however, their usability is limited by a level of noise and desired shape variability. One of the useful methods for image segmentation is the subjective surface method by [5], which was used in developmental cell biology image processing [6, 7, 8]. However, the application of this method to data with strong noise or the segmentation of complicated cellular structures is not possible without an additional model adjustment. For the biological data we work with, the conventional methods which utilize just the 3D image information, are not sufficient enough to obtain satisfactory results.

The process of point cloud surface reconstruction can be very complex and time-consuming as well. The prime cause of this difficulty is that the point cloud data has no information about ordering or connectivity. Other reasons, which can make the process difficult, are the unknown topology of the scanned object and the presence of noise in the point cloud data.

Various methods already exist to process the point cloud data sets. The most used approaches by commercial software are nonuniform B-spline surfaces, triangulated surfaces, and the substitution of point cloud by shapes defined by mathematical equations representing geometric objects as cuboid, cylinder, cone, or sphere. Several algorithms and software packages are available for different tasks, see e.g. [9, 10], nevertheless, it is difficult to select the optimal one for all requirements. A different approach for the processing of point cloud data is given by the application of the level set method [11]. In [12, 13] an efficient algorithm was proposed utilizing the level set method to solve the problem of surface reconstruction from point cloud data based on [5, 11]. The basic form of the algorithm consists of the solution of a partial differential equation (PDE) representing the level set motion. The solution of the PDE is calculated on a rectangular computational domain, which we choose according to the point cloud. The point cloud data set must be a subset of the computational domain. The solution by the level set method results in the evolution of the level set function as the deformation of an initial guess. The reconstructed surface is then created as an isosurface of the level set function at the numerical steady state.

The level set method has a wide range of applications in image processing, computer graphics, material science, and physics [14, 15]. It gives us a solid base to develop new segmentation methods for biological samples, which do not have clear boundaries and for which other automatic segmentation algorithms do not give any useful results.

The chapters of this thesis are organized in the following way. In Chapter 2 we describe in detail how the level set method is adapted to the task of surface reconstruction from point cloud data by using a specific partial differential equation. This equation is the base of our new mathematical model which we created for 3D image segmentation supported by the point cloud data. We show how we managed to combine the 3D image intensity information and point cloud specifying roughly the border of an object in the image into one algorithm. After the description of the mathematical model, Chapter 3 provides a step by step description of the numerical discretization of the partial differential equation, which is the basis for the implementation of our algorithm.

As seen throughout the thesis the calculation of the distance function is a very important part of the presented algorithms. This is the reason why Chapter 4 is fully dedicated to this topic. Here we analyze the following methods: the fast sweeping method (FSM) [16], the vector distance transform (VDT) algorithm [17], the fast marching method (FMM) [18] and the Dijkstra-Pythagoras (DP) method [19]. While working with the VDT and FSM algorithm we discovered a way to utilize them for the search of a middle surface (skeleton) between two and more input data sets. We describe how we achieved this goal. The search for a skeleton between data sets was tested in several experiments and results are presented in this section, too.

Finally, in Chapter 5 we test our numerical method on real-life data sets. First, in Section 5.1 we analyze in more depth the point cloud surface reconstruction capabilities of our algorithm and study how the resulting reconstruction depends on the parameters of our model. We do this by applying our model to a point cloud data representing an archaeological find created by 3D laser scanning. For the obtained data, we use the mean and maximum directed Hausdorff distances to provide quantifiable means to compare results. Next in Section 5.2 we arrive at our initial goal of 3D image segmentation of complex biological structures. In this section we demonstrate the segmentation capabilities on biological microscopic 3D images.

To obtain the presented results our algorithm and supporting programs were implemented with the help of the programming languages C and C++, in the development environment of Microsoft Visual Studio 2017. To accelerate the run time of our algorithm we implemented parallelization in several parts of our code using the OpenMP library [20]. We provide an example of the parallel computation improvements on the numerical experiment in Section 5.2. All visualization was created with the open source application Paraview.

We thank Dr. Jana Haličková-Brehovská, from the Monuments Board of the Slovak Republic, for providing us with the archaeological point cloud data and the members of the ImageInLife EC project team, Antonia Weberling from the University of Cambridge and Mageshi Kamaraj from CNRS - BioEmergences Gif-sur-Yvette, for providing us with the biological microscopic data.

2. Mathematical model

Our mathematical model is based on the level set method which solves the following advection equation with the curvature term,

$$\begin{gathered} u_t + v \cdot \nabla u - \delta \left|\nabla u\right| \nabla \cdot \left(\frac{\nabla u}{\left|\nabla u\right|} \right) = 0, \\ (x,t) \in \Omega \times [0,T], \end{gathered} \tag{2.1}$$

where $u(x,t)$ is an unknown function, v denotes the advective velocity, the parameter $\delta \in [0,1]$ determines influence of the curvature to the result, Ω is the computational domain, and $[0,T]$ is a time interval. This equation is coupled with homogeneous Neumann boundary conditions and an initial condition.

This level set equation was also used in works [12, 13] where the advective velocity was defined as the gradient of the distance function

$$v = -\nabla d, \tag{2.2}$$

where d is the distance to the point cloud. In these papers, the task was the reconstruction of a surface represented by the point cloud. The basic idea for this type of surface reconstruction was taken from [11, 21], but instead of following just one particular isosurface throughout the evolution of the level set function, the 3D surface is determined from a shock in the final function profile as it is standard in the subjective surface method [5]. In the following paragraphs, we want to demonstrate how our surface reconstruction by the model (2.1) and the level set method works. Let us consider a point cloud data created with parametric equations

$$\begin{aligned}
&x = s_x + r \cdot \left(0.207 + 2.003 \cdot sin^2(\varphi) - 1.123 \cdot sin^4(\varphi)\right) \cdot cos(\varphi) \cdot sin(\theta),\\
&y = s_y + r \cdot cos(\varphi) \cdot sin(\theta),\\
&z = s_z + r \cdot sin(\varphi),\\
&\varphi \in \langle 0, 2\pi), \theta \in \langle 0, \pi).
\end{aligned} \tag{2.3}$$

by using the coefficitents $r = 1.0$, $s = (0,0,0)$ and the step $\frac{\pi}{10}$ for φ and θ. We call this shape the Sponge and will use it as an example on which we will explain how the mathematical model (2.1) works.

We calculate the distance function to the point cloud data and visualize them together in the first picture of Figure 2.1. In [13] Section 2.2. it is shown how we utilize the distance function to generate an initial condition for the function u in equation (2.1). With a tagging algorithm, we find an isosurface of the distance function which contains the point cloud data Ω_0. We can see an example of such a surface in the second picture of Figure 2.1. By this surface the computational domain Ω is divided into two parts. For the initial condition in the external part, we set the value of u to 0.0 and in the internal part to 1.0. Although any surface bounded by Ω which contains Ω_0 could be used for this purpose, by using the isosurface of the distance function we can find a shape already close to the final solution of (2.1), thus speeding up the whole process of surface reconstruction. What we need to take in consideration is that we can not choose an isosurface too close to the point cloud data, because depending on the density of the data set and the density of a discretized computational domain the isosurface could consist of separated shapes around the points of the point cloud data. By this, we would get an incorrect initial condition, for example, the one which is visualized in the third picture of Figure 2.1.

After we managed to obtain a suitable initial condition, by the application of equation (2.1) it can be deformed to obtain the final reconstruction. The advective velocity (2.2) drives the initial condition to the shape of the point cloud data. The fourth picture of Figure 2.1 shows us how the vector field in (2.2) looks like. Here we can see how all the vectors are pointing towards the points in our data set, thus explaining why the gradient of the distance function pulls the initial condition to the final solution.

In theory the final solution of the function u should have the value of 0.0 in the exterior domain of the object represented by the point cloud data and 1.0 in the interior domain. This allows us to extract the desired reconstructed surface form u by visualizing its isosurface with the value 0.5, meaning the border between the two domains. If only the distance function gradient has an influence on the solution we get the reconstructed surface visualized in the fifth picture of Figure 2.1. Here we can see that the solution is too rough and angular. For this reason we include the curvature term in equation (2.1).

The mean curvature flow's level set formulation was introduced in [22]. The curvature is used to smooth out the solution. In the sixth picture of Figure 2.1 we can see the solution obtained by using the curvature coefficient $\delta = 0.025$. Compared to the solution with zero curvature influence the final result is much closer to the desired shape. The curvature term must be applied carefully because if its influence is too strong the shape can shrink too much and lose its desired form. We can see this effect in Figure 2.2 where we visualize results compared to the Sponge point cloud data for different δ values. We can see that for this example the reconstruction is already losing too much of its volume from $\delta = 0.1$. We will examine the influence of the curvature term more deeply taking into account the point cloud and computational grid density in Section 5.1.

The surface reconstruction arises also in the segmentation of 3D images and it represents a boundary of an object in the 3D image. For an illustration let us consider the Sponge shape as a 3D image where the intensity value of voxels (3D pixels) inside the Sponge is 255 and outside 0. We can see this 3D image in Figure 2.3 in the first picture. For classical image segmentation we can utilize the edge detector function $g^0 = g\left(\left|\nabla G_\sigma * I^0\right|\right)$, similarly as it is used in [5, 6, 23, 24, 25, 7]. Here, I^0 denotes the 3D image intensity, G_σ represents a smoothing kernel applied to the image for presmoothing the gradients of the intensity, and g is a function defined as

$$g(s) = \frac{1}{1 + Ks^2}, \tag{2.4}$$

where $K \geq 0$ is an empirically chosen parameter, see [26]. By using this function we can define the advective velocity in (2.1) as

$$v = -\nabla g^0, \tag{2.5}$$

and apply the mathematical model (2.1) to the segmentation of the Sponge 3D Image. For a simple demon-

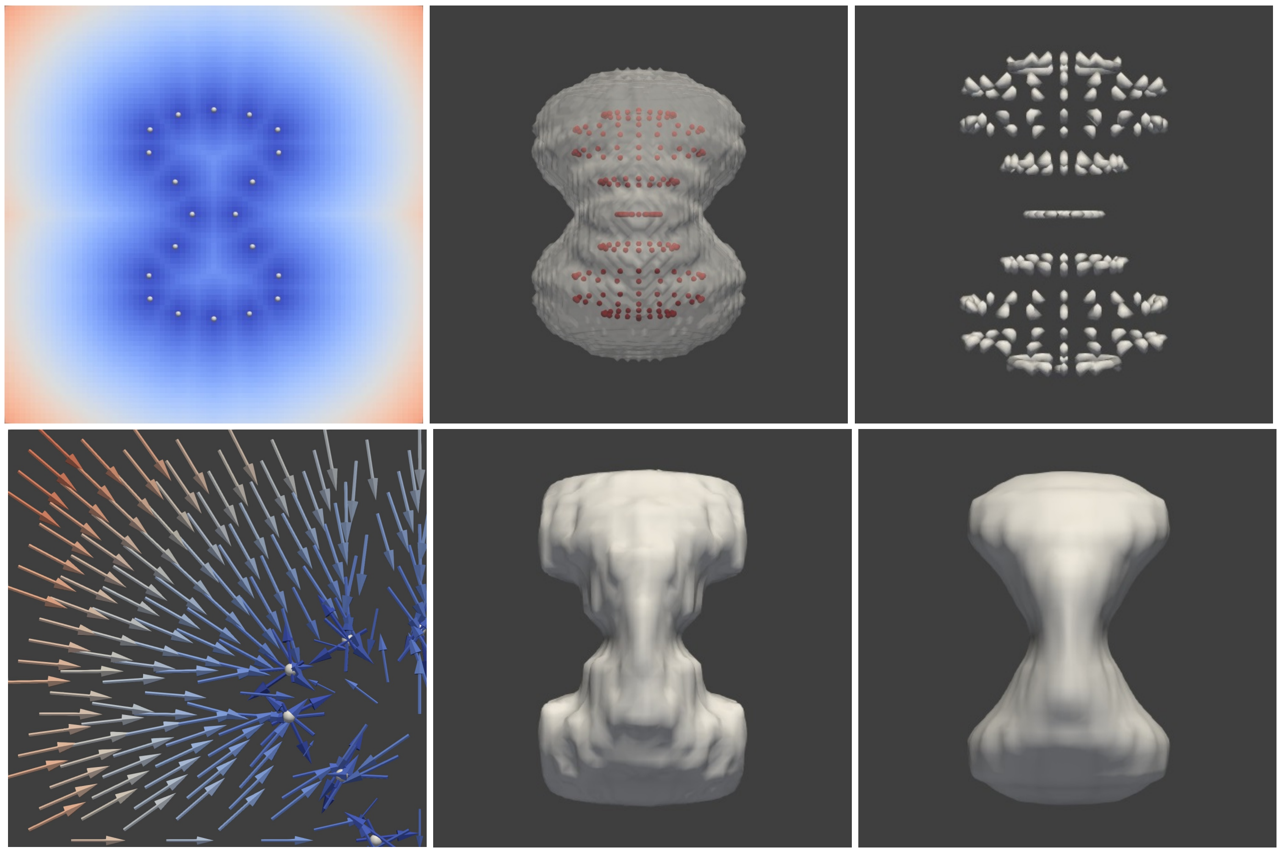

Figure 2.1: Visualization of results calculated to the Sponge data set (from left up to right bottom): the distance function together with the point cloud data; an initial condition for the function u in equation (2.1) calculated as an isosurface of the distance function; an incorrect initial condition for the function u obtained by an isosurface of the distance function too close to the point cloud data; the distance function gradient together with the point cloud data; surface reconstruction obtained by the model (2.1) and the advective velocity (2.2) with no curvature influence, i.e. $\delta = 0.0$; surface reconstruction with curvature influence $\delta = 0.025$. The data in the first column is visualized in the cutting plane $y = 0.0$. For the distance gradient vector field, just a quarter of the image is shown for more detail. The data in the rest of the pictures is visualized as an isosurface from the view parallel to the plane $y = 0.0$.

stration we apply the initial condition, as a sphere around the area we want to segment. Setting the curvature parameter to $\delta = 0.025$ and $K = 10.0$ we obtain the result visualized in the second image of Figure 2.3.

From the examples above we see how point cloud data and 3D image intensity can be processed by the mathematical model (2.1). What we are aiming for is to combine both information in the advective velocity. Our motivation for this is a segmentation task that needed to be executed on a 3D microscopic image. This image contains the microscopic scans of a mouse embryo at early peri-implantation stages provided by the group of Magdalena Zernicka-Goetz, University of Cambridge, Department of Physiology, Development and Neurosciences, working on understanding the development of cell lineages and patterning in the early mammalian embryos, see [27, 28]. We obtained the data in the course of an ImageInLife EC funded project.

At the time of implantation, the embryo is composed of three distinct tissues, the epiblast, which will give rise to the organism, is enveloped by two extraembryonic lineages, the visceral endoderm (VE) and the extra-embryonic ectoderm (ExE). Both provide essential signaling cues mediating proper embryonic development eventually giving rise to yolk sac and placenta respectively [29, 30]. Our colleagues on the biology side are trying to understand the dynamics of the two tissues. The embryo was scanned in 58 sections along the z axis in 1μm steps. In Figure 2.4, we can see section 20 (top row) and 40 (bottom row). In Figure 2.5 we visualize all sections together as a 3D image.

Since any of the tissues could not be segmented automatically by using just the 3D image intensity information, the biologists identify the tissue borders in 2D slices by marking through single points. We can see the

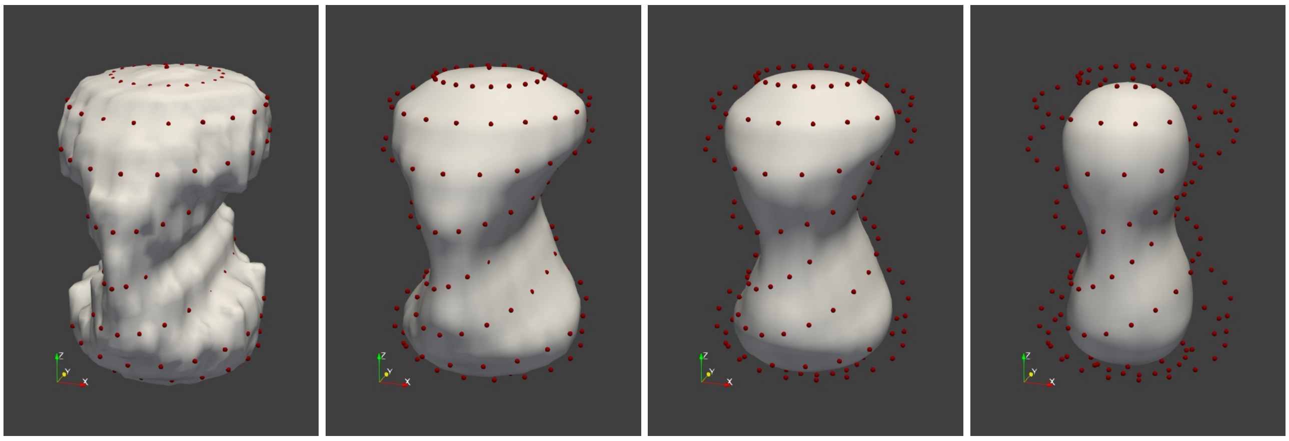

Figure 2.2: Comparison of curvature influence to the final reconstruction of the Sponge point cloud data. The solutions are visualized with the initial point cloud data. The δ parameter for curvature influence is equal to 0.0, 0.05, 0.1, 0.15 (from left to right).

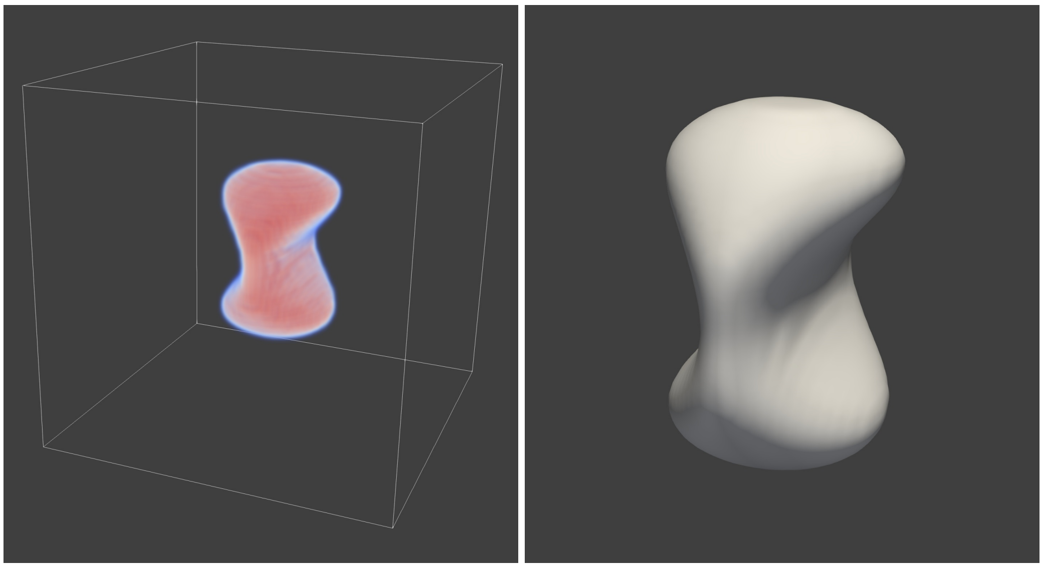

Figure 2.3: In the first picture, we can see a 3D image of the Sponge shape visualized by volume rendering where the lowest values have an opacity of 0.0 and the highest values 1.0. In the second picture, we see the result of the segmentation by the equation (2.1) with advective velocity (2.5) visualized as an isosurface 0.5 of the function u.

examples of manually identified points in Figure 2.4 for VE part. These points representing the tissue borders are the basis for 3D point cloud data entering the model equation (2.1). Gathering the points from all sections and combining them into one data set we obtain point cloud data for both ExE and VE part. We can see the visualization of the point cloud data in Figure 2.5 compared to the cell structure. The idea of using point cloud for segmentation instead of image information was first explored in [31]. In the current work, we aim to combine this idea with the 3D image intensity information.

For this we define the advective velocity v in (2.1) as the combination of the distance function and the edge detector function gradients, i.e.

$$v = \theta\left(-\nabla d\right) + \rho\left(\frac{-\nabla g^0}{\left|\nabla g^0\right|_\varepsilon}\right). \tag{2.6}$$

In the model, the first term $-\nabla d$ drives the level sets of the solution $u(x,t)$ of (2.1) to the points of the cloud and the second term $\left(-\nabla g^0/\left|\nabla g^0\right|_\varepsilon\right)$ drives them to the image edges in the neighborhood of the point cloud. The results of segmentation by using model (2.6) for the mouse embryo tissues can be seen in Section 5.2.

Let us note, that the term $\left|\nabla g^0\right|_\varepsilon$ is calculated according to the Evans-Spruck ε-regularization [32] as

$\sqrt{\varepsilon^2 + |\nabla g^0|^2}$ where $\varepsilon > 0$. The parameters $\theta, \rho \in [0,1]$ determine the influence of the two considered gradients on the surface reconstruction process. The edge detector gradient is normalized in order to get a comparable speed of advection coming from both the distance function and the edge detector function terms.

With the added information of the distance function to the segmentation process, we can again use it to obtain an ideal initial condition as described in the previous paragraphs.

The mathematical model (2.1) with the definition of advective velocity (2.6) appears in our publication [33]. In this article we also published the numerical discretization of the model presented in Chapter 3 and the numerical results appearing in Section 5.2.

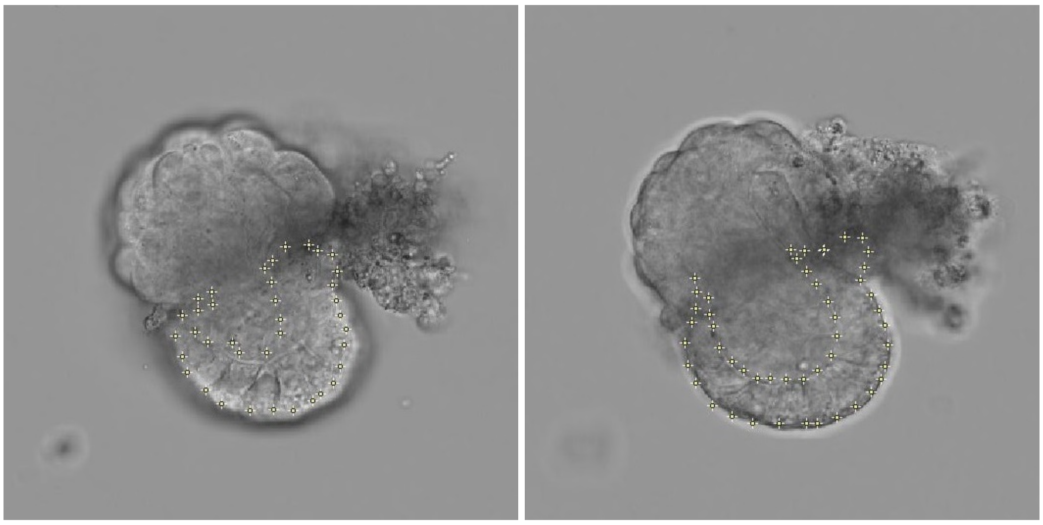
Figure 2.4: Section 20 (left) and 40 (right) of the embryo with the marked points for visceral endoderm (VE).

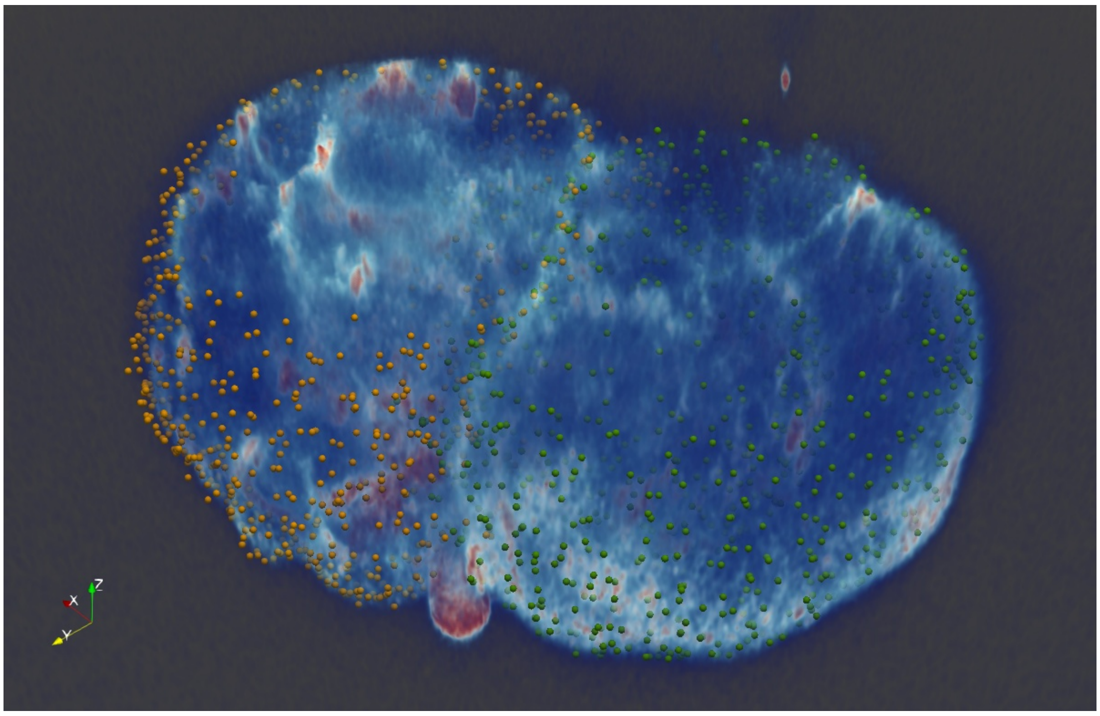

Figure 2.5: 3D image of embryo structure, visualized with volume rendering together with point cloud data for ExE part (orange points) and VE part (green points).

3. Numerical discretization

First of all, we apply a backward difference in time with a uniform time step τ and the semi-implicit time discretization in nonlinear curvature term of (2.1) to get

$$\frac{u^n - u^{n-1}}{\tau} + v \cdot \nabla u^n - \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|} \right) = 0. \tag{3.1}$$

The spatial discretization is performed on a uniform voxel grid where the voxels can be described as cubes with edge size h. The voxels are denoted by the indexes $p = (i, j, k)$ and at each voxel center the value of the image intensity is given by $I^0_{i,j,k}$, the value of computed distance function is denoted by $d_{i,j,k}$, and the computed value of unknown function u^n at time step n in voxel $p = (i, j, k)$ is denoted by both u^n_p and $u^n_{i,j,k}$. For the calculation of the norm of gradient $\left|\nabla u^{n-1}\right|$, on the voxel faces and in the voxel, and the calculation of $\left|\nabla G_\sigma * I^0\right|$ to determine the values of g^0 at voxels, we use the 3D tetrahedral finite element grid $\mathscr{T}_h$ illustrated in Figure 3.1 [6].

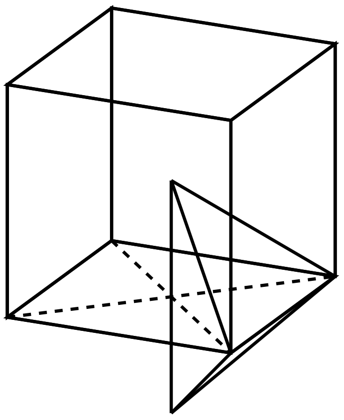

Figure 3.1: The voxel grid cell with a tetrahedral finite element.

The tetrahedral finite elements in $\mathscr{T}_h$ are created by the following approach. Every voxel is divided into six pyramid shaped elements with the base surface given by the voxel's faces and the common vertex by the voxel center. Each of these pyramids is joined with the neighboring pyramids, with which they have a common base surface. These newly formed octahedrons are then split into four tetrahedrons.

For tetrahedral grid $\mathscr{T}_h$, we construct a co-volume mesh, which consists of cells $p = (i, j, k)$ corresponding again to the voxels at our grid, see also [6]. We denote the set of all neighboring cells q of p by C_p. The centers of these cells $q \in C_p$ are all connected to center of p by a common edge σ_{pq} of four tetrahedrons with the length $h_{pq} = h$. Each cell p is bounded by a plane for every $q \in C_p$, which is perpendicular to σ_{pq} and denoted by e_{pq}. The set of tetrahedrons T, which have σ_{pq} as an edge, is denoted by ε_{pq}. For every $T \in \varepsilon_{pq}$, c^T_{pq} is the area of the intersection of e_{pq} and T. Let N_p be the set of tetrahedrons that have p as a vertex. Let u_h be a piecewise linear function on $\mathscr{T}_h$. We use the notation $u_p = u_h(x_p)$ where x_p denotes the coordinates of the center of cell p.

The spatial discretization of (3.1) is derived by using the following form of the equation:

$$\frac{u^n - u^{n-1}}{\tau} + v \cdot \nabla u^n = \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|} \right). \tag{3.2}$$

We begin by integration of (3.2) over every p:

$$\int_p \frac{u^n - u^{n-1}}{\tau} dx + \int_p v \cdot \nabla u^n dx = \int_p \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|} \right) dx. \tag{3.3}$$

For the first part of left hand side in (3.3), we get the approximation in the form

$$\int_p \frac{u^n - u^{n-1}}{\tau} dx = m(p) \frac{u_p^n - u_p^{n-1}}{\tau}, \tag{3.4}$$

where $m(p)$ is a measure in $\mathbb{R}^3$ of the cell p. The second part of the left hand side can be written in an equivalent form by

$$v \cdot \nabla u = \nabla \cdot (uv) - u \nabla \cdot v,$$

$$\int_p v \cdot \nabla u^n dx = \int_p \nabla \cdot (u^n v)\, dx - \int_p u^n \nabla \cdot v\, dx.$$

Considering u^n constant on p in the second term on the right hand side and by using the divergence theorem, we get

$$\begin{gathered}\int_p \nabla \cdot (u^n v) dx - \int_p u^n \nabla \cdot v\, dx = \int_{\partial p} u^n v \cdot \bar{n}\, d\sigma - u_p^n \int_{\partial p} v \cdot \bar{n}\, d\sigma = \\ \sum_{q \in C_p} u_{pq}^n \int_{e_{pq}} v \cdot \bar{n}\, d\sigma - \sum_{q \in C_p} u_p^n \int_{e_{pq}} v \cdot \bar{n}\, d\sigma\end{gathered} \tag{3.5}$$

where u_{pq}^n is an approximate value of the level-set function on the face e_{pq} and $\bar{n}$ is the outer normal to p. We approximate $\int_{e_{pq}} v \cdot \bar{n}\, d\sigma$ in (3.5) by $v_{pq} = h^2 v(x_{pq}) \cdot \bar{n}$, where x_{pq} is a center of e_{pq}, and get

$$\int_p v \cdot \nabla u^n dx = \sum_{q \in C_p} v_{pq} \left(u_{pq}^n - u_p^n \right). \tag{3.6}$$

In order to use the upwind approach, the set C_p is divided into $C_p = C_p^{in} \cup C_p^{out}$, where $C_p^{in} = \left\{ q \in C_p, v_{pq} < 0 \right\}$, which consists of the inflow boundaries, and $C_p^{out} = \left\{ q \in C_p, v_{pq} > 0 \right\}$ consisting of the outflow boundaries. By the upwind approach, we set the values u_{pq}^n to u_q^n if $q \in C_p^{in}$ and to u_p^n if $q \in C_p^{out}$. After these definitions, we can rewrite (3.6) into

$$\int_p v \cdot \nabla u^n dx = \sum_{q \in C_p} min(v_{pq}, 0) \left(u_q^n - u_p^n \right). \tag{3.7}$$

For discretization of the right hand side of (3.3), we consider the term in front of divergence constant on p and then we use the divergence theorem to get

$$\int_p \delta \left| \nabla u^{n-1} \right| \nabla \cdot \left(\frac{\nabla u^n}{\left| \nabla u^{n-1} \right|} \right) dx = \delta \left| \nabla u_p^{n-1} \right| \sum_{q \in C_p} \int_{e_{pq}} \frac{1}{\left| \nabla u^{n-1} \right|} \frac{\partial u^n}{\partial \bar{n}} d\sigma. \tag{3.8}$$

The integral $\int_{e_{pq}} \frac{1}{|\nabla u^{n-1}|} \frac{\partial u^n}{\partial \bar{n}} d\sigma$ and $\left| \nabla u_p^{n-1} \right|$ in (3.8) is approximated numerically using piecewise linear reconstruction of u^{n-1} on the tetrahedral grid $\mathscr{T}_h$, thus we get

$$\int_p \delta \left| \nabla u^{n-1} \right| \nabla \cdot \left(\frac{\nabla u^n}{\left| \nabla u^{n-1} \right|} \right) dx = \delta \left| \nabla u_p^{n-1} \right| \sum_{q \in C_p} \left(\sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left| \nabla u_T^{n-1} \right|} \right) \frac{u_q^n - u_p^n}{h}$$

where

$$\left| \nabla u_p^{n-1} \right| = M_p^{n-1} = \sum_{T \in N_p} \frac{m(T \cap p)}{m(p)} \left| \nabla u_T^{n-1} \right| \tag{3.9}$$

and $\left| \nabla u_T^{n-1} \right|$ denotes the gradient of u_h^{n-1} on T. Then the discretized form of the equation (3.2) is given by

$$\begin{gathered} m(p) \frac{u_p^n - u_p^{n-1}}{\tau} + \sum_{q \in C_p} min(v_{pq}, 0) \left(u_q^n - u_p^n \right) = \\ \delta M_p^{n-1} \sum_{q \in C_p} \left(\sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left| \nabla u_T^{n-1} \right|} \right) \frac{u_q^n - u_p^n}{h}. \end{gathered} \tag{3.10}$$

After rearranging (3.10) and applying the Evans-Spruck ε-regularization $\left|\nabla u_T^{n-1}\right|_\varepsilon = \sqrt{\varepsilon^2 + \left|\nabla u_T^{n-1}\right|}$, the equation has the following form

$$\begin{aligned} &u_p^n + \frac{\tau}{m(p)} \left(\sum_{q \in C_p} min\left(v_{pq}, 0\right)\left(u_q^n - u_p^n\right) \right. \\ &\left. -\delta M_p^{n-1} \sum_{q \in C_p} \left(\sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left|\nabla u_T^{n-1}\right|_\varepsilon} \right) \frac{u_q^n - u_p^n}{h} \right) = u_p^{n-1}. \end{aligned} \tag{3.11}$$

If we define the coefficients

$$a_{pq}^{n-1} = \frac{\tau}{m(p)} \left(min\left(v_{pq}, 0\right) - \delta M_p^{n-1} \frac{1}{h} \sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left|\nabla u_T^{n-1}\right|_\varepsilon} \right), \tag{3.12}$$

we get from (3.11) a system of linear equations with strictly diagonally dominant M-matrix in the form

$$u_p^n + \sum_{q \in N_p} a_{pq}^{n-1} \left(u_q^n - u_p^n\right) = u_p^{n-1}, \tag{3.13}$$

which by addition of the homogeneous Neumann boundary conditions and initial values u_p^0 is solved in every time step n. Since a_{pq}^{n-1} are non-positive coefficients a unique solution exists $(u_1^n, ..., u_M^n)$ of (3.13) where M is a number of unknowns, for any $\tau > 0$, $\varepsilon > 0$, and for every $n = 1, ..., N$.

For the numerical solution of (3.13), we need the values of matrix coefficients a_{pq}. We apply a "finite-difference notation", the cell p is denoted by the index triplet (i, j, k). The value u_p^n is associated with $u_{i,j,k}^n$. At each cell p, the set N_p consists of 24 tetrahedrons on which we compute $|\nabla u_T^n|_\varepsilon$ denoted by $Gu_{i,j,k}^{n,l}$, $l = 1, ..., 24$. To derive the computation of $Gu_{i,j,k}^{n,l}$, we introduce a similar notation as used in [25].

First, we define the index sets P^1, P^2 and P^3 as

$$\begin{aligned} P^1 &= \{(r,s,t)\,; r,s,t \in \{-1,0,1\}\,; |r|+|s|+|t| = 1\} \\ P^2 &= \{(r,s,t)\,; r,s,t \in \{-1,1\}\} \\ P^3 &= \{(r,s,t)\,; r,s,t \in \{-1,0,1\}\,; |r|+|s|+|t| = 2\} \end{aligned}$$

Considering any $(r,s,t) \in P^1$ let $x_{i,j,k}^{r,s,t}$ be the points where the edges σ_{pq} intersect the planes e_{pq}, for every $q \in C_p$. Every vertex of a cubic element is represented by $z_{i,j,k}^{r,s,t}$, for $(r,s,t) \in P^2$. The midpoints between each neighboring vertices of cubic elements are denoted by $y_{i,j,k}^{r,s,t}$, for $(r,s,t) \in P^3$. The explained notation is presented in Figure 3.2.

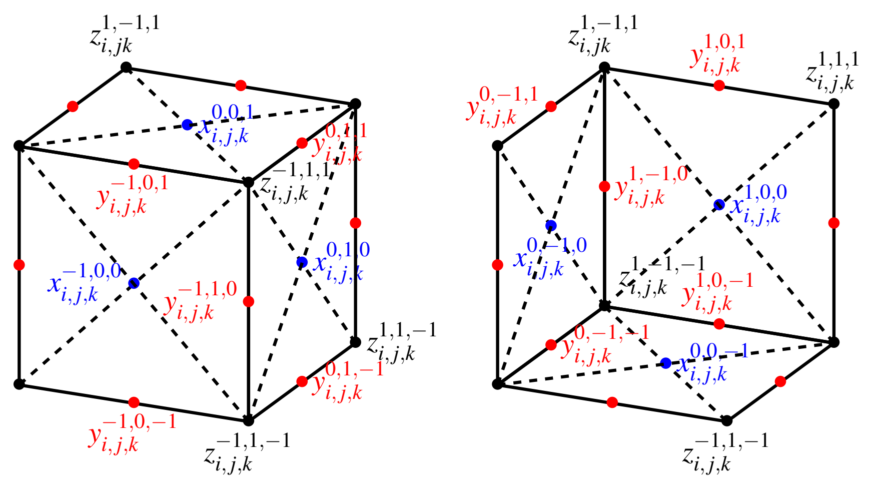

Figure 3.2: Notation for the additional points of a grid cell used for calculation of $Gu_{i,j,k}^{n,l}$, $l = 1, ..., 24$

The values u^{n-1} approximated at $x^{r,s,t}_{i,j,k}$, $z^{r,s,t}_{i,j,k}$ and $y^{r,s,t}_{i,j,k}$ are denoted by $u^{r,s,t}_{i,j,k}$, leaving out the time index. In $x^{r,s,t}_{i,j,k}$, $(r,s,t) \in P^1$, the values are calculated as the average value for two neighboring cells,

$$u^{r,s,t}_{i,j,k} = \frac{1}{2}\left(u^{n-1}_{i,j,k} + u^{n-1}_{i+r,j+s,k+t}\right), \text{for } (r,s,t) \in P^1.$$

In $z^{r,s,t}_{i,j,k}$, $(r,s,t) \in P^2$, as the average value of the 8 voxels which have $z^{r,s,t}_{i,j,k}$ as a common vertex,

$$u^{r,s,t}_{i,j,k} = \frac{1}{8}(u^{n-1}_{i,j,k} + u^{n-1}_{i+r,j,k} + u^{n-1}_{i,j+s,k} + u^{n-1}_{i,j,k+t} + u^{n-1}_{i+r,j+s,k} +$$

$$u^{n-1}_{i+r,j,k+t} + u^{n-1}_{i,j+s,k+t} + u^{n-1}_{i+r,j+s,k+t}), \text{for } (r,s,t) \in P^2.$$

In $y^{r,s,t}_{i,j,k}$, $(r,s,t) \in P^3$, as the averages of values $u^{r,s,t}_{i,j,k}$ in points $z^{r,s,t}_{i,j,k}$,

$$u^{r,s,t}_{i,j,k} = \frac{1}{2}\left(u^{r+\xi(r),s+\xi(s),t+\xi(t)}_{i,j,k} + u^{r-\xi(r),s-\xi(s),t-\xi(t)}_{i,j,k}\right), \text{for } (r,s,t) \in P^3,$$

where

$$\xi(x) = \begin{cases} 0 & \text{if } x \in \{-1,1\} \\ 1 & \text{if } x = 0 \end{cases}.$$

Norm of gradient $\left|\nabla u^{n-1}_T\right|_\varepsilon$ is computed by using values in two neighboring voxel centers, in vertices of voxels corresponding to tetrahedral edge, $z^{r,s,t}_{i,j,k}$, and center of that edge, $y^{r,s,t}_{i,j,k}$, and center of the face, $x^{r,s,t}_{i,j,k}$. Approximation of partial derivatives included in gradient computation for one tetrahedra is illustrated in Figure 3.3. By red color and dashed pattern we indicate the line which is used for approximating $\frac{\partial u_h}{\partial x}$, by green dotted line approximation of $\frac{\partial u_h}{\partial y}$, and blue dash-dotted line approximation of $\frac{\partial u_h}{\partial z}$. For other tetrahedrons it is done similarly. Thereby, we get the values $Gu^{n-1,l}_{i,j,k}$ $l = 1,...,24$.

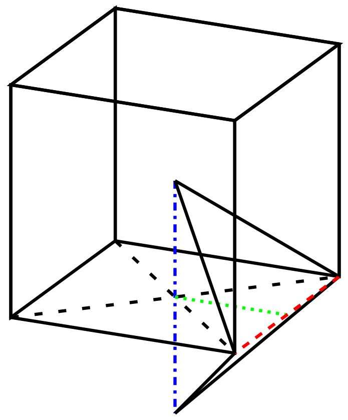

Figure 3.3: Tetrahedral finite element with marked edges for approximation of partial derivatives.

Besides $\left|\nabla u^{n-1}_T\right|_\varepsilon$, given using the above explanation and M^{n-1}_p given by (3.9), we need to determine also the values v_{pq} in (3.12). Taking into account the definition (2.6), we get

$$v_{pq} = h^2\left(\theta(-\nabla d) + \rho\left(\frac{-\nabla g^0}{|\nabla g^0|_\varepsilon}\right)\right)\Big|_{x_{pq}} \cdot \bar{n}. \tag{3.14}$$

On the six voxel faces we can write

$$v^t_{i,j,k} = -\theta h\left(d_{i,j,k+1} - d_{i,j,k}\right) - \rho h\left(\left(g^0_{i,j,k+1} - g^0_{i,j,k}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right)$$

$$v^b_{i,j,k} = \theta h\left(d_{i,j,k} - d_{i,j,k-1}\right) + \rho h\left(\left(g^0_{i,j,k} - g^0_{i,j,k-1}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right)$$

$$v^n_{i,j,k} = -\theta h\left(d_{i+1,j,k} - d_{i,j,k}\right) - \rho h\left(\left(g^0_{i+1,j,k} - g^0_{i,j,k}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right)$$

$$v^s_{i,j,k} = \theta h \left(d_{i,j,k} - d_{i-1,j,k}\right) + \rho h \left(\left(g^0_{i,j,k} - g^0_{i-1,j,k}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right)$$

$$v^e_{i,j,k} = -\theta h \left(d_{i,j+1,k} - d_{i,j,k}\right) - \rho h \left(\left(g^0_{i,j+1,k} - g^0_{i,j,k}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right)$$

$$v^w_{i,j,k} = \theta h \left(d_{i,j,k} - d_{i,j-1,k}\right) + \rho h \left(\left(g^0_{i,j,k} - g^0_{i,j-1,k}\right) / \left|\nabla g^0_{i,j,k}\right|_\varepsilon\right).$$

To obtain these values, we need to evaluate the discrete values of $g^0_{i,j,k} = g\left(\left|\nabla G_\sigma * I^0_{i,j,k}\right|\right)$ for every voxel. To achieve this, we solve the convolution $I^\sigma = G_\sigma * I^0$ through solving the linear heat equation by the implicit scheme with a time step related to σ, where also the homogeneous Neumann boundary conditions are applied. Now, we can calculate $\left|\nabla I^\sigma_{i,j,k}\right|$ analogously to (3.9) and finally we get

$$g^0_{i,j,k} = g\left(\frac{1}{24}\sum_{l=1}^{24} GI^{\sigma,l}_{i,j,k}\right). \tag{3.15}$$

With the discrete values of g^0_p, we compute norm of regularized gradient $\left|\nabla g^0_p\right|_\varepsilon = \sqrt{\varepsilon^2 + \left|\nabla g^0_p\right|^2}$ where $\left|\nabla g^0_p\right|$ is given according to the "magic formula" as described in [34]:

$$\left|\nabla g^0_p\right|^2 = \frac{1}{m(p)} \sum_{e_{pq}\in C_p} m(e_{pq}) \frac{1}{d_{pq}} \left(g^0_{pq} - g^0_p\right)^2 \tag{3.16}$$

where d_{pq} is the distance between the center of cell p and the center of e_{pq}. With $m(p) = h^3$, $m(e_{pq}) = h^2$, $d_{pq} = \frac{h}{2}$ and using $g^0_{pq} = \frac{g^0_q + g^0_p}{2}$, we get:

$$\left|\nabla g^0_p\right| = \sqrt{\frac{1}{2}\sum_{q\in C_p}\left(\frac{g^0_q - g^0_p}{h}\right)^2}. \tag{3.17}$$

So we get

$$\left|\nabla g^0_{i,j,k}\right| = \sqrt{\frac{1}{2}\sum_{(r,s,t)\in P^1}\left(\frac{g^0_{i+r,j+s,k+t} - g^0_{i,j,k}}{h}\right)^2} \tag{3.18}$$

By using the above considerations, we define

$$b_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^b_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=1}^{4}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

$$t_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^t_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=5}^{8}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

$$n_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^n_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=9}^{12}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

$$s_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^s_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=13}^{16}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

$$e_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^e_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=17}^{20}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

$$w_{i,j,k} = \frac{\tau}{h^3}\left(min\left(v^w_{i,j,k}, 0\right) - \delta M^{n-1}_{i,j,k}\frac{h}{4}\sum_{l=21}^{24}\left(\varepsilon^2 + \left(Gu^{n-1,l}_{i,j,k}\right)^2\right)^{-\frac{1}{2}}\right)$$

where $M^{n-1}_{i,j,k}$ is given by

$$M^{n-1}_{i,j,k} = \sqrt{\varepsilon^2 + \left(\frac{1}{24}\sum_{l=1}^{24} Gu^{n-1,l}_{i,j,k}\right)^2}$$

and the diagonal coefficient is defined as

$$c_{i,j,k} = 1 - b_{i,j,k} - t_{i,j,k} - n_{i,j,k} - s_{i,j,k} - e_{i,j,k} - w_{i,j,k}.$$

With these notations, we rewrite (3.13) into the form of a system of linear equations

$$\begin{aligned} &c_{i,j,k}u^n_{i,j,k} + b_{i,j,k}u^n_{i,j,k-1} + t_{i,j,k}u^n_{i,j,k+1} + n_{i,j,k}u^n_{i+1,j,k} \\ &+ s_{i,j,k}u^n_{i-1,j,k} + e_{i,j,k}u^n_{i,j+1,k} + w_{i,j,k}u^n_{i,j-1,k} = u^{n-1}_{i,j,k}, \end{aligned} \tag{3.19}$$

for all (i,j,k). This system is solved at every time step by the SOR (Successive Over Relaxation) iterative method. We stop the calculation when the residuum between two consecutive time steps is lower than a chosen tolerance.

For the calculation, we need discrete initial values for the level set function $u(x,0)$ in the zero time step. This initial condition is determined by a simple tagging algorithm using distance function to the point cloud. With further modification of this tagging algorithm, we are able to construct a band around the area between the initial surface and point cloud data for computation acceleration [13]. For finding the surface, which we want to reconstruct, it is sufficient to update the solution values on grid cells contained only in such band.

4. Calculation of the distance function

The calculation of the distance function is a very important part of the numerical solution of (2.1). In our previous works [35, 13, 33] for the calculation of the distance function we mainly used the so-called fast sweeping method described in [16]. In this chapter, we will analyze other options for this task by comparing their accuracy and efficiency. This analysis is limited to the data sets we work with and the methods which we found helpful in our research, namely the fast sweeping method (FSM) [16], the vector distance transform (VDT) algorithm [17], the fast marching method (FMM) [18] and the Dijkstra-Pythagoras (DP) method [19]. More detailed surveys can be found in works such as [36, 37]. Besides the analysis, we propose how to use two of the discussed methods for the search of the middle surface, the so-called skeleton, between two or more data sets.

4.1 Comparing methods

To compare the mentioned algorithms, we construct the following experiment. Let us have a rectangular computational domain which has its grid point of minimum coordinates at $(-0.4,-0.4,-0.4)$ and of maximum coordinates at $(1.4,1.4,1.4)$. In this domain we insert a cube which has its vertex with minimum coordinates at $(0.0,0.0,0.0)$ and maximum coordinates at $(1.0,1.0,1.0)$. We choose these parameters thus we can discretize the computational domain in a way that some of the grid points will lie on the surface of the cube. In this way, we can initialize the value of the distance function at these points with the exact value 0. With this setup, we computed the distance function for the Cube with the four algorithms on the computational domain discretized to a grid by voxels with different edge sizes, namely 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625, 0.003125. You can see that the next size is always half of the previous. We demonstrate how the distance function looks like on these grids in Figure 4.1 calculated by the FSM algorithm.

To compare the precision of the algorithms, we calculated the mean squared difference from the exact solution for all grids. We are listing these results in Table 4.1. In the first column, we list the number of grid points in x, y, z directions of our computational grid. In the second column, we see the length of the voxel edges. In the following columns, we see the mean squared difference for FSM, VDT, FMM, and DP methods. We can see that values for the VDT and DP methods are basically 0. This is because they yield Euclidean distance results. The results of FSM and FMM are less precise. We also measured the CPU time (in seconds) which was needed to calculate the distance function with the different methods. We are listing this in Table 4.2. Here again, we list the parameters of our grid first. In the third column, we list the CPU time for the initialization phase of the algorithms. The initialization is the same for all four methods. Because of the simplicity of the experiment, this takes just a few seconds even for the biggest grid. The following columns show the measured times for the four algorithms. Comparing all the data we can see that with respect to computation duration the FSM algorithm outperforms all other methods.

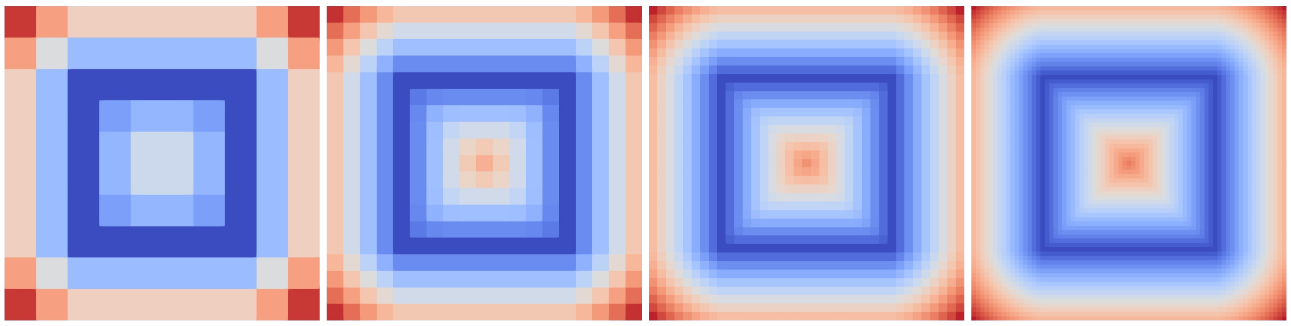

Figure 4.1: Distance function visualization for the Cube experiment, we visualize the section in a constant z plane for voxel edge sizes 0.2, 0.1, 0.05, 0.025. Values go from the highest dark red to the lowest dark blue. Results were calculated by FSM.

Number of grid points	Voxel edge size	FSM	VDT	FMM	DP
10^3	0.2	2.5692e-03	1.4791e-34	2.5692e-03	4.6622e-33
19^3	0.1	9.7901e-04	1.8869e-34	9.7902e-04	6.4759e-33
37^3	0.05	3.7697e-04	1.0579e-34	3.7697e-04	1.2404e-32
73^3	0.025	1.4352e-04	5.0275e-35	1.4352e-04	1.0659e-32
145^3	0.0125	5.3092e-05	2.5195e-35	5.3092e-05	9.9842e-33
289^3	0.00625	1.8949e-05	1.3057e-35	1.8949e-05	3.5044e-32
577^3	0.003125	6.5244e-06	6.5770e-36	6.5244e-06	1.4129e-31

Table 4.1: Mean squared difference comparison for distance function calculation methods tested on the Cube experiment.

Number of grid points	Voxel edge size	Initialization	FSM	VDT	FMM	DP
10^3	0.2	0	0.001	0.002	0.001	0.001
19^3	0.1	0	0.002	0.017	0.002	0.004
37^3	0.05	0.001	0.008	0.029	0.022	0.032
73^3	0.025	0.014	0.059	0.186	0.187	0.274
145^3	0.0125	0.105	0.352	1.416	2.14	2.781
289^3	0.00625	0.816	2.881	11.352	26.904	28.869
577^3	0.003125	6.375	24.442	88.836	322.29	284.488

Table 4.2: CPU time comparison for distance function calculation methods tested on the Cube experiment. CPU time was measured in seconds.

4.2 Searching for the middle surface (skeleton)

While analyzing the algorithms for distance function calculation, we realized that methods which track the source of the distance in each grid point can be easily modified for the search of middle surfaces between data sets. The problem of finding the skeleton of the input data, in other words, the identification of the medial axis, occurs in other computational tasks, such as the optimal mesh generation [38], which uses the information about the skeleton to densify or coarsen the computational grid in the computational domain. We propose how to use the VDT and FSM algorithms to find the middle surface if we have more data sets in our computational domain. Our approach is based on information propagation throughout which we track the source of the information.

First, let us discuss how the algorithms need to be modified if we deal with more data sets at the same time. When we initialize the distance function we need to create a large enough computational domain to fit all data sets and we will set the initial values for each separately. What we need to keep track of additionally, during the initialization and the execution of the algorithms, is the information from which data set the values come from. For VDT, the distance function values in the grid points of the computational domain are calculated as an exact Euclidian distance between the grid point and an element of the input data set. For this reason, VDT keeps track of the sources for distance calculation. In these sources, we will add the information to which data set the source belongs to. FSM originally does not contain any information of sources, thus we needed to include this in the proper manner. In every iteration of the algorithm when we cycle through the grid points, we take the distance value from the neighboring points to solve a quadratic equation. We need to keep track, from which neighbors the distance values enter the quadratic equation, thus we save the indexes (r,s,t), $r,s,t \in \{-1,0,1\}$, which identify them. When the solution is calculated for the equation we add up the indexes (r,s,t) and this will show us which source to save for the current grid point from the sources of the 26 neighbors.

With this approach at the end of the algorithm, the grid points in our computational domain will be divided into volumes belonging to the different data sets. To obtain the middle surface between the data sets we just need to find the borders between these volumes. For this, we use two methods. For any number of data sets, we can cycle through all points of the computational domain and find every point which has a neighbor belonging to a different volume. If we apply this for every data set separately, for each of them we obtain the set of points which are at the borders between the volumes. If we have just two data sets, we can treat the obtained information about which data set the grid points belong to, as a function of values 0 or 1, and visualize the isosurface of the function with the value 0.5. We can see this method in our first numerical experiment for finding the skeleton of the distance function between two data sets.

4.2.1 Experiment 1: Sponge & Sphere

Let us have two point cloud data sets. The first is the Sponge point cloud data created with parametric equations (2.3). The parameter r of the Sponge is 1.0. The second point cloud data is a sphere with a radius of 0.5. The distance between the centers of the two objects is 2.0. To create the point cloud data we used a step of $\frac{\pi}{10}$ for both angles in the parametric equations. We calculated the distance function on a grid with dimensions $168 \times 123 \times 123$ and voxel edge size 0.025. For the middle surface calculated by the Modified VDT algorithm, we obtained the result seen in Figure 4.2 visualized as an isosurface together with the point cloud data sets.

When we tested the Modified FSM algorithm, we expected similar results but we discovered that it can cause some issues in specific situations. With the regular initialization of the distance function calculated to point cloud data on a grid with density higher than the point cloud density, we get an initial value which consists of separated volumes around the points. We can see this in a 2D section for our experiment in the first picture of the left column in Figure 4.3. The problem is that these gaps in the initial value do not contain any source information and if we apply the Modified FSM algorithm this lack of information can propagate through our computational grid. This leads to errors when we are trying to detect grid points on the borders of volumes belonging to the different point cloud data sets or when we want to visualize the border as an isosurface of a function. The isosurface with errors can be seen in the second picture of the left column in Figure 4.3.

To solve this problem, we need to modify also the initialization to point cloud data for the FSM algorithm. The idea is to get a "continuous" subvolume for the initialized grid points. For this we need to increase the area around the single points in which we calculate the distance function for the grid points. We need to find the minimum size of this area so that for two neighboring cloud points the areas will intersect. We found that for this minimum size we can use the maximum minimal distance between all elements of a point cloud data. With its value, we build a cube around every cloud point which determines the subvolume in which we will calculate the exact distance values for the grid points. We can see the result of this modification in the right column of Figure 4.3. In the first picture, visualizing a section of the new initial condition, we can see that now we have a "continuous" subvolume of grid points. In the second picture we can see that the isosurface is now obtained without any error.

In the following experiments we will show different cases how we can apply the described algorithms and show how different the results from the two methods can be.

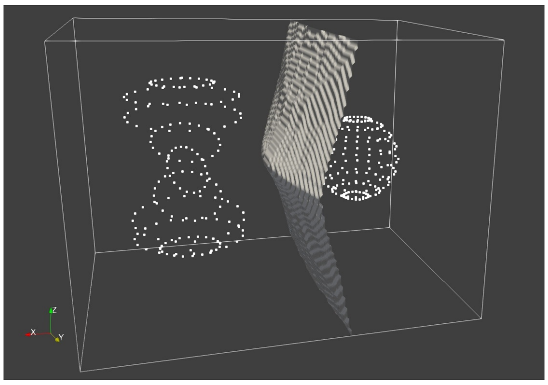

Figure 4.2: Experiment 1: Border between Sponge and Sphere point cloud data calculated by the VDT algorithm.

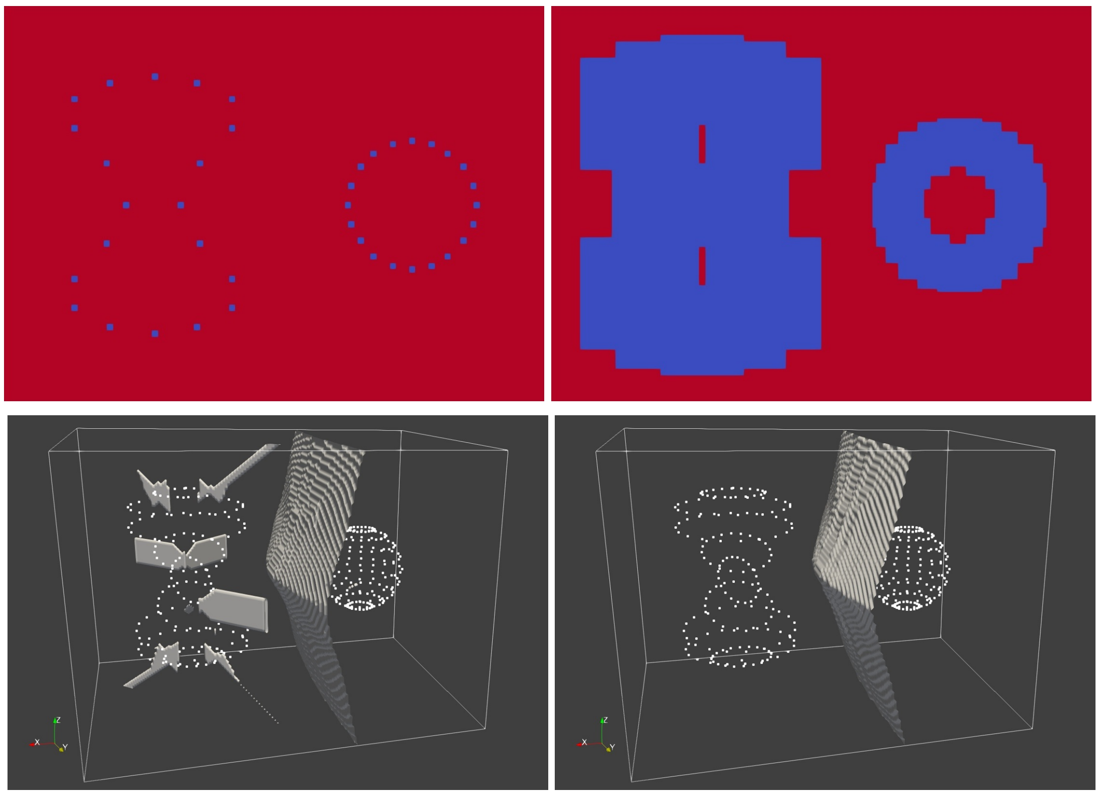

Figure 4.3: Experiment 1: Finding the border between Sponge and Sphere point cloud data by the FSM algorithm. In the first picture of the left column, we see the section of the original initial condition in a constant y. In the second picture of the left column, the incorrect isosurface between divided areas of the computational grid is visualized. In the first picture of the right column, we can see the corrected initial condition. In the second picture of the right column, the correct isosurface (skeleton) between divided areas of the computational grid is visualized.

4.2.2 Experiment 2: Subsets of the Cube

We return to the case of the Cube data set which has coinciding points with the grid as described at the beginning of Section 4.1. We will use the computational grid with the voxel edge size 0.05. The points on every subset of the Cube (vertex, edge, wall) will be treated as a separate data set. The vertices are treated as one-point data sets, the edges do not contain the vertices and the walls do not contain either the edges or the vertices. Now in this setup, we apply both the Modified VDT and Modified FSM algorithms.

First we analyze the results for VDT. In Figure 4.4 we can see the borders of the volumes assigned to the different subsets. For clearer visualization, we show just some of the separate borders with the outlines of the Cube as white lines. We can identify by color to which subset of the Cube the points belong to. For the other subsets, the volumes are symmetrical. What we need to notice is that to the inside of the Cube only information from the walls propagates. From the vertices and edges, the information only propagates outward. For this reason, the borders of the volumes inside of the cube are not uniform. While the volume borders on the outside of the Cube can be separated into sets of points lying along different planes, on the inside of the Cube this is not possible.

Let us compare this to the results of the FSM algorithm visualized in Figure 4.5. We can identify by the colors that the information propagates inward from all subsets of the Cube. Inside of the Cube, the grid points belonging to vertexes are along a line, for edges, the grid points are confined to a triangle, and for the walls, they are inside a pyramid. In these results, the borders of the separated volumes are much clearer and sharper. This difference can be explained by the fact that in the VDT algorithm we take into consideration only the sources from 6 neighbors, ones next to the grid point, but in the FSM algorithm, the source information can come from all 26 neighbors.

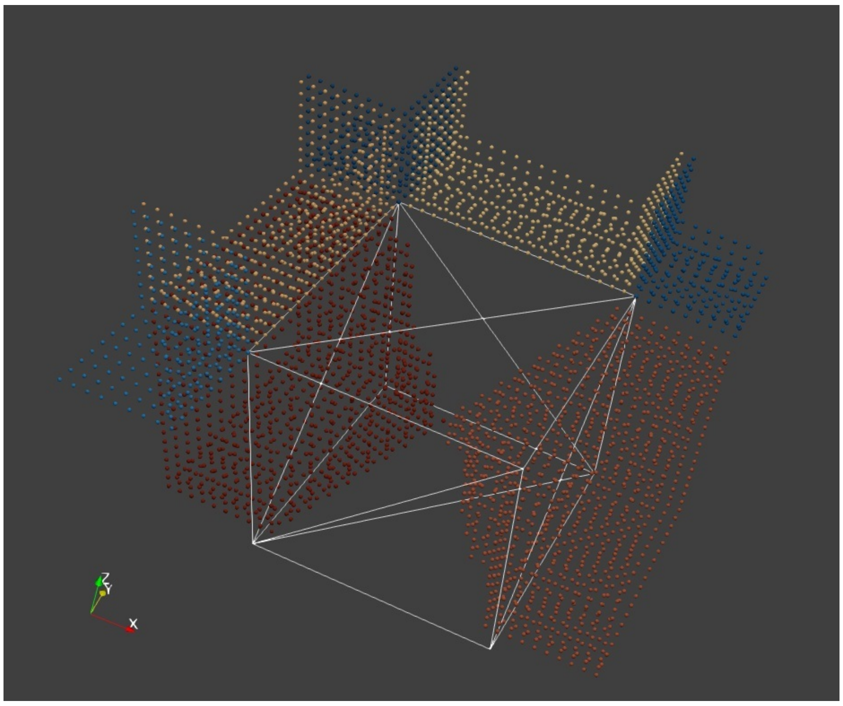

Figure 4.4: Experiment 2: Visualization of borders from sources on the Cube data set for the VDT method.

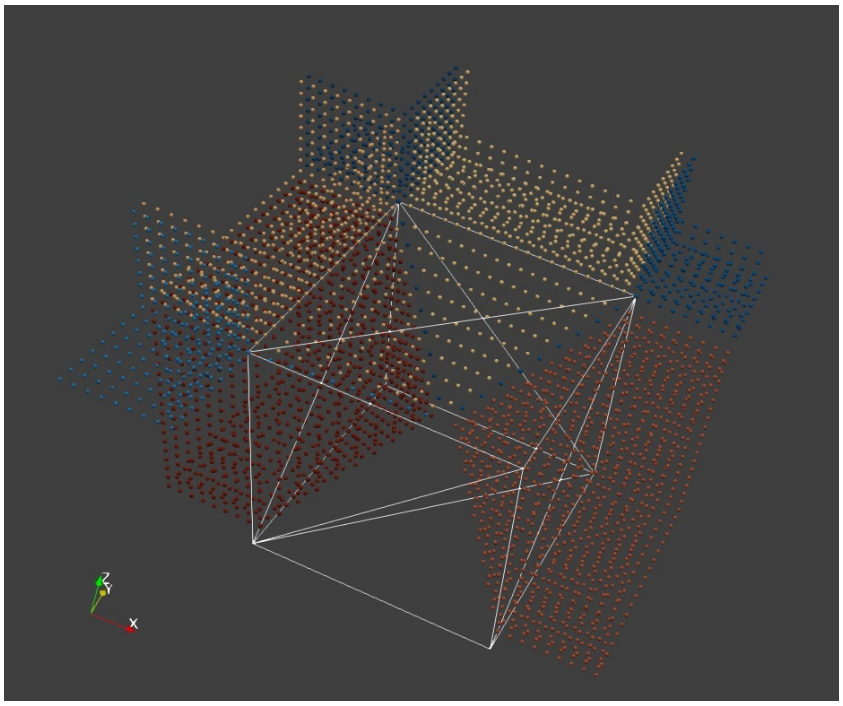

Figure 4.5: Experiment 2: Visualization of borders from sources on the Cube data set for the FSM algorithm.

4.2.3 Experiment 3: Two parallel surfaces

For the last experiment in this section we want to show how accurately the algorithms can find the middle surface between two parallel data sets. For this purpose, we will use wawe-like surfaces generated as point cloud data by functions

$$\begin{aligned} f(x,y) &= 0.2 * cos(x*y) + 0.5, \\ f(x,y) &= 0.2 * cos(x*y) - 0.5, \\ (x,y) &\in < -5.0, 5.0 > \times < -5.0, 5.0 > \end{aligned} \tag{4.1}$$

with a step of 0.05 for both x and y parameters.

We can see the visualization of the point cloud data generated by the first equation of (4.1) in the first picture of Figure 4.6 In the second picture we can see the result of the calculations by FSM algorithm on a computational grid with voxel edge size 0.025 represented as an isosurface. This isosurface lies between the two parallel point cloud data sets. Visually the results for the two methods do not show noticeable differences, thus we show only the results for FSM.

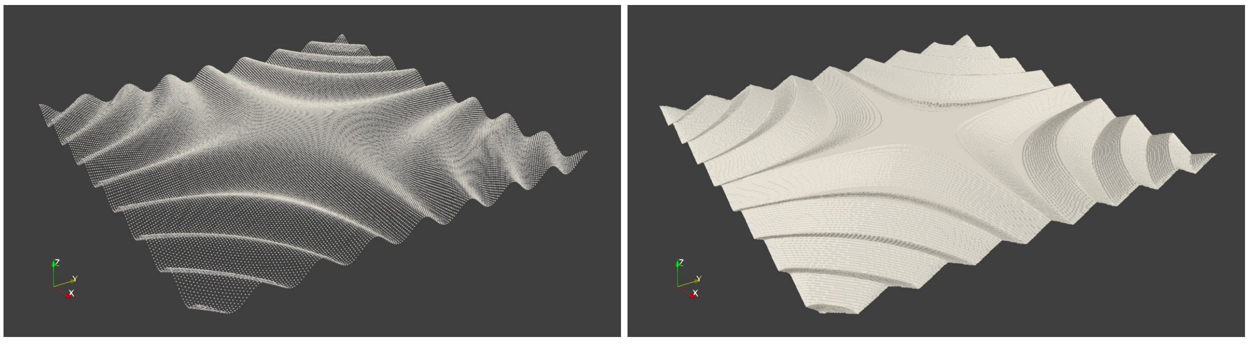

Figure 4.6: Experiment 3: Finding the border between two parallel wave-like point cloud data sets generated with equations (4.1). In the first picture, we see the visualization of one point cloud data. The other one is identical just shifted along the z axis. In the second picture, we visualize the isosurface which divides the computational domain between the two point cloud data sets.

5. Numerical experiments

5.1 Point cloud surface reconstruction

In this experiment we are testing the influence of different aspects of the mathematical model (2.1) with advective velocity (2.2) on the final solution of point cloud surface reconstruction. We demonstrate how the final solution can differ depending on the density of the computational background grid, by changing the size of voxel edges h, and the influence of the curvature part of equation (2.1), given by the parameter δ. We also compare the influence of methods for distance function calculation. As we have shown in Section 4.1 in Table 4.1 the pairs of FSM, FMM and VDT, DP yield similar results, thus we will demonstrate results only for one method of each pair, namely the FSM and the VDT algorithms. To be able to quantify our results we calculate the mean and maximum directed Hausdorff distance between input point cloud data and the resulting reconstructed surface, which is represented by a triangulated isosurface of the level set function u, solution of the equation (2.1). For finding the triangulated isosurface we use the Marching Cubes algorithm [39] from the Visualization Toolkit open source software system [40].

Let $A = \{a_1, a_2, ..., a_p\}$ be the set of points in the point cloud data and $B = \{b_1, b_2, ..., b_p\}$ the set of triangles in the reconstructed surface. By this notation we can define the mean and the maximum directed Hausdorff distance as

$$\begin{aligned} MeanHD(A,B) &= \frac{1}{p}\sum_{i=1}^{p} \min_{b\in B} \|a_i - b\|, \\ MaxHD(A,B) &= \max_{a\in A}\left(\min_{b\in B} \|a - b\|\right), \end{aligned} \tag{5.1}$$

where $\|...\|$ is the distance between a point a and a triangle b. With $HD(A,B)$ for every point in the cloud we search for the closest triangle from the triangulated surface which gives us a good estimate of point cloud representation by the triangulated surface. For the value of $MeanHD(A,B)$ we calculate the average of all distances and for the value of $MaxHD(A,B)$ we take the largest distance.

For the experiment, we use a real-life data set representing a 3D scan of an archaeological find, an ancient Sealer. We can see the visualization of the point cloud data in the first picture of Figure 5.1. This point cloud data contains 42970 points with an average point density of 0.408695. The results are presented in Tables 5.1 and 5.2. In the tables we can see that, we choose the voxel edge sizes $h \in \{0.8, 0.4, 0.2, 0.1\}$. This choice was made to show h influence if it is bigger, equal, or smaller than the average point cloud density. For the parameter δ we present results in the range 0.0 to 0.05, because higher values yielded no valuable information and we already saw in the example presented in Figure 2.2 that a higher δ causes too big deformations in the reconstructed shape.

Table 5.1 shows the results which were obtained by calculating the distance function d for the advective velocity (2.2) by the FSM algorithm. Here we see that already for a grid density approximate to point cloud density, we get good results, but when voxel edge size is twice and four times smaller the values of Hausdorff functions are significantly better. We conclude from this that the best choice is a background grid with density at least twice the density of the point cloud data. To analyze the influence of the parameter δ we need to take a look at the values of function $MaxHD(A,B)$. These values sharply decrease with δ greater than 0.0 and just slowly increase with δ greater than 0.1. From this, we see that it is necessary to have a curvature influence with a coefficient around $\delta = 0.01$. The Table 5.2 shows the results which have been obtained by calculating the distance function by the VDT method. We can see that the results show lower values for the Hausdorff distance, thus we obtained more accurate reconstruction, but these differences are not large. In Section 4.1 we have shown that on a background grid with high density the calculation of the distance function by VDT method can take 2 to 3 times longer, thus one has to also consider the balance between accuracy and computational time. We show the visualization of the reconstruction with parameters $h = 0.1$ and $\delta = 0.01$ in pictures two and three of Figure 5.1.

δ	0.000	0.01	0.02	0.03	0.04	0.05
$h = 0.8$; Grid dimensions: $70 \times 115 \times 95$						
MeanHD(A,B)	0.0892532	0.0893122	0.0928595	0.0964292	0.100009	0.103503
MaxHD(A,B)	2.18784	1.54144	1.54456	1.5622	1.5926	1.61414
$h = 0.4$; Grid dimensions: $139 \times 229 \times 189$						
MeanHD(A,B)	0.0387838	0.0376088	0.0394284	0.0413076	0.0431461	0.0449941
MaxHD(A,B)	1.38006	0.524292	0.652781	0.77268	0.844057	0.903317
$h = 0.2$; Grid dimensions: $277 \times 457 \times 377$						
MeanHD(A,B)	0.0179671	0.0178176	0.0188439	0.0199281	0.0210222	0.0221271
MaxHD(A,B)	0.71095	0.12891	0.149643	0.171411	0.201191	0.24425
$h = 0.1$; Grid dimensions: $553 \times 913 \times 753$						
MeanHD(A,B)	0.0114735	0.0100917	0.0108925	0.0116861	0.0124757	0.013269
MaxHD(A,B)	0.751527	0.0742524	0.0754402	0.0858235	0.0997837	0.114525

Table 5.1: Values for the Hausdorff distance calculated by equations (5.1) for the surface reconstruction of the Sealer point cloud data. For the surface reconstruction the distance function d in (2.2) was calculated by the FSM algorithm.

δ	0.000	0.01	0.02	0.03	0.04	0.05
$h = 0.8$; Grid dimensions: $70 \times 115 \times 95$						
MeanHD(A,B)	0.0881375	0.0880087	0.0914124	0.0948419	0.0982602	0.101619
MaxHD(A,B)	2.19765	1.55754	1.55219	1.57347	1.59943	1.61448
$h = 0.4$; Grid dimensions: $139 \times 229 \times 189$						
MeanHD(A,B)	0.0389542	0.0358571	0.0377077	0.0396454	0.0415404	0.0434863
MaxHD(A,B)	1.47521	0.513041	0.634763	0.772745	0.853498	0.925827
$h = 0.2$; Grid dimensions: $277 \times 457 \times 377$						
MeanHD(A,B)	0.0172233	0.0161051	0.0172925	0.0185797	0.0198954	0.0212255
MaxHD(A,B)	1.06208	0.135259	0.147878	0.171798	0.218427	0.265875
$h = 0.1$; Grid dimensions: $553 \times 913 \times 753$						
MeanHD(A,B)	0.0101758	0.00852418	0.00962756	0.0107071	0.0117708	0.0128195
MaxHD(A,B)	0.917726	0.0774886	0.0882566	0.096055	0.1161	0.13332

Table 5.2: Values for the Hausdorff distance calculated by equations (5.1) for the surface reconstruction of the Sealer point cloud data. For the surface reconstruction the distance function d in (2.2) was calculated by VDT algorithm.

5.2 3D image segmentation supported by the point cloud

In this experiment we used the model (2.1) with advective velocity (2.6) for our initial goal and segmented the extra-embryonic tissues from the mouse embryo visualized in Figure 2.4 and Figure 2.5. To gain further information on the 3D shape of both tissues, we decided to reconstruct both ExE and VE. As mentioned in Chapter 2, the point cloud data for this microscopic image was obtained by rough manual identification of border points in both tissues. This manual process took around 2 hours for each tissue in the whole 3D image and was done by Antonia Weberling from the University of Cambridge. For this point cloud data, we applied linear interpolation between neighboring points in every 2D slice to get a more accurate representation of the 3D surfaces. We applied our algorithm to solve (2.1) and present the results in Figure 5.2 visualized with the supporting points.

In this example the, dimension of the 3D image was 512x512x58 voxels. The calculations were performed on a computer with an Intel(R) Core(TM) i7-5820 CPU 3.30GHz processor and 128 GB RAM. We measured the CPU times for both cases. For ExE part the calculation lasted 1017 seconds and for the VE part 1420 seconds. We also applied parallelization to our implementation with the utilization of the OpenMP library. For parallel calculations we used 6 threads on the mentioned computer. After this the calculation of the ExE part took 448 seconds and for the VE part 599 seconds.

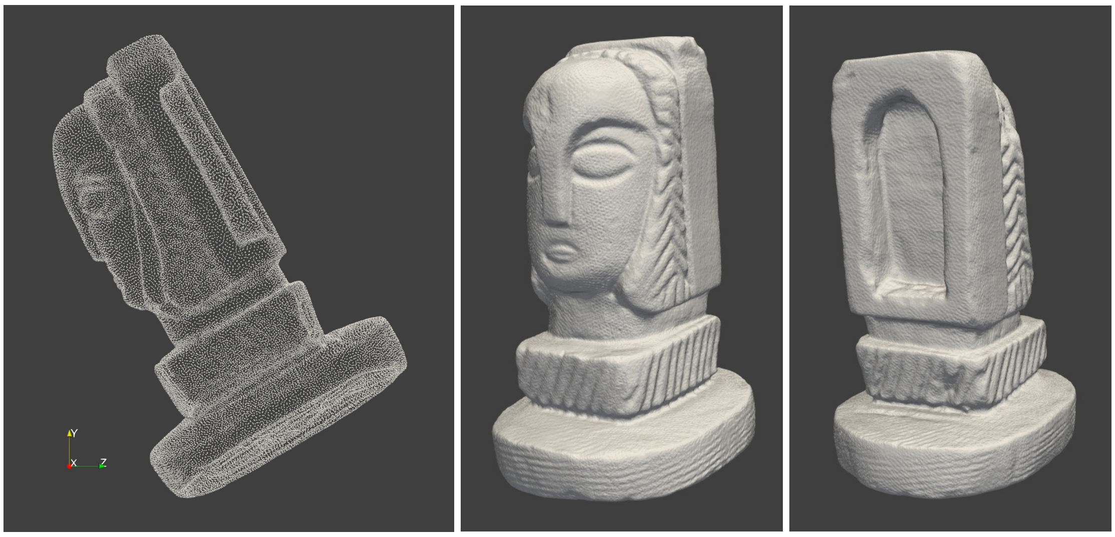

Figure 5.1: Visualizations for the Sealer data set. In the first picture we see the point cloud data. In the following pictures, we see the surface reconstruction result with $h = 0.1$ and $\delta = 0.01$. The 3D model is shown from different views.

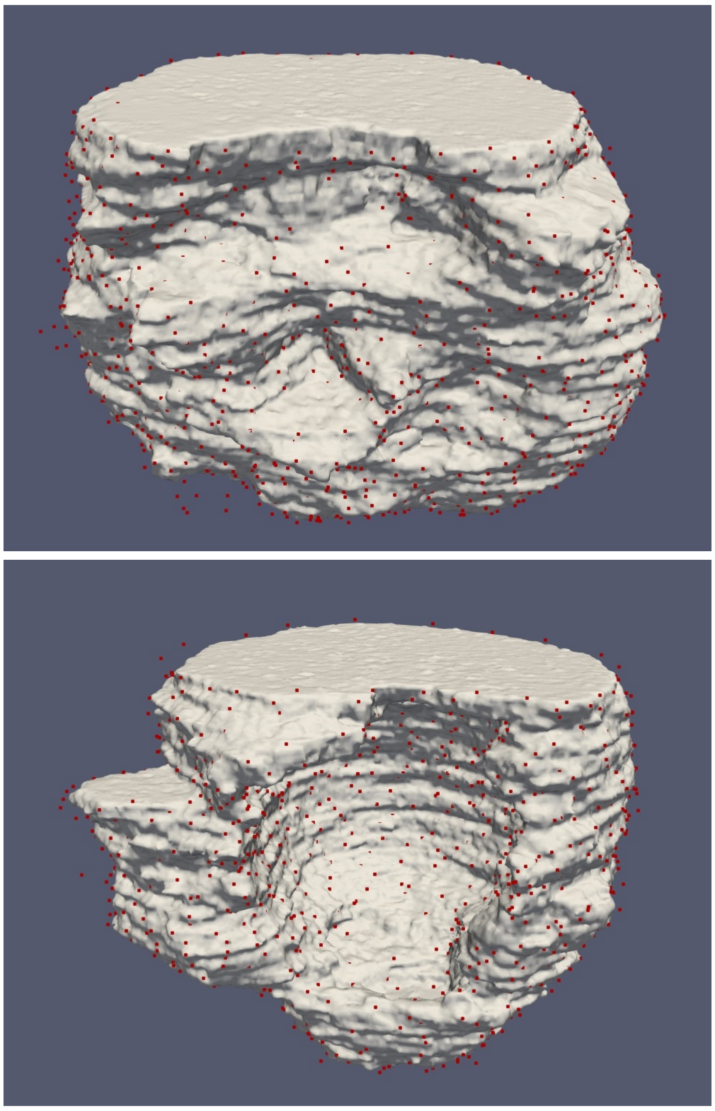

Figure 5.2: The reconstructed surface of extraembryonic ectoderm (ExE) and visceral endoderm (VE), visualized together with the point cloud.

References

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, p. 321–331, 1988.

[2] A. X. Falcao, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. de A. Lotufo, "User-steered image segmentation paradigms: Live wire and live lane," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 233–260, 1998.

[3] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[4] A. P. Dhawan, "Image segmentation," *Medical Image Analysis, IEEE*, pp. 229–264, 2010.

[5] A. Sarti, R. Malladi, and J. A. Sethian, "Subjective surfaces: A method for completing missing boundaries," *Proceedings of the National Academy of Sciences*, vol. 97, no. 12, pp. 6258–6263, 2000.

[6] S. Corsaro, K. Mikula, A. Sarti, and F. Sgallari, "Semi-implicit covolume method in 3D image segmentation," *SIAM Journal on Scientific Computing*, vol. 28, no. 6, pp. 2248–2265, 2006.

[7] C. Zanella, M. Campana, B. Rizzi, C. Melani, G. Sanguinetti, P. Bourgine, K. Mikula, N. Peyriéras, and A. Sarti, "Cells segmentation from 3-D confocal images of early zebrafish embryogenesis," *IEEE Transactions on Image Processing*, vol. 19, no. 3, pp. 770–781, 2010.

[8] E. Faure, T. Savy, B. Rizzi, C. Melani, O. Stašová, D. Fabreges, R. Špir, M. Hammons, R. Čunderlík, G. Recher, B. Lombardot, L. Duloquin, I. Colin, J. Kollár, S. Desnoulez, P. Affaticati, B. Maury, A. Boyreau, J. Y. Nief, P. Calvat, P. Vernier, M. Frain, G. Lutfalla, Y. Kergosien, P. Suret, M. Remešíková, R. Doursat, A. Sarti, K. Mikula, N. Peyriéras, and P. Bourgine, "A workflow to process 3D+time microscopy images of developing organisms and reconstruct their cell lineage," *Nature Communications*, vol. 7, no. 8674, 2016.

[9] P. Dorninger and N. Pfeifer, "A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds," *Sensors*, vol. 8, no. 11, pp. 7323–7343, 2008.

[10] R. E. Meouche, M. Rezoug, I. Hijazi, and D. Maes, "Automatic reconstruction of 3D building models from terrestrial laser scanner data," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-4, no. W1, pp. 7–12, 2013.

[11] H. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method," *Computer Vision and Image Understanding*, vol. 80, no. 3, pp. 295–319, 2000.

[12] J. Haličková and K. Mikula, "Level set method for surface reconstruction and its application in surveying," *Journal of Surveying Engineering*, vol. 142, no. 3, 2016.

[13] B. Kósa, J. Haličková-Brehovská, and K. Mikula, “New efficient numerical method for 3D point cloud surface reconstruction by using level set methods,” *Proceedings of Equadiff 2017 Conference*, pp. 387–396, 2017.

[14] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge: Cambridge University Press, 1999.

[15] S. Osher and R. Fedkiw, “Level set methods and dynamic implicit surfaces,” *Applied Mathematical Sciences*, vol. 153, 2003.

[16] H. Zhao, “A fast sweeping method for Eikonal equations,” *Mathematics of Computation*, vol. 74, pp. 603–627, 2005.

[17] P.-E. Danielsson, “Euclidean distance mapping,” *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227–248, 1980.

[18] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[19] M. Smíšek, *Analysis of 3D and 4D images of organisms in embryogenesis*. PhD thesis, Faculty of Civil Engineering, Slovak University of Technology Bratislava, 2015.

[20] OpenMP, “OpenMP C and C++ application program interface,” 2002.

[21] H. Zhao, S. Osher, and R. Fedkiw, “Fast surface reconstruction using the level set method,” *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, 2001.

[22] S. Osher and J. A. Sethian, “Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations,” *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1998.

[23] K. Mikula and A. Sarti, “Parallel co-volume subjective surface method for 3D medical image segmentation,” *Parametric and Geometric Deformable Models: An application in Biomaterials and Medical Imagery, Volume-II, Springer Publishers*, pp. 123–160, 2007.

[24] K. Mikula, N. Peyriéras, M. Remešíková, and A. Sarti, “3D embryogenesis image segmentation by the generalized subjective surface method using the finite volume technique,” *Finite Volumes for Complex Applications V: Problems and Perspectives*, p. 585–592, 2008.

[25] K. Mikula and M. Remešíková, “Finite volume schemes for the generalized subjective surface equation in image segmentation,” *Kybernetika*, vol. 45, no. 4, pp. 646–656, 2009.

[26] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[27] S. E. Harrison, B. Sozen, N. Christodoulou, C. Kyprianou, and M. Zernicka-Goetz, “Assembly of embryonic and extraembryonic stem cells to mimic embryogenesis in vitro,” *Science*, vol. 356, no. 6334, 2017.

[28] M. N. Shahbazi, A. Scialdone, N. Skorupska, A. Weberling, G. Recher, M. Zhu, A. Jedrusik, L. G. Devito, L. Noli, I. C. Macaulay, C. Buecker, Y. Khalaf, D. Ilic, T. Voet, J. C. Marioni, and M. Zernicka-Goetz, “Pluripotent state transitions coordinate morphogenesis in mouse and human embryos,” *Nature*, vol. 552, no. 7684, pp. 239–243, 2017.

[29] N. Christodoulou, C. Kyprianou, A. Weberling, R. Wang, G. Cui, G. Peng, N. Jing, and M. Zernicka-Goetz, “Sequential formation and resolution of multiple rosettes drive embryo remodeling after implantation,” *Nature Cell Biology*, vol. 20, pp. 1278–1289, 2018.

[30] M. N. Shahbazi and M. Zernicka-Goetz, "Deconstructing and reconstructing the mouse and human early embryo," *Nature Cell Biology*, vol. 20, pp. 878–887, 2018.

[31] S. Dyballa, T. Savy, P. Germann, K. Mikula, M. Remešíková, R. Špir, A. Zecca, N. Peyriéras, and C. Pujades, "Distribution of neurosensory progenitor pools during inner ear morphogenesis unveiled by cell lineage reconstruction," *eLife*, vol. 6, 2017.

[32] L. C. Evans and J. Spruck, "Motion of level sets by mean curvature. I," *Journal of Differential Geometry*, vol. 33, no. 3, pp. 635–681, 1991.

[33] B. Kósa, K. Mikula, M. Uba, A. Weberling, N. Christodoulou, and M. Zernicka-Goetz, "3D image segmentation supported by a point cloud," *Discrete and Continuous Dynamical Systems - Series S*, 2020.

[34] R. Eymard, A. Handlovičvá, and K. Mikula, "Study of a finite volume scheme for the regularised mean curvature flow level set equation," *IMA Journal of Numerical Analysis*, vol. 31, no. 3, pp. 813–846, 2011.

[35] B. Kósa, "3D point cloud surface reconstruction by using level set methods," Master's thesis, Faculty of Civil Engineering, Slovak University of Technology Bratislava, 2015.

[36] M. W. Jones, J. A. Baerentzen, and M. Sramek, "3D distance fields: a survey of techniques and applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 581–599, 2006.

[37] R. Fabbri, L. da F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys*, vol. 40, no. 1, pp. 2:1–2:44, 2008.

[38] P.-O. Persson, *Mesh Generation for Implicit Geometries*. PhD thesis, Department of Mathematics, Massachusetts Institute Of Technology, 2005.

[39] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, p. 163–169, 1987.

[40] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. 2018. https://vtk.org/documentation/.

Zoznam publikačnej činnosti

KÓSA, Balázs - MIKULA, Karol - UBA, Markjoe O. - WEBERLING, Antonia - CHRISTODOULOU, Neophytos - ZERNICKA-GOETZ, Magdalena. 3D image segmentation supported by a point cloud. Accepted by *Discrete & Continuous Dynamical Systems - S (DCDS-S), 16. May 2020* ; DOI: 10.3934/dcdss.2020351

PARK, Seol Ah - SIPKA, Tamara - KRIVÁ, Zuzana - AMBROZ, Martin - KOLLÁR, Michal - KÓSA, Balázs - NGUYEN-CHI, Mai - LUTFALLA, Georges - MIKULA, Karol. Macrophage image segmentation by thresholding and subjective surface method. In *Tatra Mountains Mathematical Publications*. No. 75 (2020), s. 103-120. ISSN 1210-3195 (2019: 0.214 - SJR, Q4 - SJR Best Q). V databáze: SCOPUS: 2-s2.0-85085758015 ; DOI: 10.2478/tmmp-2020-0007.

KÓSA, Balázs - MIKULA, Karol. Fast numerical method for 3D point cloud surface reconstruction. In *Advances in Architectural, Civil and Environmental Engineering [elektronický zdroj] : 27th Annual PhD Student Conference on Applied Mathematics, Applied Mechanics, Geodesy and Cartography, Landscaping, Building Technology, Theory and Structures of Buildings, Theory and Structures of Civil Engineering Works, Theory and Environmental Technology of Buildings, Water Resources Engineering. 25. October 2017, Bratislava, Slovakia*. 1. vyd. Bratislava : Spektrum STU, 2017, CD-ROM, s. 42-46. ISBN 978-80-227-4751-6.

KÓSA, Balázs - HALIČKOVÁ-BREHOVSKÁ, Jana - MIKULA, Karol. New efficient numerical method for 3D point cloud surface reconstruction by using level set methods. In *Proceedings of Equadiff 2017 Conference [elektronický zdroj] : July 24-28, 2017, Bratislava, Slovakia*. 1. vyd. Bratislava : Spektrum STU, 2017, online, s. 387-396. ISBN 978-80-227-4757-8. V databáze: WOS: 000426796700044.

KÓSA, Balázs - MIKULA, Karol. Surface reconstruction by using 3D point cloud and image information. In *Advances in Architectural, Civil and Environmental Engineering [elektronický zdroj] : 28th Annual PhD Student Conference on Applied Mathematics, Applied Mechanics, Building Technology, Geodesy and Cartography, Landscaping, Theory and Environmental Technology of Buildings, Theory and Structures of Buildings, Theory and Structures of Civil Engineering Works, Water Resources Engineering. October 24th 2018, Bratislava*. 1. vyd. Bratislava : Spektrum STU, 2018, CD-ROM, s. 27-31. ISBN 978-80-227-4864-3.