**SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA**

**FACULTY OF CIVIL ENGINEERING**

---

**Markjoe Olunna UBA**

Dissertation Thesis Abstract

# 3D+Time Image Segmentation Methods

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

**Bratislava 2021**

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

**Submitter:** **Markjoe Olunna UBA**
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:** **Reader 1**
Department of Mathematics and Descriptive Geometry
Faculty of Civil Engineering, STU, Bratislava.

**Reader 2**
Reader's department
Reader's university

**Reader 3**
Reader's department
Reader's university

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 Uncík, PhD.**
Dean of Faculty of Civil Engineering

# Abstract

In this thesis, we introduce and study a novel segmentation method for 4D images based on surface evolution governed by a nonlinear partial differential equation, the generalized subjective surface equation. The new method uses 4D digital image information and information from a thresholded 4D image in a local neighborhood. Thus, the 4D image segmentation is accomplished by defining the edge detector function's input as the weighted sum of the norm of gradients of presmoothed 4D image and norm of presmoothed thresholded 4D image in a local neighborhood. Additionally, we design and study a numerical method based on the finite volume approach for solving the new model. The reduced diamond cell approach is used for approximating the gradient of the solution. We use a semi-implicit finite volume scheme for the numerical discretization and show that our numerical scheme is unconditionally stable. The new 4D method was tested on artificial data and applied to real data representing 3D+time microscopy images of cell nuclei within the zebrafish pectoral fin and hind-brain. In a real application, processing 3D+time microscopy images, e.g. with dimensions $567 \times 577 \times 147 \times 70$, amounts to solving a linear system with 3 366 466 110 unknowns and requires over 1000 GB of memory; thus, it may not be possible to process these images on a serial machine without parallel implementation utilizing the MPI. Consequently, we develop and present in the thesis OpenMP and MPI parallel implementation of designed algorithms. Finally, we include cell tracking results to show how our new method serves as a basis for finding trajectories of cells during embryogenesis.



**Keywords: Image segmentation, subjective surface method, level set method, finite volume method, semi-implicit scheme, cell microscopy images, zebrafish.**



# Abstrakt

V dizertačnej práci predstavujeme a študujeme novú segmentačnú metódu pre 4D obrazy, založenú na vývoji plôch, ktorý je riadený nelineárnou parciálnou diferenciálnou rovnicou, tzv. zovšeobecnenou rovnicou subjektívnych plôch. Nová metóda využíva jednak informácie o 4D digitálnom obraze ako aj informácie z lokálne prahovaného 4D obrazu. Segmentácia je teda riadená vstupom funkcie hranového detektora ako váženého súčtu normy gradientu vyhladeného 4D obrazu a normy gradientu vyhladeného lokálne prahovaného 4D obrazu. Ďalej práca predstavuje a študuje numerickú metódu na riešenie zovšeobecneného 4D modelu založenú na metóde konečných objemov. Na aproximáciu gradientu riešenia sa používa tzv. redukovaná diamond-cell metóda. Na časovú diskretizáciu bola použitá semi-implicitná schéma, o ktorej ukazujeme, že je bezpodmienečne stabilná. Navrhnutá 4D metóda bola testovaná na umelých dátach a následne bola aplikovaná na reálne dáta predstavujúce 3D videá (4D obrazy) mikroskopických snímok jadier buniek v prsnej plutve a zadnom mozgu embrya rybky zebrafish. V reálnych situáciach, ako je napríklad spracovanie 3D videí mikroskopických obrazov s rozmermi $567 \times 577 \times 147 \times 70$, navrhnutá metóda vedie na riešenie lineárneho systému rovníc s 3 366 466 110 neznámymi, čo vyžaduje viac ako 1000 GB operačnej pamäte počítača. Je zrejmé, že takéto úlohy nie možné vyriešiť bez paralelnej implementácie. Preto v dizertačnej pr8ci vytvárame a prezentujeme OpenMP a MPI paralelné implementácie navrhnutých algoritmov. Na záver uvádzame výsledky trekingu buniek, ktoré ukazujú, ako naša nová metóda 4D segmentácie obrazu slúži ako základ pre nájdenie trajektórií buniek počas embryogenézy.



**Kľúčové slová: segmentácia obrazu, metóda subjektívnych plôch, level set metóda, metóda konečných objemov, semi-implicitná schéma, mikroskopické obrazy buniek, zebrafish.**

# Contents

# 1 Introduction

Image segmentation aims to partition the image domain into "meaningful" components in image processing and computer vision. Segmentation amounts to finding curves in two-dimensional (2D) images, computing surfaces in three-dimensional (3D) images, and computing evolving surfaces in four-dimensional (4D) images. It is well known that image segmentation is one of the fundamental and most studied problems in image processing. Consequently, there are several approaches to image segmentation in literature [7, 9, 10, 11, 15, 30, 33, 36, 41]. However, in this thesis, we will study and use the level set method for image segmentation. The level set method is the evolution of curves and surfaces by means of a dynamical embedding function (often referred to as the level set function). In other words, the level set method is the implicit representation of boundaries, curves (or surfaces). Furthermore, in the level set formulation, the subjective surface (SUBSURF) method for image segmentation will be the focus of this thesis. This method, in the context of image processing, was introduced in [34, 35], studied and applied in several biomedical research [4, 12, 13, 17, 18, 19, 24, 25, 27, 34, 35, 44]. SUBSURF segmentation method is based on the idea of evolution of segmentation function governed by a nonlinear diffusion equation [4, 34, 35, 43] which can be understood as an advection-diffusion model [24]. Hence, a segmentation seed (the starting point that determines the approximate position of an object in the image) is usually needed to segment an object. Then an initial segmentation function $u^0(x)$ is constructed with reference to the segmentation seed. Finally, this segmentation function is allowed to evolve to the final state following the SUBSURF model. Ideally, the evolution process ends up with a function whose isosurfaces all have the object's shape that is intended to be segmented. At each time step in the evolution process, the values of the segmentation function are locally rescaled to interval $[0, 1]$. After the last time step in the evolution, the contour 0.5 is chosen as the approximate boundary of the segmented object. Significant research efforts have been made in the study and application of the SUBSURF model. The idea of following the interfaces in immiscible fluid flow by the Eulerian approach was proposed by Dervieux and Thomasset [5, 6], and Osher and Sethian [29] then introduced the general level set methodology.

In this thesis, we introduce a generalization of the classical SUBSURF model. This generalization is due to the fact that in real applications where the object intended to be segmented possesses internal structures or edges, it is usually challenging to obtain optimal results using the classical SUBSURF segmentation approach. The reason is that this approach works with edge information throughout the segmentation process. Hence, edges within the internal structures in an object of interest are also respected during segmentation. To overcome the effect of the internal structures or edges, we introduced thresholding of image intensity values within a ball of appropriate radius around the object center. This local thresholding serves to eliminate the internal structures or edges. Additionally, we combined the information obtained from thresholding and original image intensities to get an accurate final segmentation result. Unlike the works in [19, 24, 25, 26, 27, 28], at each segmentation step, the values of the segmentation function are rescaled to interval $[0, 1]$. Finally, considering that the computation task involving 4D microscopy images requires solving a linear system with several billion unknowns, we developed an efficient parallel implementation of the method for the generalized model in the case of 4D images.

The remaining part of this work is organized as follows: Chapter 2 provides an overview of the SUBSURF method and some numerical examples of the application of the SUBSURF segmentation method to artificial and real data. Chapter 3 contains the details of the proposed generalization of the classical method. It contains the new model combining thresholded image information and original image data, the numerical scheme for solving the model, the stability of the numerical scheme, experimental order of convergence, an overview of the computer implementation, and parallel implementation using OpenMP and MPI. In Chapter 4, the details of the application of the new segmentation algorithm to 3D cell image segmentation were presented. These details include numerical experiments with cell nuclei and membranes images. Furthermore, the application of the new segmentation algorithm to 4D image segmentation was presented in Chapter 5. Finally, cell tracking based on the result of the new segmentation algorithm was presented in Chapter 6.

## 2 The SUBSURF segmentation method

The SUBSURF segmentation method is based on the idea of evolution of segmentation function, which is governed by an advection-diffusion model. The SUBSURF model was proposed by Sarti, Malladi, and Sethian [35] and is given by

$$\frac{\partial u}{\partial t} = |\nabla u| \nabla \cdot \left( g^0 \frac{\nabla u}{|\nabla u|} \right) \text{ in } (0,T] \times \Omega, \tag{1}$$

where $u(t,x)$ is the unknown segmentation function, $g^0 = g(|\nabla G_\sigma * I^0|)$, $I^0$ is image to be segmented, and $g$ is the Perona-Malik function [32], and $G_\sigma$ is a smoothing kernel. Equation (1) is accompanied by Dirichlet boundary conditions

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

or Neumann boundary conditions

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

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

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

Dirichlet boundary condition is used in the image segmentation, and without loss of generality $u^D = 0$ may be assumed. The zero Neumann boundary conditions are often used in computations involving interface motions in free boundary problems or morphological image smoothing (see, e.g., [4] and references therein).

In computations involving model (1), one possible choice of initial segmentation function, $u^0$, is the following function

$$u^0(x) = \frac{1}{|s-x| + v}\ , \tag{5}$$

where $s$ corresponds to the focus point, and $\frac{1}{v}$ gives a maximum of $u^0$ whose peak is centered in a "focus point" inside the segmented object [27]. This function can also be considered only at a circle with center $s$ and radius $R$. Outside this circle, the value of $u^0$ is taken to be $\frac{1}{R+v}$. If (1) is accompanied by zero Dirichlet boundary condition, then finally $\frac{1}{R+v}$ is subtracted from such peak-like profile. In the case of small objects, a smaller choice of value for $R$ can be used to speed up computations. Also, $u^0(x) = 1 - \frac{|x-s|}{R}$ is another possible choice of initial segmentation function.

We now present some numerical and illustrative examples to demonstrate the application of the SUBSURF model to image segmentation. The model (1) for surface reconstruction was tested on artificial examples and applied to real data representing 3D microscopy images of cell nuclei (see also Section 4.1). For the first numerical experiment, a sphere and a sphere with holes were generated, as can be seen in the first column of Figure 1. Afterward, model (1) was used to reconstruct the shapes of these spheres, and the results obtained after reconstruction are shown in the second column of Figure 1. In the second experiment, the model (1) was applied to real data representing 3D microscopy images of cell nuclei and membrane. In the first column of Figures 2, 3, and 4, 3D image of these microscopy images are shown, whereas, in the second column of Figures 2, 3, and 4 the results obtained after application of model (1) are shown and colored in black.

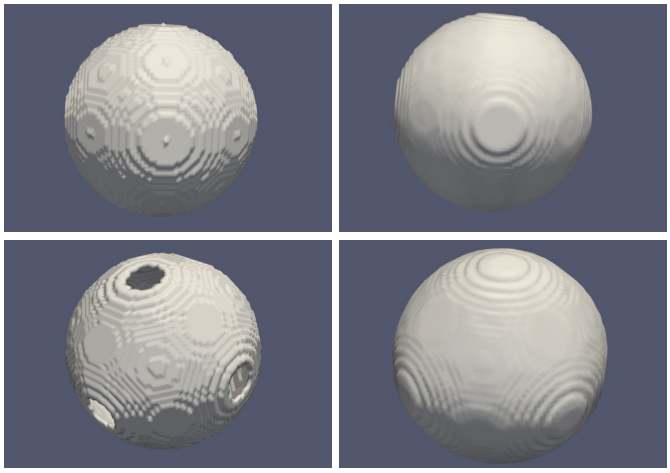

Figure 1: First column shows a solid sphere and sphere with six symmetric holes, whereas the second column shows results of the segmentation after the application of the model (1).

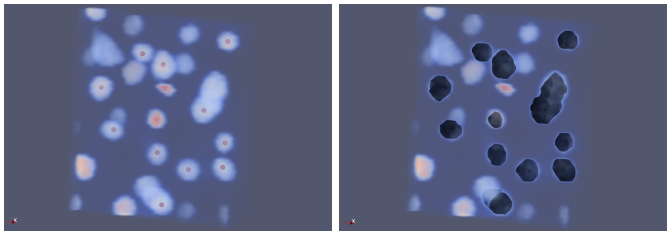

Figure 2: First column of this figure shows the 3D microscopy image of cell nuclei together with approximate cell centers (these are points marked in red color), whereas the second column shows the 3D image of microscopy image of cell nuclei after application of (1) to the cells with centers marked in red color.

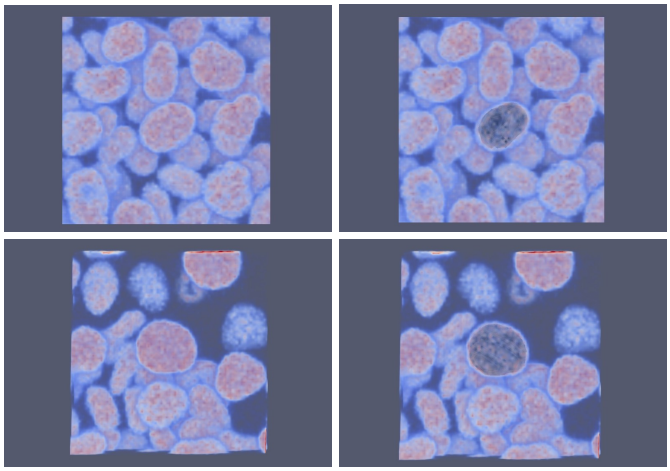

Figure 3: First column of this figure shows the 3D microscopy images of cell nuclei to be segmented, whereas the second column shows the result after application of the model (1); the cell nuclei of interest and their corresponding results after segmentation (which are depicted in black) are located approximately at the center of each picture.

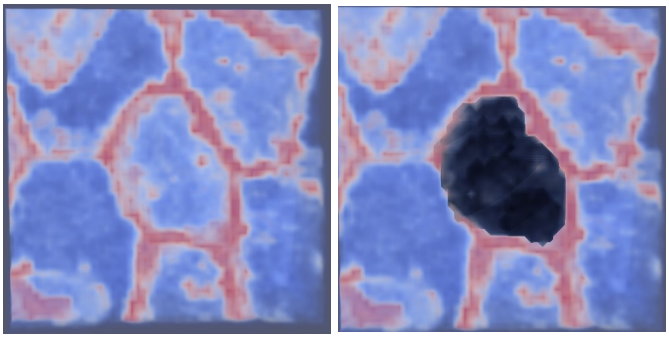

Figure 4: First column of this figure shows the 3D microscopy image of cell membrane to be segmented, whereas the second column shows the result after application of the model (1); the cell membrane of interest and its corresponding result after segmentation (which are depicted in black) is located approximately at the center of each picture.

## 3 4D Image Segmentation Algorithm

In the previous chapter, we presented the SUBSURF segmentation method and some of its successful applications. However, in many real applications where the object to be segmented has internal structures or edges, it is usually challenging to obtain optimal results using the SUBSURF segmentation approach (see, e.g., Figures 5, 6, and 7). The reason is that this approach works with edge information throughout the segmentation process. Hence, edges within the internal structures in an object of interest are also respected during segmentation. For instance, Figures 5, 6, and 7 show the results of segmentation using the classical SUBSURF approach for 3D images of zebrafish cell nuclei, cell membrane, and mouse embryo cell nuclei, respectively. The results of segmentation in these figures are colored in black, and the shapes of interest are located approximately at the center of each image.

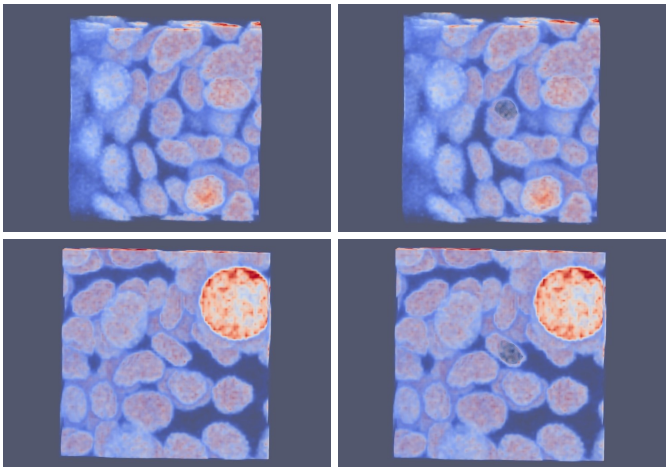

Figure 5: First column of this figure shows the 3D microscopy images of cell nuclei to be segmented, whereas the second column shows the result of segmentation using the classical SUBSURF approach.

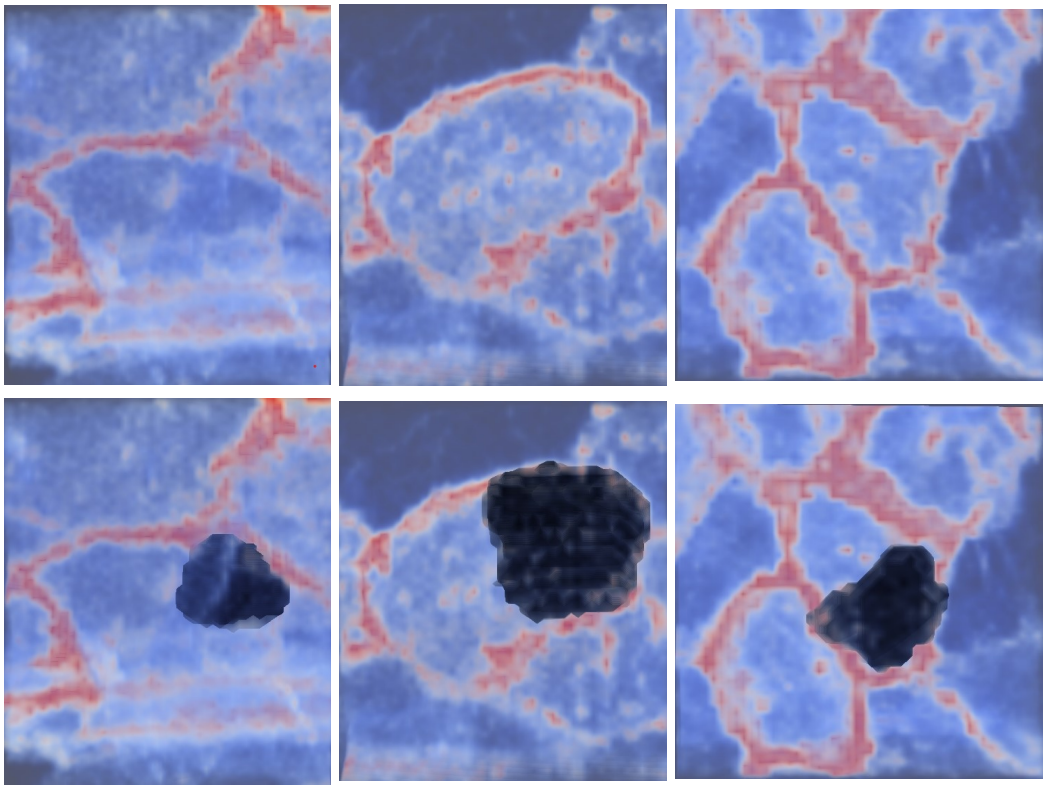

Figure 6: First row of this figure shows the 3D microscopy images of cell membranes to be segmented, while the second row shows the result of segmentation using the classical SUBSURF approach.

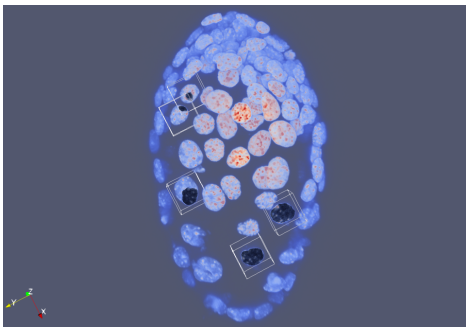

Figure 7: This figure shows the 3D microscopy image of mouse embryo cell nuclei segmentation result using classical SUBSURF approach.

In this chapter, to overcome the effect of the internal structures or edges, we introduce the thresholding of values within a ball of appropriate radius around the object center. This local thresholding serves to eliminate the internal structures or edges. Finally, we combine the information given by thresholding and original image intensities to get a segmentation result.

In [20], the first numerical scheme for 4D image segmentation based on generalized SUBSURF model (**??**) was introduced. In this thesis, we introduce and study a 4D numerical scheme for the solution of the new 4D segmentation model (6) presented in the next section. This numerical method is based on the finite volume approach. The stability of the numerical scheme is also presented. Additionally, we introduced the local rescaling of the values of the segmentation function at each segmentation step to the interval $[0, 1]$. Furthermore, the application of the new model to 3D and 4D image segmentation is studied. OpenMP and MPI parallelizations are employed for efficient computer implementation. For example, in the real application presented in the third and fourth experiments of Chapter 5, to process the 3D+time microscopy images, with dimensions $567 \times 577 \times 147 \times 70$, 1012 GB of memory was needed. This clearly shows that it may not be possible to process these images on a serial machine without parallel implementation utilizing the MPI.

## 3.1 Mathematical model

Let $I^0 : \mathbb{O} \subset \mathbb{R}^4 \longrightarrow \mathbb{R}$, be the intensity function of a 4D image where $\mathbb{O} = \Omega \times [0, \theta_F]$, $\Omega \subset \mathbb{R}^3$, and $\theta_F$ represents the final real video time. For each real video time $\theta \in [0, \theta_F]$, let $C_\theta = \{c_m^\theta\}_{m=1}^{N_\theta}$ denote the set of "centers", where $c_m^\theta$ and $N_\theta$ represent the $m^{th}$ center and the total number of centers, respectively, at time $\theta$. Furthermore, for each $c_m^\theta \in C_\theta$ and $\theta \in [0, \theta_F]$, let $\alpha_m^\theta = \min\limits_{y \in B_m^\theta(c_m^\theta, r)} I^0(y, \theta)$, $\beta_m^\theta = \max\limits_{y \in B_m^\theta(c_m^\theta, r)} I^0(y, \theta)$, where $B_m^\theta(c_m^\theta, r)$ is the $m^{th}$ ball at time $\theta$ with radius $r$ centered at $c_m^\theta$, a given point ("center") inside the object to be segmented. Then the threshold value (which is used for local thresholding) may be chosen as $TH_m^\theta = \lambda\ \alpha_m^\theta + (1 - \lambda)\ \beta_m^\theta$, $\lambda \in [0, 1]$ and the ball radius may be chosen with respect to the approximate size of the object to be segmented. So, the idea (of local thresholding) is to set all intensity values in the local neighborhood of center $c_m^\theta$ to $\beta_m^\theta$ if they are above $TH_m^\theta$ and $\alpha_m^\theta$ otherwise. Hence, the 4D image intensity of the thresholded image within a ball of radius $r$ is defined by

$$I^{TH}(y, \theta) = \begin{cases} \beta_m^\theta, \ I^0(y, \theta) \geq TH_m^\theta, \\ \alpha_m^\theta, \ otherwise, \end{cases}$$

where $y \in B_m^\theta(c_m^\theta, r)$.

Our new method is based on solution of the following generalized SUBSURF equation

$$\frac{\partial u}{\partial t} = |\nabla u| \nabla \cdot \left( G^0 \frac{\nabla u}{|\nabla u|} \right) \text{ in } (0, T] \times \mathbb{O}, \tag{6}$$

where $u(t, x)$ is the unknown segmentation function, $x \in \mathbb{O}$, $t \in [0, T]$ is a "segmentation time," $G^0 = g(\delta |\nabla G_\sigma * I^0| + \vartheta |\nabla G_\sigma * I^{TH}|)$, $g$ is the Perona-Malik function, $\delta, \vartheta \in [0, 1]$ determine the influence of information obtained from thresholded and original image intensities, $G_\sigma$ is the smoothing kernel, e.g., the Gaussian function, $\nabla G_\sigma * I^0$ and $\nabla G_\sigma * I^{TH}$ are presmoothing of $I^0$ and $I^{TH}$ by convolution with $G_\sigma$, respectively. It is well known, see e.g., [42, 16], that the convolution with Gaussian function $G_\sigma$ is equivalent to solving the linear heat equation (LHE) obtained by setting $G^0 \equiv 1$ in equation (6), with $t = \frac{1}{2}\sigma^2$. Thus, the presmoothing of $I^0$ and $I^{TH}$ by convolution with Gaussian function is realized by solving the LHE. Equation (6) is accompanied by Dirichlet boundary conditions

$$u(t, x) = u^D \text{ on } [0, T] \times \partial\mathbb{O}, \tag{7}$$

and with the initial condition

$$u(0, x) = u^0(x) \text{ in } \mathbb{O}. \tag{8}$$

Without loss of generality, $u^D = 0$ is assumed. The initial condition in 4D is constructed using equation (5), where $x \in \Omega$.

## 3.2 Numerical discretization

In this section, the details of time and space discretization is presented.

### 3.2.1 Time discretization

For time discretization of (6), semi-implicit approach which guarantees unconditional stability is used. Suppose that the (6)–(8) is solved in time interval $I = [0, T]$ and $N$ equal number of time steps. If $\tau = \frac{T}{N}$ denotes the time step, then the time discretization of (6) is given by

$$\frac{1}{\sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}} \frac{u^n - u^{n-1}}{\tau} = \nabla \cdot \left( G^0 \frac{\nabla u^n}{\sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}} \right), \tag{9}$$

where $\varepsilon$ is the regularization factor (Evans - Spruck [8]), $u^0$ is given initial segmentation function, and $u^n$, $n = 1, \cdots, N$ is the solution of the model in segmentation time step $n$.

### 3.2.2 Space discretization

For space discretization, we start with introduction of some notations which will be used subsequently. We have adopted similar notations as those used in [24] and [19]. Let $\mathcal{T}_h$ denote finite volume mesh containing the doxels of 4D image, while $V_{ijkl}$, $i = 1, \cdots, N_1$, $j = 1, \cdots, N_2$, $k = 1, \cdots, N_3$, $l = 1, \cdots, N_4$ denote each finite volume. For each $V_{ijkl} \in \mathcal{T}_h$, let $h_1$, $h_2$, $h_3$, $h_4$ be the size of the volumes in $x_1$, $x_2$, $x_3$, $x_4$ direction. Let the volume of $V_{ijkl}$ and its barycenter be denoted by $m(V_{ijkl})$ and $c_{ijkl}$, respectively. Let the approximate value of $u^n$ in $c_{ijkl}$ be denoted by $u^n_{ijkl}$. For every $V_{ijkl} \in \mathcal{T}_h$, we have the following definitions:

- $N_{ijkl} = \{(p, q, r, s) : p, q, r, s \in \{-1, 0, 1\},\ |p| + |q| + |r| + |s| = 1\}$
- $P_{ijkl} = \{(p, q, r, s) : p, q, r, s \in \{-1, 0, 1\},\ |p| + |q| + |r| + |s| = 2\}$

For each $(p, q, r, s) \in N_{ijkl}$, denote the line connecting the center of $V_{ijkl}$ and the center of its neighbor $V_{i+p,j+q,k+r,l+s}$ by $\sigma^{pqrs}_{ijkl}$ and its length $m(\sigma^{pqrs}_{ijkl})$. We denote the sides, its measure, and normal in coordinate directions of the finite volume $V_{ijkl}$ by $e^{pqrs}_{ijkl}$, $m(e^{pqrs}_{ijkl})$, and $\nu^{pqrs}_{ijkl}$, respectively. Furthermore, for each $(p, q, r, s) \in P_{ijkl}$, $y^{pqrs}_{ijkl}$ is defined by

$$\begin{aligned}
y^{pq00}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i+p,j,k,l} + c_{i,j+q,k,l} + c_{i+p,j+q,k,l}\Big),\\
y^{0qr0}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i,j+q,k,l} + c_{i,j,k+r,l} + c_{i,j+q,k+r,l}\Big),\\
y^{00rs}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i,j,k+r,l} + c_{i,j,k,l+s} + c_{i,j,k+r,l+s}\Big),\\
y^{p0r0}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i+p,j,k,l} + c_{i,j,k+r,l} + c_{i+p,j,k+r,l}\Big),\\
y^{p00s}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i+p,j,k,l} + c_{i,j,k,l+s} + c_{i+p,j,k,l+s}\Big),\\
y^{0q0s}_{ijkl} &= \frac{1}{4}\Big(c_{ijkl} + c_{i,j+q,k,l} + c_{i,j,k,l+s} + c_{i,j+q,k,l+s}\Big).
\end{aligned}$$

The approximate value of $u^{n-1}$ in $y^{pqrs}_{ijkl}$, with $(p, q, r, s) \in P_{ijkl}$, is denoted by $u^{pqrs}_{ijkl}$; the time index is omitted because only the values from the time level $n-1$ will be needed at these points.

With these notations, integration of (9) over finite volume $V_{ijkl}$ yields

$$\int\limits_{V_{ijkl}} \frac{1}{\sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}} \frac{u^n - u^{n-1}}{\tau} dx = \int\limits_{V_{ijkl}} \nabla \cdot \left( G^0 \frac{\nabla u^n}{\sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}} \right) dx. \tag{10}$$

Let the average value of $A_\varepsilon = \sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}$ in finite volume $V_{ijkl}$ be denoted by $\bar{A}^{n-1}_{\varepsilon,ijkl}$. If we consider the fact that $u^n$ and $u^{n-1}$ are asummed to be piecewise constant over the finite volume mesh on the left hand side of (10) and using the divergence theorem on the right hand side of (10), we obtain

$$\frac{m(V_{ijkl})}{\bar{A}^{n-1}_{\varepsilon,ijkl}} \frac{u^n_{ijkl} - u^{n-1}_{ijkl}}{\tau} = \sum_{N_{ijkl}} \int\limits_{e^{pqrs}_{ijkl}} G^0 \frac{\nabla u^n}{\sqrt{\varepsilon^2 + |\nabla u^{n-1}|^2}} \cdot \nu^{pqrs}_{ijkl} dS, \tag{11}$$

where beneath the summation sign, we used just $N_{ijkl}$ instead of $(p, q, r, s) \in N_{ijkl}$ to simplify notation. If we approximate normal derivative $\nabla u^n \cdot \nu^{pqrs}_{ijkl}$ by $(u^n_{i+p,j+q,k+r,l+s} - u^n_{ijkl})/m(\sigma^{pqrs}_{ijkl})$ and define $A^{pqrs;n-1}_{\varepsilon,ijkl}$ and $G^{pqrs}_{ijkl}$ to be the average of $A_\varepsilon$ and $G^0$ on $e^{pqrs}_{ijkl}$ then (11) reduces to

$$m(V_{ijkl}) \frac{u^n_{ijkl} - u^{n-1}_{ijkl}}{\tau} = \bar{A}^{n-1}_{\varepsilon,ijkl} \sum_{N_{ijkl}} m(e^{pqrs}_{ijkl}) G^{pqrs}_{ijkl} \frac{u^n_{i+p,j+q,k+r,l+s} - u^n_{ijkl}}{A^{pqrs;n-1}_{\varepsilon,ijkl} m(\sigma^{pqrs}_{ijkl})}. \tag{12}$$

Equation (12) can be rewritten as

$$u^n_{ijkl} = u^{n-1}_{ijkl} + \frac{\tau}{m(V_{ijkl})} \bar{A}^{n-1}_{\varepsilon,ijkl} \sum_{N_{ijkl}} m(e^{pqrs}_{ijkl}) G^{pqrs}_{ijkl} \frac{u^n_{i+p,j+q,k+r,l+s} - u^n_{ijkl}}{A^{pqrs;n-1}_{\varepsilon,ijkl} m(\sigma^{pqrs}_{ijkl})}, \tag{13}$$

which can be written in the form of system of equations

$$\begin{aligned}\left(1+\frac{\tau}{m(V_{ijkl})}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} G^{pqrs}_{ijkl}\frac{m(e^{pqrs}_{ijkl})}{A^{pqrs;n-1}_{\varepsilon,ijkl}m(\sigma^{pqrs}_{ijkl})}\right)u^{n}_{ijkl} \quad - \\ \frac{\tau}{m(V_{ijkl})}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} G^{pqrs}_{ijkl}\frac{m(e^{pqrs}_{ijkl})}{A^{pqrs;n-1}_{\varepsilon,ijkl}m(\sigma^{pqrs}_{ijkl})}u^{n}_{i+p,j+q,k+r,l+s} \quad = \quad u^{n-1}_{ijkl},\end{aligned} \tag{14}$$

$i=1,\cdots,N_1$, $j=1,\cdots,N_2$, $k=1,\cdots,N_3$, and $l=1,\cdots,N_4$. The average values $G^{pqrs}_{ijkl}$, $A^{pqrs;n-1}_{\varepsilon,ijkl}$, and $\bar{A}^{n-1}_{\varepsilon,ijkl}$ either in doxels or on doxel sides are determined using the reduced diamond cell strategy (see [19]) adapted to 4D.

In the sense of this reduced diamond cell approach, the approximate values of $u^{n-1}$ are obtained in the points $y^{pqrs}_{ijkl}$. These values are defined for each $(p,q,r,s)\in P_{ijkl}$ by

$$\begin{aligned}u^{pq00}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i+p,j,k,l}+u^{n-1}_{i,j+q,k,l}+u^{n-1}_{i+p,j+q,k,l}\Big),\\ u^{0qr0}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i,j+q,k,l}+u^{n-1}_{i,j,k+r,l}+u^{n-1}_{i,j+q,k+r,l}\Big),\\ u^{00rs}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i,j,k+r,l}+u^{n-1}_{i,j,k,l+s}+u^{n-1}_{i,j,k+r,l+s}\Big),\\ u^{p0r0}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i+p,j,k,l}+u^{n-1}_{i,j,k+r,l}+u^{n-1}_{i+p,j,k+r,l}\Big),\\ u^{p00s}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i+p,j,k,l}+u^{n-1}_{i,j,k,l+s}+u^{n-1}_{i+p,j,k,l+s}\Big),\\ u^{0q0s}_{ijkl} &= \frac{1}{4}\Big(u^{n-1}_{ijkl}+u^{n-1}_{i,j+q,k,l}+u^{n-1}_{i,j,k,l+s}+u^{n-1}_{i,j+q,k,l+s}\Big).\end{aligned}$$

The components of the averaged gradient on $e^{pqrs}_{ijkl}$, $(p,q,r,s)\in N_{ijkl}$, are approximated by 2D diamond cell approach which use the values $u^{pqrs}_{ijkl}$ given above (see also [19]). Additionally, approximation of the gradient on the face $e^{pqrs}_{ijkl}$ is denoted by $\nabla^{pqrs}u^{n-1}_{ijkl}$. This implies that

$$\begin{aligned}\nabla^{p000}u^{n-1}_{ijkl}=\frac{1}{m(e^{p000}_{ijkl})}\int\limits_{e^{p000}_{ijkl}}\nabla u^{n-1}dx \quad &\approx \Big(p(u^{n-1}_{i+p,j,k,l}-u^{n-1}_{ijkl})/h_1,(u^{p,1,0,0}_{ijkl}-u^{p,-1,0,0}_{ijkl})/h_2,\\ &\quad (u^{p,0,1,0}_{ijkl}-u^{p,0,-1,0}_{ijkl})/h_3,(u^{p,0,0,1}_{ijkl}-u^{p,0,0,-1}_{ijkl})/h_4\Big),\end{aligned}$$

$$\begin{aligned}\nabla^{0q00}u^{n-1}_{ijkl}=\frac{1}{m(e^{0q00}_{ijkl})}\int\limits_{e^{0q00}_{ijkl}}\nabla u^{n-1}dx \quad &\approx \Big((u^{1,q,0,0}_{ijkl}-u^{-1,q,0,0}_{ijkl})/h_1,q(u^{n-1}_{i,j+q,k,l}-u^{n-1}_{ijkl})/h_2,\\ &\quad (u^{0,q,1,0}_{ijkl}-u^{0,q,-1,0}_{ijkl})/h_3,(u^{0,q,0,1}_{ijkl}-u^{0,q,0,-1}_{ijkl})/h_4\Big),\end{aligned}$$

$$\begin{aligned}\nabla^{00r0}u^{n-1}_{ijkl}=\frac{1}{m(e^{00r0}_{ijkl})}\int\limits_{e^{00r0}_{ijkl}}\nabla u^{n-1}dx \quad &\approx \Big((u^{1,0,r,0}_{ijkl}-u^{-1,0,r,0}_{ijkl})/h_1,(u^{0,1,r,0}_{ijkl}-u^{0,-1,r,0}_{ijkl})/h_2,\\ &\quad r(u^{n-1}_{i,j,k+r,l}-u^{n-1}_{ijkl})/h_3,(u^{0,0,r,1}_{ijkl}-u^{0,0,r,-1}_{ijkl})/h_4\Big),\end{aligned}$$

$$\nabla^{000s}u^{n-1}_{ijkl} = \frac{1}{m(e^{000s}_{ijkl})}\int\limits_{e^{000s}_{ijkl}} \nabla u^{n-1}dx \;\; \approx \Bigg( (u^{1,0,0,s}_{ijkl} - u^{-1,0,0,s}_{ijkl})/h_1, (u^{0,1,0,s}_{ijkl} - u^{0,-1,0,s}_{ijkl})/h_2,$$
$$(u^{0,0,1,s}_{ijkl} - u^{0,0,-1,s}_{ijkl})/h_3, s(u^{n-1}_{i,j,k,l+s} - u^{n-1}_{ijk,l})/h_4 \Bigg).$$

If the same approach for computation of gradients of image intensities is used, then the following approximation of $G^{pqrs}_{ijkl}$ in (14) is obtained

$$G^{pqrs}_{ijkl} = g\Big(\delta|\nabla^{pqrs}I^0_{\sigma;ijkl}| + \vartheta|\nabla^{pqrs}I^{TH}_{\sigma;ijkl}|\Big), \tag{15}$$

where $I^0_\sigma = G_\sigma * I^0$, $I^0_{\sigma;ijkl}$ is the value of $I^0_\sigma$ in doxel $V_{ijkl}$, and $I^{TH}_\sigma = G_\sigma * I^{TH}$, $I^{TH}_{\sigma;ijkl}$ is the value of $I^{TH}_\sigma$ in doxel $V_{ijkl}$. Finally, incorporating $\varepsilon-$regularization, we obtain the following remaining terms in (14)

$$A^{pqrs;n-1}_{\varepsilon,ijkl} = \sqrt{\varepsilon^2 + |\nabla^{pqrs}u^{n-1}_{ijkl}|^2},\;\; \bar{A}^{n-1}_{\varepsilon,ijkl} = \sqrt{\varepsilon^2 + \frac{1}{8}\sum_{N_{ijkl}} |\nabla^{pqrs}u^{n-1}_{ijkl}|^2}, \tag{16}$$

Equations (14) accompanied by the zero Dirichlet boundary condition, represent a linear system of equations which can be solved efficiently, e.g., using the Successive Overrelaxation (SOR) method.

**Remark 1** (**Local rescaling**). *For each segmentation time step $n$, and $l = 1, \cdots, N_4$, let $C_l = \{c^l_m\}_{m=1}^{N_l}$ denote the set of centers, where $c^l_m$ and $N_l$ represent the $m^{th}$ center and the total number of centers, respectively for each $l$. Let $B^l_m(c^l_m, r)$ be a ball with radius $r$ and center $c^l_m$ ($c^l_m$ is a given point inside the object to be segmented), $\mu^l_m = \min\limits_{B^l_m(c^l_m,r)} u^n_{ijkl}$ and $\xi^l_m = \max\limits_{B^l_m(c^l_m,r)} u^n_{ijkl}$. Then the locally rescaled version of $u^n_{ijkl}$ given by* (14) *within the ball $B^l_m(c^l_m, r)$ is obtained by the following relation*

$$u^{resc;n}_{ijkl} = \frac{1}{\xi^l_m - \mu^l_m}(u^n_{ijkl} - \mu^l_m). \tag{17}$$

Consequently, we have that for each time step $n$, rescaled version $u^{resc;n}_{ijkl} \in [0,1]$ and it is used instead of $u^{n-1}_{ijkl}$ in (14) in the next segmentation time step. So in each segmentation step we solve the following system of equations representing the final formulation of our method:

$$\begin{aligned} \Bigg(1 + \frac{\tau}{m(V_{ijkl})}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} G^{pqrs}_{ijkl}\frac{m(e^{pqrs}_{ijkl})}{A^{pqrs;n-1}_{\varepsilon,ijkl}m(\sigma^{pqrs}_{ijkl})}\Bigg)u^n_{ijkl} \;\; &- \\ \frac{\tau}{m(V_{ijkl})}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} G^{pqrs}_{ijkl}\frac{m(e^{pqrs}_{ijkl})}{A^{pqrs;n-1}_{\varepsilon,ijkl}m(\sigma^{pqrs}_{ijkl})}u^n_{i+p,j+q,k+r,l+s} \;\; &= \;\; u^{resc;n-1}_{ijkl}, \end{aligned} \tag{18}$$

$i = 1, \cdots, N_1$, $j = 1, \cdots, N_2$, $k = 1, \cdots, N_3$, and $l = 1, \cdots, N_4$. The coefficients of (18) are computed using $u^{resc;n-1}_{ijkl}$ instead of $u^{n-1}_{ijkl}$.

Finally, we note that if $h = h_1 = h_2 = h_3 = h_4$, then $m(V_{ijkl}) = h^4$, $m(e^{pqrs}_{ijkl}) = h^3$, and $m(\sigma^{pqrs}_{ijkl}) = h$, and the equation (18) simplifies to

$$\begin{aligned} \Bigg(1 + \frac{\tau}{h^2}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} \frac{G^{pqrs}_{ijkl}}{A^{pqrs;n-1}_{\varepsilon,ijkl}}\Bigg)u^n_{ijkl} \;\; &- \\ \frac{\tau}{h^2}\bar{A}^{n-1}_{\varepsilon,ijkl}\sum_{N_{ijkl}} \frac{G^{pqrs}_{ijkl}}{A^{pqrs;n-1}_{\varepsilon,ijkl}}u^n_{i+p,j+q,k+r,l+s} \;\; &= \;\; u^{resc;n-1}_{ijkl}. \end{aligned} \tag{19}$$

Since in our implementation we used common $h$, in the sequel, we deal with the properties of the system (19). All

derived properties are simply adopted also to the system (18).

### 3.3 Stability of the numerical scheme

In this section, we present a short proof that the linear system given by equation (19) has a unique solution and that numerical scheme employed is unconditionally stable. The method or technique of proof presented in [27, 14] has been adopted.

**Definition 3.1.** *The semi-implicit finite volume scheme given by equation* (19) *for solving equation* (6) *is unconditionally stable if for each* $\epsilon > 0$, $\tau > 0$ *and* $n \in \{1, \cdots, N\}$, *the following inequality (the discrete minimum–maximum principle) holds*

$$\min_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{resc;n-1} \leq \min_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{n} \leq \max_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{n} \leq \max_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{resc;n-1}. \tag{20}$$

**Theorem 3.2.** *The linear scheme given by equation* (19) *has a unique solution* $u_{ijkl}^{n}$ *and is unconditionally stable for each* $\epsilon > 0$, $\tau > 0$ *and* $n \in \{1, \cdots, N\}$.

*Proof.* The equation (19) together with Dirichlet boundary condition is a system of linear equations with square matrix whose off diagonal elements are given by

$$-\frac{\tau}{h^2} \bar{A}_{\varepsilon,ijkl}^{n-1} \frac{G_{ijkl}^{pqrs}}{A_{\varepsilon,ijkl}^{pqrs;n-1}}, \ (p,q,r,s) \in N_{ijkl}. \tag{21}$$

Also, the diagonal elements given by

$$1 + \frac{\tau}{h^2} \bar{A}_{\varepsilon,ijkl}^{n-1} \sum_{N_{ijkl}} \frac{G_{ijkl}^{pqrs}}{A_{\varepsilon,ijkl}^{pqrs;n-1}} \tag{22}$$

are non-negative and dominate the sum of absolute value of the nondiagonal elements in each row. Hence, the matrix of the linear system (19) is a strictly diagonally dominant M-matrix. Consequently, the existence of a unique solution of (19) is guaranteed [3, 37]. Next, we show that the scheme is unconditionally stable. For this, it is enough to show that equation (19) satisfies (20). Clearly, equation (19) is same as

$$u_{ijkl}^{n} + \frac{\tau}{h^2} \bar{A}_{\varepsilon,ijkl}^{n-1} \sum_{N_{ijkl}} G_{ijkl}^{pqrs} \frac{u_{ijkl}^{n} - u_{i+p,j+q,k+r,l+s}^{n}}{A_{\varepsilon,ijkl}^{pqrs;n-1}} = u_{ijkl}^{resc;n-1}. \tag{23}$$

Let

$$\max_{V_{abcd} \in \mathcal{T}_h} u_{abcd}^{n} = u_{ijkl}^{n}.$$

Then from (23) we have that

$$\frac{\tau}{h^2} \bar{A}_{\varepsilon,ijkl}^{n-1} \sum_{N_{ijkl}} G_{ijkl}^{pqrs} \frac{u_{ijkl}^{n} - u_{i+p,j+q,k+r,l+s}^{n}}{A_{\varepsilon,ijkl}^{pqrs;n-1}} \geq 0.$$

Hence,

$$\max_{V_{abcd} \in \mathcal{T}_h} u_{abcd}^{n} \leq u_{ijkl}^{resc;n-1} \leq \max_{V_{abcd} \in \mathcal{T}_h} u_{abcd}^{resc;n-1}. \tag{24}$$

Using similar arguments, we have that

$$\min_{V_{abcd} \in \mathcal{T}_h} u_{abcd}^{resc;n-1} \geq u_{ijkl}^{resc;n-1} \geq \min_{V_{abcd} \in \mathcal{T}_h} u_{abcd}^{n}. \tag{25}$$

Equations (24) and (25) implies that

$$\min_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{resc;n-1} \leq \min_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{n} \leq \max_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{n} \leq \max_{V_{ijkl} \in \mathcal{T}_h} u_{ijkl}^{resc;n-1}, \tag{26}$$

which yield equation (20). Hence, the numerical scheme given by equation (19) is unconditionally stable. □

### 3.4 Experimental order of convergence

Assuming that the error of a scheme in some given norm is proportional to some power, $\alpha$, of the grid size $h$. Then we have that $Err(h) = Ch^{\alpha}$ where $C$ is a contant of proportionality. If half of the grid size is considered, i.e., $h := \frac{h}{2}$ then
$Err(\frac{h}{2}) = C(\frac{h}{2})^{\alpha}$. Consequently,

$$\alpha = \log_2 \Big( \frac{Err(h)}{Err(\frac{h}{2})} \Big). \tag{27}$$

The $\alpha$ is called the *experimental order of convergence* ($EOC$) and is determined by comparing numerical solutions and exact solutions on subsequently refined grids [4].
We now test our method using the exact solution (see also [4])

$$u(x_1, x_2, x_3, x_4, t) = \frac{x_1^2 + x_2^2 + x_3^2 + x_4^2 - 1}{6} + t. \tag{28}$$

of the level set equation

$$u_t = |\nabla u| \nabla \cdot \frac{\nabla u}{|\nabla u|} \tag{29}$$

and consider Dirichlet boundary conditions given by this exact solution.
This problem is solved in the spatial domain $\Omega = [-1.25,\ 1.25]^4$ and in the time interval $T = 0.0625$. We have taken subsequent grid refinement $N_1 = N_2 = N_3 = N_4 = 10,\ 20,\ 40,\ 80,\ 160$ and considered number of discrete time steps $1,\ 4,\ 16,\ 64,\ 256$ respectively. The time step $\tau$ is chosen proportionally to $h^2$ and we measure errors in $L_2((0,T), L_2(\Omega))-$norm. Table 1 shows errors in $L_2((0,T), L_2(\Omega))-$norm for refined grids and $\epsilon = h^2$.

| $n$ | $h$ | final step | Error ($\epsilon = h^2$) | EOC |
|---|---|---|---|---|
| 10 | 0.25 | 1 | $1.680403e-2$ | |
| 20 | 0.125 | 4 | $4.653133e-3$ | 1.852533 |
| 40 | 0.0625 | 16 | $1.208771e-3$ | 1.944661 |
| 80 | 0.03125 | 64 | $3.029600e-4$ | 1.996342 |
| 160 | 0.015625 | 256 | $8.340820e-5$ | 1.860866 |

Table 1: Errors in $L_2((0,T), L_2(\Omega))-$norm, and EOC comparing numerical and exact solution (28).

We observe that $\alpha \approx 2$ as the grids get more refined, implying that the method converges with an order of almost two. Thus, we can conclude that the scheme is reliable and can be used in practical applications.

### 3.5 Brief overview of computer implementation

The following steps provide an overview of the implementation steps for the new 4D model.

- Read input 3D+time image together with the corresponding centers of cells.
- Using the input centers, generate initial segmentation function (or initial condition).
- Locally rescale the initial segmentation function to interval $[0, 1]$.
- Using the input centers, perform local thresholding of 4D image.
    1. Compute the coefficients $G_{ijkl}^{pqrs}$, $A_{\varepsilon,ijkl}^{pqrs;n-1}$, and $\bar{A}_{\varepsilon,ijkl}^{n-1}$ either in doxels or on doxel sides using the locally rescaled segmentation function.
    2. Solve the linear system given by (19).

3. Locally rescale the computed segmentation function to interval $[0, 1]$.

- Repeat steps 1, 2, and 3 until the total number of segmentation steps is reached.
- Output the result of segmentation

For complete serial and parallel implementation in C programming language, see https://github.com/88MARK08/4D-image-segmentation-algorithm.

### 3.6 Parallel implementation using OpenMP

OpenMP is a multi-threading implementation. In C/C++, omp.h header file includes all OpenMP functions. In our implementation of the new 4D model, *#pragma omp parallel private*$\{\cdots\}$ is used to instruct the OpenMP system to divide tasks among the working threads. In the for loops, the first loop is the loop for the real-time length $\theta$. In other words, we split a series of 3D volumes among working threads. Furthermore, *#pragma omp parallel for private*$\{\cdots\}$ *reduction(operator:variable)* is used to accomplish reduction operations. For instance, *reduction (operator: variable)* is used to specify that the operation given by the "*operator*" should be performed on the values of the "*variable*" from all threads at the end of the parallel construct. Finally, to measure CPU time in parallel implementation, *omp_ get_ wtime()* function is used.

In table 2, we present a comparison of CPU times with OpenMP parallel implementation and serial implementation. In this experiment, a PC with 8192 MB RAM and processor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz (8 CPUs), $\sim$ 2.8 GHz was used.

| # threads | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| time (secs) | 225.8700 | 119.0000 | 71.9270 | 42.3050 |
| speed-up | 0 | 1.898 | 3.140 | 5.339 |

Table 2: Computing times and speed-up of OpenMP parallel program for computing the EOC in Section 3.4, with $n = 40$.

### 3.7 Parallel implementation using MPI

The two main goals of parallel program implementation are handling huge amounts of data that cannot be placed in the memory of a single serial computer and the shortest possible program execution time. Assuming that in terms of execution time, a fraction $P$ of a program can be parallelized. In an ideal case, while executing a parallel program on $n_p$ processors, the execution time will be $1-P+\frac{P}{n_p}$. Furthermore, the theoretical speed-up according to the Amdahl's law [1] is given by $\frac{1}{(1-P)+\frac{P}{n_p}}$. Consequently, if only 90% of the program can be parallelized, for example, then with infinitely many processors the maximal speed-up (estimated from Amdahl's law by $\frac{1}{1-P}$) cannot exceed 10. To minimize the time spent on communication, it is necessary to require that the data transmitted (e.g., multidimensional arrays) be contiguous in memory. This is to ensure that data are exchanged directly among processes in one message using only one call of MPI send and receive subroutines. In this section, discrete 4D image is represented by a four-dimensional array indexed by $i, j, k, l$, with $i = 1, \cdots, N_1$, $j = 1, \cdots, N_2$, $k = 1, \cdots, N_3$, $l = 1, \cdots, N_4$. The discrete computational domain including the boundary conditions is given by $i = 0, \cdots, N_1 + 1$, $j = 0, \cdots, N_2 + 1$, $k = 0, \cdots, N_3 + 1$, $l = 0, \cdots, N_4 + 1$. The boundary positions - in the computational domain - with $i = 0$, $i = N_1 + 1$, $j = 0$, $j = N_2 + 1$, $k = 0$, $k = N_3 + 1$, $l = 0$, $l = N_4 + 1$ are reserved for Dirichlet boundary conditions and all the inner doxel positions correspond to the original 4D image. Let $n_p$ be the number of processes, then to distribute the 4D data, define $n_4 = \lceil \frac{N_4}{n_p} \rceil$, $n_4^{last} = N_4 - (n_p - 1)n_4$, $n_3 = N_3$, $n_2 = N_2$, $n_1 = N_1$. Hence, the processes with rank from 0 to $n_p - 2$ deal with part of the discrete 4D image which is the series of 3D volumes given by the array with indices $i = 0, \cdots, n_1 + 1$, $j = 0, \cdots, n_2 + 1$, $k = 0, \cdots, n_3 + 1$, indexed locally by $l$ in the range $l = 0, \cdots, n_4 + 1$. Additionally, on the last process with rank $n_p - 1$, the index $l$ of the last 3D volume is $n_4^{last} + 1$ instead of $n_4 + 1$. The merging of all 3D volumes for $l = 1, \cdots, n_4$ ($n_4^{last}$ on the last process) from all processes gives the non-distributed complete 4D image. For the purpose of iterative solution of the linear system and computation of its coefficients, the data overlap is needed. The overlap with the necessity of information exchange between neighbouring processes is

given by the slices $n_4$, $n_4+1$ and slices 0, 1 of the subsequent processes. It is good to mention at this point that in our computer implementation, $iMax = n_1 + 2$, $jMax = n_2 + 2$, $kMax = n_3 + 2$, $lMax = n_4 + 2$, $N = proc\_lMax = n_4$. Furthermore, the following relationship is used (see Listing 1) to transform 4D to 1D array:

$$T_{1D}(i, j, k, l) = l * iMax * jMax * kMax + k * iMax * jMax + j * iMax + i.$$

```
int ijkl(int i, int j, int k, int l)
{
   return l * iMax * jMax * kMax + k * iMax * jMax + j * iMax + i;
   // or equivalently return ((l * kMax + k) * jMax + j) * iMax + i
}
```

Listing 1: Transformation of 4D array to 1D array

In each segmentation time step, the solution values are updated as the linear system is solved iteratively. In each iteration, four of these neighbors should be known already. Consequently, each consecutive process must wait until its preceding process is completed to get the unknown neighbors' values updated in a parallel run. The RED-BLACK SOR method (see, e.g., [2, 26]) is employed to do away with this dependency. In the RED-BLACK SOR, all doxels in the computational domain are split into RED elements, given by the condition that the sum of its indices is an even number, and BLACK elements, given by the condition that the sum of its indices is an odd number. So, the eight neighbors of RED elements are BLACK elements and the value of RED elements depends only on those of the BLACK elements, and vice versa [26]. As a result of this method, one SOR iteration is split into two steps. In the first step, RED elements are updated and BLACK elements are updated in the second step. This splitting is perfectly parallelizable [26].

After computation of one RED-BLACK SOR iteration for the RED elements on every parallel process, RED updated values have to be exchanged in the overlapping regions, and then one iteration for BLACK elements can be computed. The data exchange is implemented using non-blocking MPI_Isend and MPI_Irecv subroutines. For the residual's computation, partial information from all the processes are collected and send to all processes to check the stopping criterion by every process.

We note that all parallel computations were performed on a Linux cluster comprising six computational servers (nodes) and 192 processors. Each computational node has 252 GB of memory and 32 processors; thus, the cluster has 1512 GB of memory available for computations. Additionally, we used processors belonging to one of the computational nodes for the experiment involving computing the EOC in Section 3.4, whose results are shown in Table 3. Furthermore, we used the six servers to ensure sufficient memory resources for the third and fourth experiments in Chapter 5, which involves solving a linear system with 3 366 466 110 unknowns.

Table 3 shows a comparison of CPU times with MPI parallel implementation and serial implementation. In this experiment, processors in one of the six servers in the Linux cluster were utilized.

| # processors | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| time (secs) | 41681.08 | 20976.89 | 10675.87 | 5487.309 | 3077.166 |
| speed-up | 0 | 1.987 | 3.904 | 7.596 | 13.545 |

Table 3: Computing times and speed-up of MPI parallel program running on 1 to 16 processors for computing the EOC in Section 3.4, with $n = 80$.

Table 3 shows a linear speed-up with 2, 4, and 8 processors. However, with 16 processors, the speed-up dropped from approximately 16 (expected) to 13.545. This drop may be attributed to the amount of time spent on communication between processes. For instance, one, three, seven, and 15 2-way communications are involved with two, four, eight,

and 16 processors, respectively. Thus, we may conclude that the communication time impacts the speed-up of the MPI parallel program.

# 4 Application of the new model to 3D Cell Image Segmentation

In this section, we present a special case of the new 4D model. That is the generalized SUBSURF method (6) in 3D. The 3D model has been applied to 3D cell image segmentation (see also, [39, 40]).

## 4.1 3D Numerical experiments with zebrafish data

Numerical experiments were performed on a biological image of a developing pectoral fin in a zebrafish embryo to demonstrate the performance of our mathematical model (6) in 3D. In all experiments, isosurface 0.5 was displayed. 3D microscopy images of cell nuclei and membranes in the pectoral fin were provided by Hanh Nguyen from the group of Nadine Peyriéras (CNRS BioEmergences, France).

In the next two subsections, we present the results of numerical experiments involving the application of (6) to real data representing 3D microscopy images of cell nuclei and membranes.

### 4.1.1 Experiments with cell nuclei images

In this subsection, we present segmentation results after the application of (6) with different choices of $\delta$ and $\vartheta$. The cell nucleus of interest and the corresponding result after segmentation (which is depicted in black) are located approximately at the center of each picture as shown in Figures 8−10.

For the first numerical experiment, $\delta = 1.0$ and $\vartheta = 0.0$ were used. This choice of parameters reduced equation (6) to the classical SUBSURF model [35]. Some good results of the segmentation were obtained (see e.g., Figure 3 in Section 2). However, several results obtained using these parameters were not optimal (see for example Figure 5 in Chapter 3).

In the second experiment, $\delta = 0.0$ and $\vartheta = 1.0$ were used. By visual inspection, one can see from Figure 8 that the results obtained are accurate. Nevertheless, the problem with this choice of parameters is that the reconstructed surface is not smooth. This can be seen in the last column of Figure 8. Moreover, it can be seen from Figure 9 that the result obtained after applying model (6) with $\delta = 0.5$ and $\vartheta = 0.5$ is very similar to the one shown in Figure 8 and the isosurface representing the reconstructed surface is smooth.

Finally, several examples of 3D image segmentation involving the use of model (6) with $\delta = 0.5$ and $\vartheta = 0.5$ were performed. Results of these numerical experiments are shown in Figure 10. In this Figure 10, the first row shows the 3D microscopy images of cell nuclei to be segmented and the second row shows their corresponding results after segmentation.

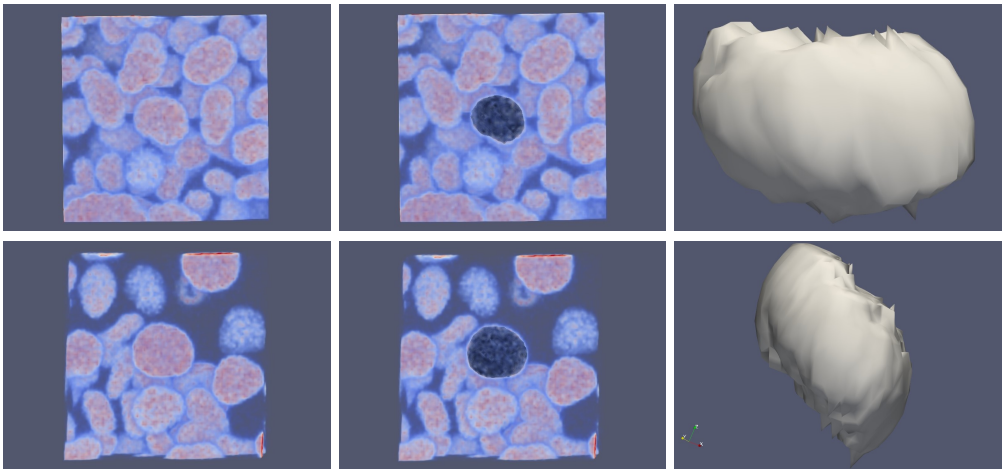

Figure 8: First row shows the 3D microscopy image of a cell nucleus, its reconstruction using $\delta = 0.0$ and $\vartheta = 1.0$ in (6) and 0.5 isosurface. Second row shows the 3D image of another cell nucleus, its reconstruction using $\delta = 0.0$ and $\vartheta = 1.0$ in (6) and 0.5 isosurface.

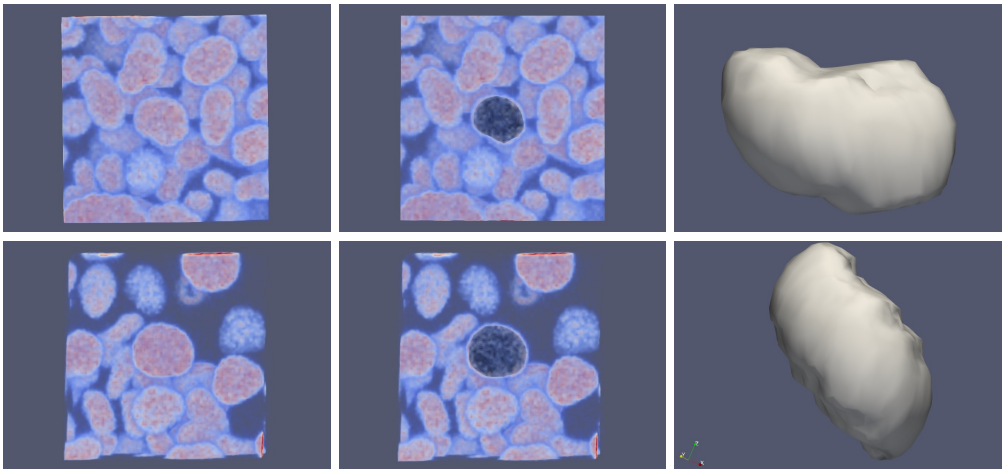

Figure 9: First row shows the 3D microscopy image of a cell nucleus, its reconstruction using $\delta = 0.5$ and $\vartheta = 0.5$ in (6) and 0.5 isosurface. Second row shows the 3D image of another cell nucleus, its reconstruction using $\delta = 0.5$ and $\vartheta = 0.5$ in (6) and 0.5 isosurface.

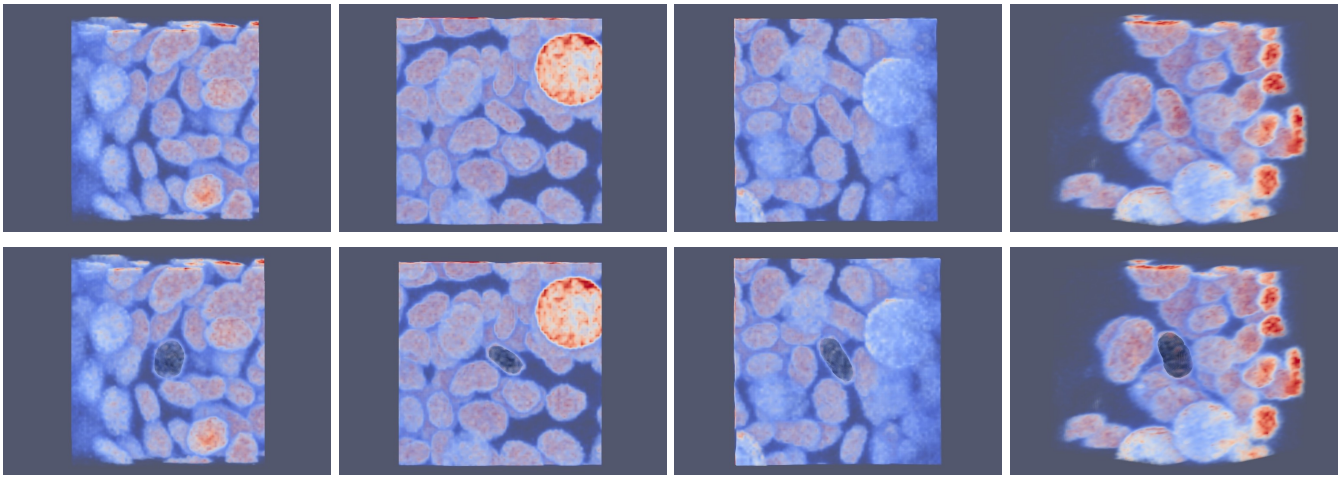

Figure 10: In this figure, first row shows the 3D microscopy images of four different cell nuclei; second row shows segmentation result using thresholded image intensity information and original image intensity information in (6). That is, the result after application of (6) with $\delta = 0.5$ and $\vartheta = 0.5$.

In these numerical experiments, calculations were executed on a subdomain grid with $72 \times 72 \times 32$ voxels, and computational method parameters were set to $h = 0.01$, $\tau = 0.1$, $\varepsilon = 1$, $\lambda = 0.7$, $r = 12$, $K = 5$ and $N = 100$.

#### 4.1.2 Experiments with cell membrane images

For the segmentation of membrane images, the classical SUBSURF segmentation approach seems not to give the optimal result. Hence, we use the idea of local thresholding of image intensity values within a ball of appropriate radius around the approximate center of a membrane image to improve the quality of the final segmentation result. Additionally, like the model (6) in 3D, the idea is to presmooth the original membrane image, apply the local thresholding to the presmoothed image, and combine the information obtained from the thresholding with the information from the original membrane image intensities to get a final membrane segmentation model. Thus, in the formulation of model (6) in Chapter 3 for membrane image segmentation,

$$\alpha_m^\theta = \min_{y \in B_m^\theta(c_m^\theta, r)} G_\sigma * I^0(y, \theta), \ \beta_m^\theta = \max_{y \in B_m^\theta(c_m^\theta, r)} G_\sigma * I^0(y, \theta).$$

In the first numerical experiment, $\delta = 1.0$ and $\vartheta = 0.0$ were used. This choice of parameters reduced equation (6) to the classical SUBSURF model [35]. Figure 4 in Chapter 2 and Figure 6 in Chapter 3 show the results obtained using these parameters. Figure 4 shows good segmentation result; however, several results obtained using these parameters were not optimal (Figure 6 in Chapter 3). Furthermore, for other numerical experiments, $\delta = 0.8$ and $\vartheta = 0.2$, $\delta = 0.7$

and $\vartheta = 0.3$, and $\delta = 0.6$ and $\vartheta = 0.4$ were used. The results obtained using these parameters are shown in Figure 11. With these nonzero choices of $\vartheta$, it can be seen by visual comparison that the results are correct.

We note that in Figure 6 of Chapter 3, the first row shows the visualization of the original 3D membrane image intensity, and the second row shows, in black color, their corresponding results after segmentation. The partial reconstruction of the membrane images shown in the row column of Figure 6 may be attributed to the fact that some internal structures or edges in the image did not allow the segmentation function to get to the actual boundary of the cell. However, with the local thresholding, the segmentation function is able to grow to the boundary of the cell, giving rise to the result, which is shown in the second, third, and fourth columns of Figure 11.

Finally, we note that in these numerical experiments, computations were performed on a grid with $50 \times 50 \times 40$ voxels, and computational method parameters were set to $h = 0.01$, $\tau = 0.01$, $\varepsilon = 1$, $\lambda = 0.95$, $r = 26$, $K = 0.8$ and $N = 100$.

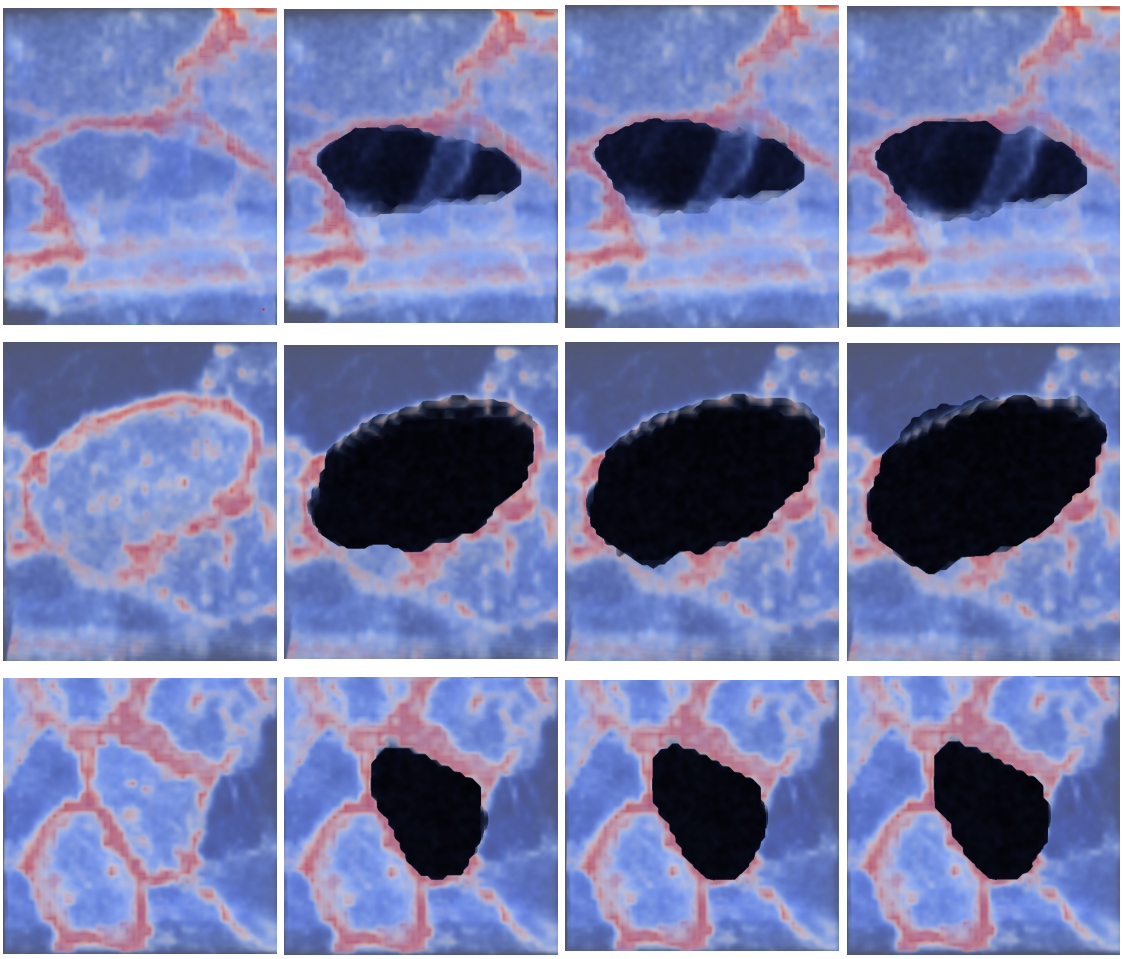

Figure 11: First column of this figure shows the 3D microscopy images of cell membrane to be segmented. The second, third, and fourth columns show the result after the application of (6) with $\delta = 0.8$ and $\vartheta = 0.2$, $\delta = 0.7$ and $\vartheta = 0.3$, and $\delta = 0.6$ and $\vartheta = 0.4$, respectively.

## 4.2 3D Numerical experiments with mouse data

In this section, 3D experiments were performed on biological dataset of the embryonic development of mouse to demonstrate the performance of our mathematical model (6). In all the experiments, isosurface 0.5 was displayed. 3D microscopy images of cell nuclei in the mouse embryo were provided by Antonia Weberling from the group of Magdalena Zernicka-Goetz (Department of Physiology, Development, and Neuroscience, University of Cambridge, UK).

Figure 12 shows the original 3D image intensity of mouse embryo cell nuclei. The image intensities in the five small visible cubes represent the cell nuclei of interest while the visible cubes represent the computational subdomain. In Figure 7 (see Chapter 3) and Figure 12, the image intensities in the small visible cubes coloured in black represent

results after segmentation using different choices of $\delta$ and $\vartheta$ in equation (6). For the first numerical experiment, $\delta = 1.0$ and $\vartheta = 0.0$ were used. Results obtained using these parameters were not optimal, see Figure 7 in Chapter 3. In the second experiment, $\delta = 0.5$ and $\vartheta = 0.5$ were used. The results obtained, as can be seen from Figure 12, were satisfactory.

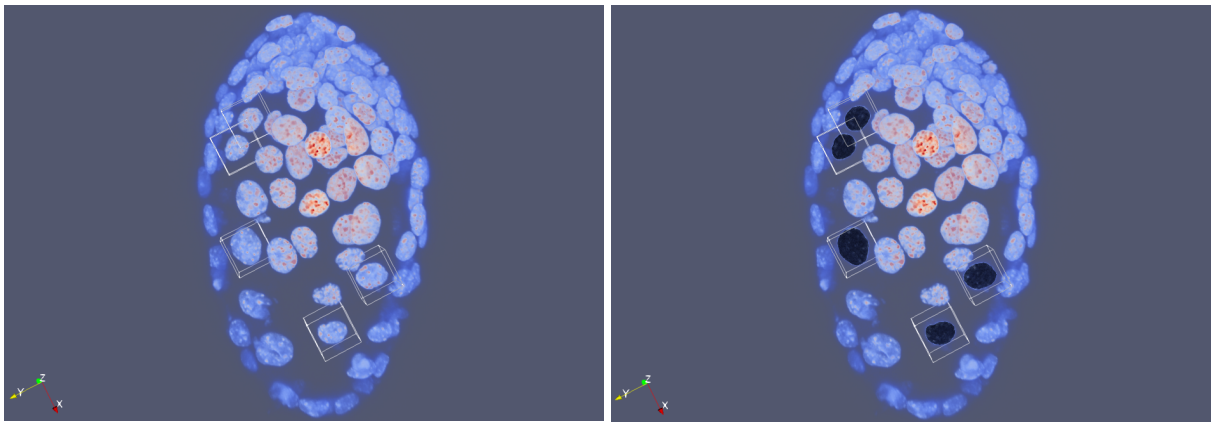

Figure 12: This figure shows the 3D microscopy image of mouse embryo cell nuclei.

In these numerical experiments, calculations were executed on a subdomain grid with $40 \times 40 \times 34$ voxels, and computational method parameters were set to $h = 0.01$, $\tau = 0.1$, $\varepsilon = 1$, $\lambda = 0.85$, $r = 28$, $K = 1$ and $N = 100$. Finally, the results presented in Sections 4.1 and 4.2 of this chapter show that mathematical model (6) is a useful and successful generalization of the classical SUBSURF model for 3D cell image segmentation.

# 5 Application to 4D image segmentation

In this chapter, the new 4D method is tested on artificially generated 3D+time videos and applied to real data representing 3D+time microscopy images of cell nuclei within the zebrafish pectoral fin and hind-brain.

In the first experiment (see https://doi.org/10.5281/zenodo.5513089 for the videos), 3D+time video of one sphere was artificially generated. The goal of this simple experiment is to show that the 4D method (6) can approximate a missing shape in a 3D+time video. This is because SUBSURF models can complete a missing part of an object. In the 4D case, the method is expected to complete a missing volume. Thus, from a 3D+time video of 20 frames, we removed time frames 5, 10, and 15 and tried reconstructing the entire video using our 4D model. In Figure 13, the first column shows the 3D frame of the video at the time step 10, the second and third columns show their corresponding reconstruction using the 4D method; the third column shows the result when the sphere at volume/frame 10 is removed. We note that in the second column, all frames were considered in the segmentation. The results of the segmentation shown in columns two and three are colored in blue. Furthermore, the second column of Figure 13 shows that the segmentation results of this moving sphere are good. However, the third column of this figure shows that the segmentation result of the sphere at the missing volume/frame is an ellipsoidal shape instead of a spherical shape. The approximation of a sphere with an ellipsoid is a good result that can be very useful during cell tracking. Thus, we can conclude from this experiment that our 4D method can approximate a missing shape in 3D+time.

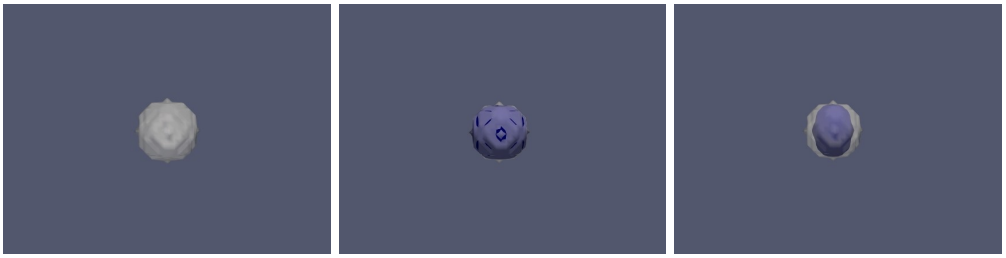

Figure 13: First column of this figure shows the 3D frame of 4D image of a sphere at time step 10, the second column shows the corresponding reconstruction using (6), and the third column shows the result when the sphere at volume/frame 10 is removed.

In the second experiment (see https://doi.org/10.5281/zenodo.5513089 for the videos), 3D+time videos of spheres were artificially generated. There are four spheres in time steps $1-3$, seven spheres in time steps $4-17$, and five spheres in time steps $18-20$. In Figure 14, the first row shows the 3D frames of the video at the time steps 1 (first column), 10 (second column), and 20 (third column), and the second row shows their corresponding reconstruction using the new method. The results of the segmentation are colored in blue. Furthermore, the segmentation results of the 3D+time videos of these spheres show a very good performance of the 4D method (6) on this artificial dataset.

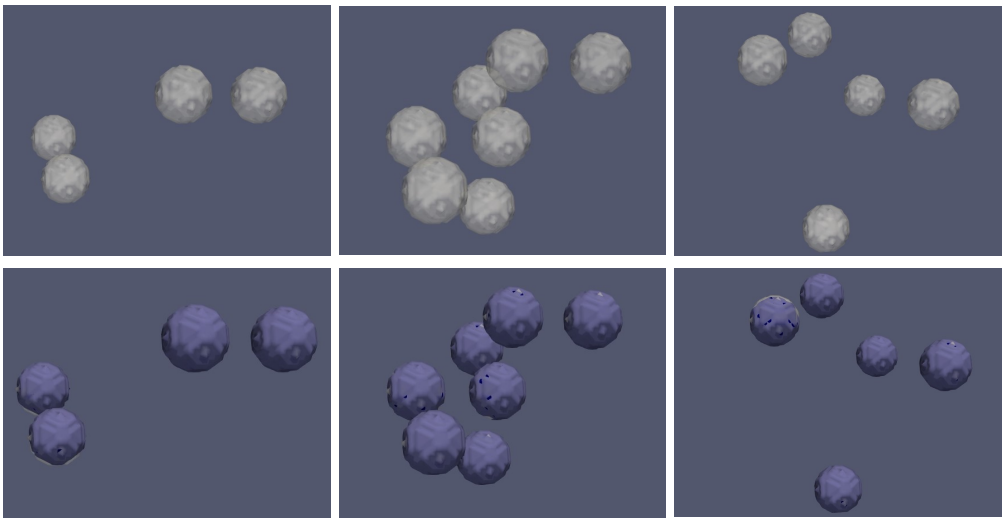

Figure 14: First row of this figure shows the 3D frames of 4D image of spheres at time steps 1 (first column), 10 (second column), and 20 (third column) and the second row shows their corresponding reconstruction using (6).

In the following two experiments, we present the results of experiments with data from zebrafish hindbrain. 3D microscopy images of cell nuclei in the hindbrain of developing zebrafish embryos were provided by Mageshi Kamaraj from the group of Nadine Peyriéras (CNRS BioEmergences, France).

In the subsequent two experiments, seven arbitrary cell nuclei centers in zebrafish hindbrain were selected, and 23 time frames were considered. For each time frame, we considered a small $30 \times 30 \times 30$ computational domain around each of these seven nuclei centers. The motivation for this construction is to reduce the memory requirement of our serial implementation and ensure easy visualization. Furthermore, image intensities from the original dataset are copied to these small computational domains around nuclei centers in each time frame. Hence, when the 3D volumes containing these seven small cubes are put together, we obtain the 3D+time image shown in Figure 15.

In the third experiment, using the 3D method presented in Chapter 4, we performed 3D segmentation of these seven cell nuclei within the small $30 \times 30 \times 30$ computational domain in each 3D volume containing these seven small cubes. The 3D segmentation results of these cell nuclei in each 3D volume are put together over time and used as an input 3D+time image to the 4D method (first column of Figure 15). This is another artificial experiment that is more closer to experiments with real data. Hence, in generating the 3D+time image for this experiment, we consider shapes closer to real cell shapes than spheres. We note that in the first, second, and third experiments, the following pairs of

parameters ($\delta = 1, \vartheta = 0$), ($\delta = 0, \vartheta = 1$), and ($\delta = 0.5, \vartheta = 0.5$) in model (6) yield the same result. This is because the thresholded and original image intensities are the same.

First image of Figure 15 shows the visualization of seven zebrafish cell nuclei, segmented in 3D and put together to form a 3D+time image. Second image of Figure 15 shows the corresponding segmentation result using the 4D model (6). The segmentation results are colored in blue and show accurate correspondence of the segmented moving cells in 4D with the original 4D image.

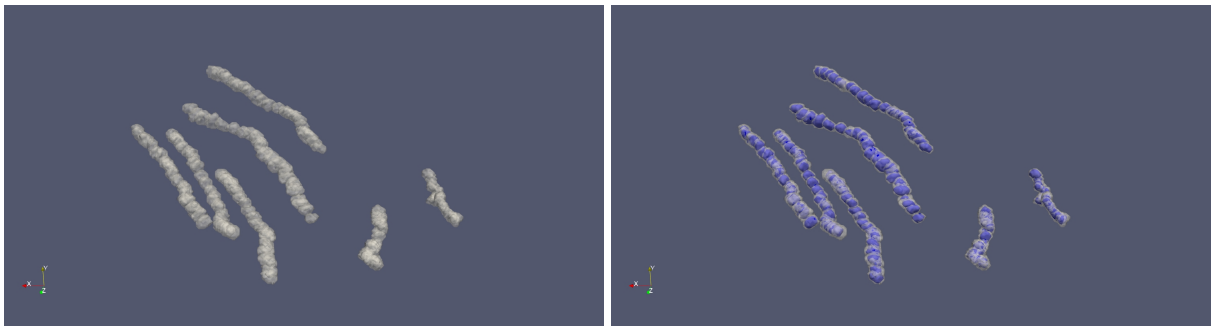

Figure 15: This figure shows the 3D image of seven zebrafish cell nuclei moving in time (first image). Second image shows the segementation result using model 6.

In the fourth experiment, we worked with original image intensity within the small $30 \times 30 \times 30$ computational domain. For each of the seven selected nuclei centers, the intensities around each nucleus are copied to the $30 \times 30 \times 30$ computational domain. Hence, there are seven small computational domains in each time frame or 3D volume, and each domain contains the original image intensity. The 3D+time image processed is obtained by putting these original 3D images (containing seven small 3D volumes) together. Additionally, we restricted computations to a small 3D volume around the nuclei. The images presented in Figure 16 show a visualization of the 3D+time images. First image of Figure 16 shows the 3D image of seven zebrafish cell nuclei that are moving in time and second image shows the corresponding segmentation result using the 4D model (6). The results of the segmentation are colored in black.

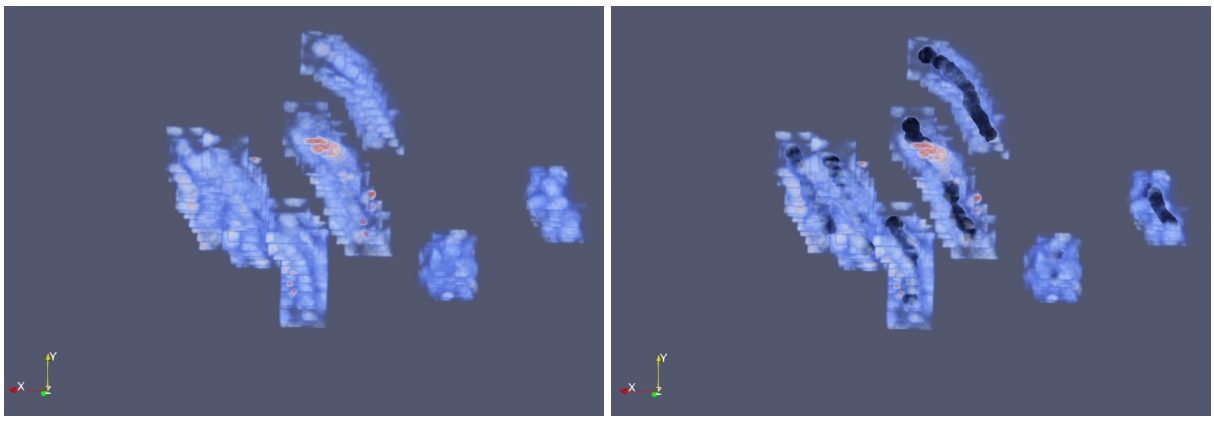

Figure 16: This figure shows the visualization of seven zebrafish cell nuclei moving in time.

Unlike in the previous four experiments where serial implementations were used, we will use the parallel implementation in subsequent experiments; thus, the entire 3D volumes of image intensities will be considered. We remark that the dataset used in the two subsequent experiments comprises several frames of 3D volumes. Each of these volumes has a dimension of $567 \times 577 \times 147$. Therefore, the maximum number of frames of the 3D volumes that can be processed at once in the cluster is 90. The reason for this maximum number is that the cluster has 1512 GB of memory available for computations. Additionally, to process the 90 frames of 3D volumes, with volume dimensions $567 \times 577 \times 147$, more than 1130 GB of memory is needed, and we decided to limit the number of frames in our computations to a maximum of 70. Furthermore, to process these 70 frames of 3D volumes, with volume dimensions $567 \times 577 \times 147$, 1012 GB of memory was needed. This clearly shows that it may not be possible to process these images on a serial machine without parallel implementation utilizing the MPI.

In the fifth experiment (see https://doi.org/10.5281/zenodo.5513118 for the videos of this experiment), we selected seven cell nuclei in 70 time frames within the zebrafish pectoral fin. The selected cell nuclei were clearly visible in all time frames. Thus, easy visualization of segmentation results in all time frames is the motivation for this selection. Additionally, in the two previous experiments of this chapter, we used an arbitrary number "seven;" hence, we have used seven in this experiment too.

In Figure 17, the first row shows the 3D frames of the video at the time steps 1 (first column), 20 (second column), 40 (third column), and 70 (fourth column) and the second row shows the corresponding reconstruction of the selected seven nuclei images using (6). Figure 17 (see also, https://doi.org/10.5281/zenodo.5513118) shows that in each time frame, the seven selected cell nuclei, located nearly at the bottom of each image in the figure, were accurately reconstructed using the 4D method. The segmentation results of these cell nuclei are colored in black at the bottom of each image in the second row of this figure.

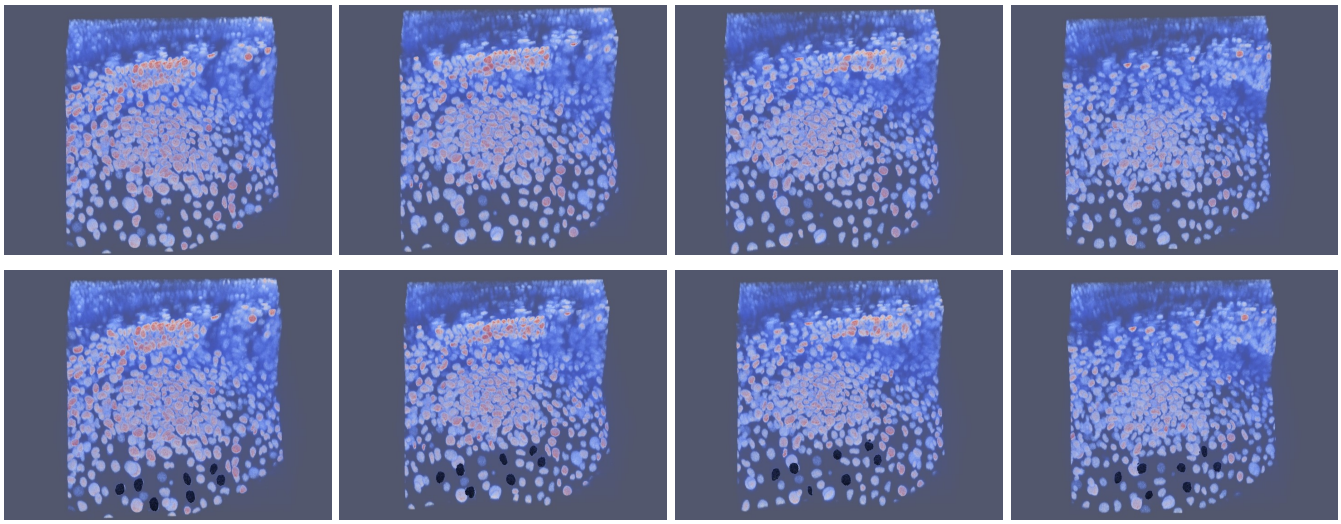

Figure 17: First row of this figure shows the 3D frames of 3D+time microscopy images of cell nuclei within the zebrafish pectoral fin at the time steps 1 (first column), 20 (second column), 40 (third column), and 70 (second column). The second row shows the corresponding reconstruction of the seven nuclei images using (6).

In the last experiment (see https://doi.org/10.5281/zenodo.5513118 for the videos of this experiment), we reconstructed a group of selected cell nuclei in 70 time frames within the zebrafish pectoral fin. This cell population was selected by the biologists who provided the dataset. In Figure 18, the first row shows the 3D frames of the video at the time steps 1 (first column), 20 (second column), 40 (third column), and 70 (fourth column) and the second row shows the corresponding reconstruction of the selected nuclei images using the 4D method (6). The segmentation results of this group of cell nuclei are colored in black at the center of each image in the second row of this figure. Furthermore, the selected group of cells are located inside the fin tissue and are covered by other cells nearer to the fin's surface. Consequently, it is not easy to visualize the segmentation results together with the original image intensity. This is why we selected the seven cell nuclei that are easily visualizable in the previous experiment. Moreover, Figure 21 in Chapter 6 shows segmentation results of this experiment, which were used for cell tracking. Thus, we can deduce from these figures that the segmentation results appear to be correct.

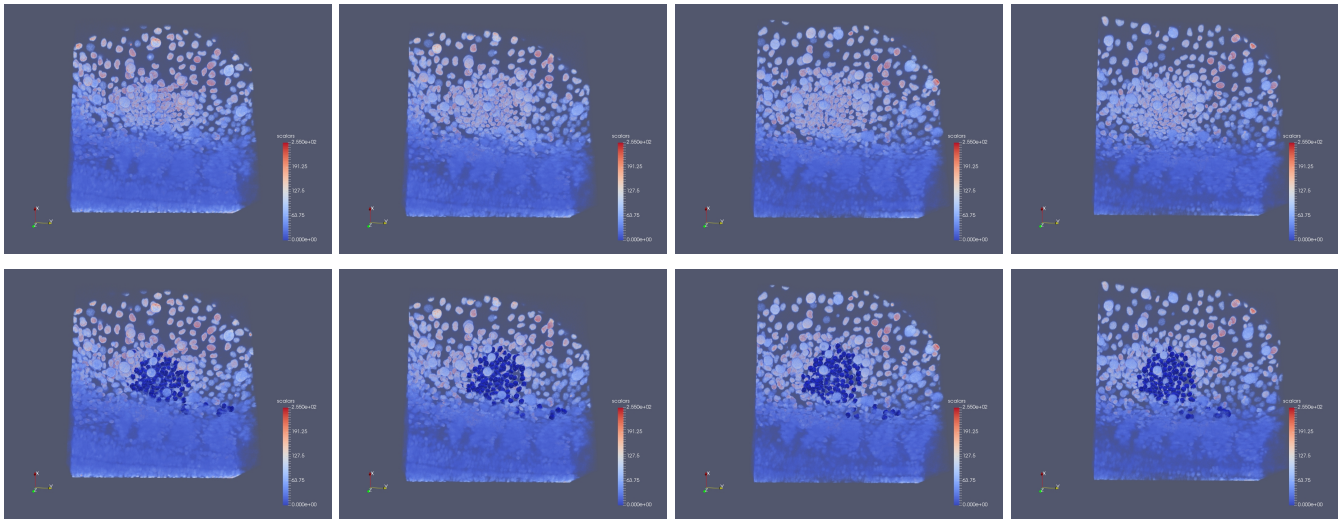

Figure 18: First row of this figure shows the 3D frames of 3D+time microscopy images of cell nuclei within the zebrafish pectoral fin at the time steps 1 (first column), 20 (second column), 40 (third column), and 70 (fourth column) and the second row shows the corresponding reconstruction of the nuclei images using (6).

# 6 Cell tracking based on 4D segmentation

In this chapter, the results of cell tracking based on the 4D segmentation method are presented. The cell tracking procedures for the presented results were part of the work done by Seol Ah Park [31]; for previous works on cell tracking, see [21, 22, 23, 38]. Consequently, we have obtained permission to include this chapter to demonstrate how our 4D method serves as a basis for cell tracking.

## 6.1 Numerical experiments

In the first experiment (see https://doi.org/10.5281/zenodo.5513089 for the complete video), results of segmentation by the 4D method (6) were used as a basis to track the artificially generated spheres shown in Figure 14. In Figure 19, 3D frames of the sphere trajectories (in this case, straight lines) at the time steps 1, 10, and 20 were shown. In this artificial dataset, we may recall that there are four spheres in time steps $1-3$, seven spheres in time steps $4-17$, and five spheres in time steps $18-20$. So, this explains why there are four, seven, and five spheres in this figure. In each of the three pictures, the straight lines show the trajectories of the spheres, and the square at the end of each line or trajectory shows the sphere's present time. The trajectories are correct because, from the construction of the dataset, all spheres move in straight lines. We note that the viewpoint of the images presented in this figure is different from the viewpoint of the images in Figure 14. In presenting the result of this experiment, we choose this viewpoint because it has the best view of all trajectories.

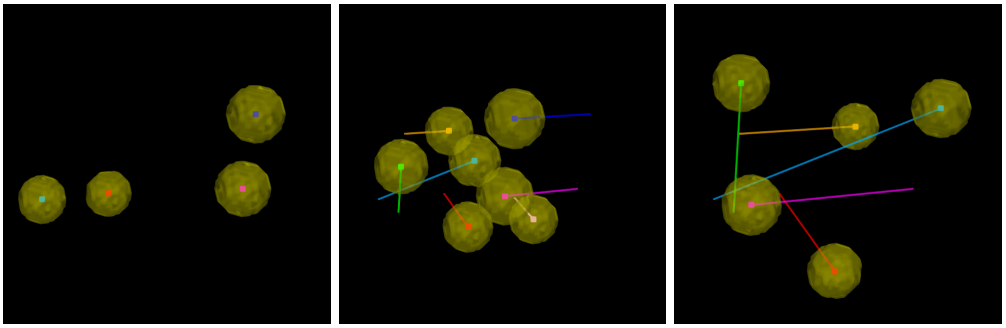

Figure 19: This figure shows 3D frames of the sphere trajectories at time steps 1 (first column), 10 (second column), and 20 (third column).

In the second experiment (see https://doi.org/10.5281/zenodo.5513118 for the complete video), results of segmentation by the 4D method were also used as a basis to track the 3D+time microscopy images of selected seven cell

nuclei within the zebrafish pectoral fin shown in Figure 17. In Figure 20, 3D frames of the trajectories of 3D+time microscopy images of seven cell nuclei within the zebrafish pectoral fin at the time steps 1, 20, 40, and 70 were shown. The trajectories of these tracked cells show the movement of the cells over time. Additionally, from the last frame (second column) in Figure 20, we see that these cells are not moving in straight lines. Instead, they move in an arbitrary random direction.

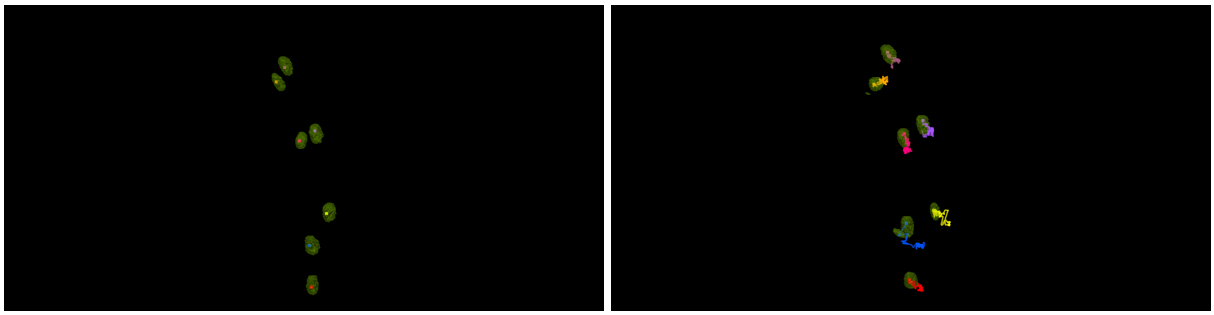

Figure 20: This figure shows 3D frames of the trajectories of 3D+time microscopy images of seven cell nuclei within the zebrafish pectoral fin at time steps 1 (first column) and 70 (second column).

In the third experiment (see https://doi.org/10.5281/zenodo.5513118 for the complete video), results of segmentation by our 4D method were again used as a basis to track the 3D+time microscopy images of a group of selected cell nuclei in 70 time frames within the zebrafish pectoral fin shown in Figure 18. In Figure 21, 3D frames of the trajectories of 3D+time images of five cell nuclei selected from the original population within the zebrafish pectoral fin at the time steps 1, 20, 40, and 70 were presented. The five selected cell trajectories are the ones that showed significant cell movement in the selected population. Additionally, the trajectories of these five selected cells are colored yellow, blue, red, cyan, and pink. Furthermore, in the second image of Figure 21, the initial positions of the nuclei were colored gray, while the current positions are colored green.

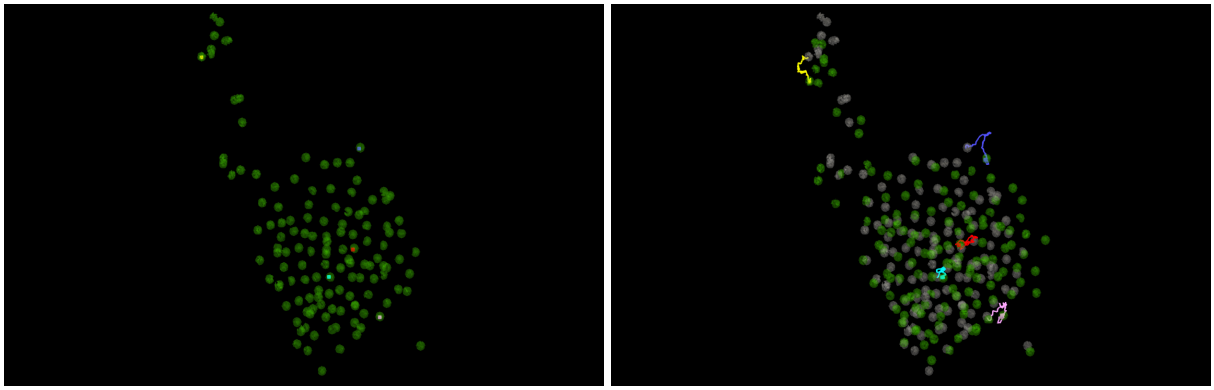

Figure 21: This figure shows 3D frames of the trajectories of 3D+time images of five cell nuclei within the zebrafish pectoral fin at time steps 1 (first row) and 70 (second row).

Finally, from the results presented in Figures 19, 20, and 21, we can conclude that our 4D method (6) serves as basis for cell tracking.

# References


[1] G. M. Amdahl, Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities, *AFIPS Conference Proceedings*, (30): 483–485.

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

[3] K. Briggs, Diagonally Dominant Matrix, *From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein.* https://mathworld.wolfram.com/DiagonallyDominantMatrix.html

[4] S. Corsaro, K. Mikula, A.Sarti, and F. Sgallari, Semi-implicit co-volume method in 3D image segmentation, *SIAM J. Sci. Comput.*, vol. 28, num. 6, p. 2248 – 2265, 2006.

[5] A. Dervieux, and F. Thomasset, A finite element method for the simulation of a Rayleigh-Taylor instability, *In: Rautmann R. (eds) Approximation Methods for Navier-Stokes Problems. Lecture Notes in Mathematics,* Springer, Berlin, Heidelberg, 1980, vol 771.

[6] A. Dervieux, and F. Thomasset, Multifluid incompressible flows by a finite element method, *In: Reynolds W. C., MacCormack R. W. (eds) Seventh International Conference on Numerical Methods in Fluid Dynamics. Lecture Notes in Physics,* Springer, Berlin, Heidelberg, 1981, vol 141.

[7] H. Digabel, and C. Lantuejoul, Iterative Algorithms, *In Proc. of 2nd European Symposium on Quantitative Analysis of Micro-structures in Material Science*, pages 85 – 99, Stuttgart, 1978. Riederer.

[8] L. C. Evans, and J. Spruck, Motion of level sets by mean curvature I, *J. Differential Geom.*, 33 (1991), 635 – 681. doi:10.4310/jdg/1214446559.

[9] A. X. Falcao, and J. K. Udupa, Segmentation of 3D Objects using Live Wire, *In Proc. of SPIE Medical Imaging*, volume 3034(1), pages 228 – 239, 1997.

[10] A. X. Falcao, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. de Alencar Lofufo, User-steered image segmentation paradigms: Live-wire and live-lane, *Graphics Models and Image Processing*, 60(4):223 – 260, 1998.

[11] A. X. Falcao, K. Jayaram, J. K. Udupa, and F. K. Miyazawa, An ultra-fast usersteered image segmentation paradigm: Live-wire-on-the fly, *Proc. of SPIE Medical Imaging*, 3661: 184 – 191, 1999.

[12] E. Faure, T. Savy, B. Rizzi, C. Melani, O. Drblíková, D. Fabregès, 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. 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, An algorithmic workflow for the automated processing of 3D+time microscopy imaging of developing organisms and reconstruction of their cell lineage, *Nature Communications*, 7 (2016), 8674.

[13] P. Frolkovič, K. Mikula, N. Peyriéras, and A.Sarti, A counting number of cells and cell segmentation using advection-diffusion equations, *Kybernetika*, vol. 43, num. 6, p. 817 – 829, 2007.

[14] A. Handlovičová, K. Mikula, and F. Sgallari, Semi-implicit complementary volume scheme for solving level set like equations in image processing and curve evolution, *Numerische Mathematik*, (2003) 93: 675 – 695. doi: 10.1007/s002110100374.

[15] M. Kass, A. Witkin, and D. Terzopoulos, Snakes: active contour models- *International Journal of Computer Vision*, 1(4): 321 – 331, 1988.

[16] J. Koenderink, The structure of images, *Biological Cybernetics*, vol. 50, pp. 363–370, 1984.

[17] B. Kósa, K. Mikula, M. O. Uba, A. Weberling, N. Christodoulou, and M. Zernicka-Goetz, 3D Image Segmentation Supported by A Point Cloud, *Discrete and Continuous Dynamical Systems — Series S (DCDS-S)*, 2021, 14(3): 971 – 985.

[18] K. Mikula, N. Peyriéras, M. Remešíková and 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.

[19] K. Mikula, and M. Remešíková, Finite volume schemes for the generalized subjective surface equation in image segmentation, *Kybernetika*, Vol. 45, No. 4 (2009) pp. 646 – 656.

[20] K. Mikula, N. Peyriéras, M. Remešíková, and M. Smíšek, 4D Numerical Schemes for Cell Image Segmentation and Tracking. In: Fořt J., Fürst J., Halama J., Herbin R., Hubert F. (eds) Finite Volumes for Complex Applications VI Problems & Perspectives, *Springer Proceedings in Mathematics*, 4(2011). https://doi.org/10.1007/978-3-642-20671-9_73

[21] K. Mikula, N. Peyriéras, and R. Špir, Numerical algorithm for tracking cell dynamics in 4D biomedical images, *Discrete and Continuous Dynamical Systems — Series S (DCDS-S)* Volume 8, Number 5 (2015) pp. 953 – 967.

[22] K. Mikula, R. Špir, M.Smíšek, E. Faure, and N. Peyriéras, Nonlinear PDE based numerical methods for cell tracking in zebrafish embryogenesis, *Applied Numerical Mathematics*, Vol. 95 (2015) pp. 250 – 266

[23] K. Mikula, R. Špir, and N. Peyriéras, Parallelization and validation of algorithms for zebrafish cell lineage tree reconstruction from big 4D image data, *Acta Polytechnica Hungarica*, Vol. 14/5 (2017) pp. 65 – 84.

[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, *in Finite Volumes for Complex Applications V: Problems and Perspectives (Eds. R.Eymard, J.M.Herard),*, ISTE and WILEY, London, 2008, pp. 585 – 592.

[25] K. Mikula, N. Peyriéras, M. Remešíková, and O. Stašová, Segmentation and analysis of 3D zebrafish cell image data, *Proceeding of the 3rd International Congress on Image and Signal Processing CISP-BMEI 2010, Yantai, China*, Vol. 3 (2010) pp. 1444 – 1448.

[26] 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, (Eds. Jasjit S. Suri and Aly Farag), ISBN: 0-387-31204-8*, 2007, pp. 123 – 160.

[27] K. Mikula, A. Sarti, and F. Sgallari, Co-volume level set method in subjective surface based medical image segmentation, *in: Handbook of Medical Image Analysis: Segmentation and Registration Models (J. Suri et al., Eds.), Springer, New York*, 2005, pp. 583–626.

[28] K. Mikula, A. Sarti, F. Sgallari, Co-volume method for Riemannian mean curvature flow in subjective surfaces multiscale segmentation, *Computing and Visualization in Science*, Vol. 9, No. 1 (2006) pp. 23–31.

[29] S. Osher, and J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, *Journal of Computational Physics*, Vol. 79, Num. 1, pp. 12 – 49, 1988.

[30] N. A. Otsu, A Threshold Selection Method from Gray-Level Histograms, *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62 – 66, 1979.

[31] S. Park, Tracking cells in 3D + time image sequences — written part of dissertation exam, *Faculty of Civil Engineering, Slovak University of Technology in Bratislava*, 2020.

[32] P. Perona, and J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (1990), 629 – 639. doi:10.1109/34.56205.

[33] B. Preim, and D. Bartz, Image Analysis for Medical Visualization, *Visualization in Medicine*, 83 – 133, (2007). doi:10.1016/b978-012370596-9/50007-9.

[34] A. Sarti, R. Malladi, and J. A. Sethian, Subjective Surfaces: A Geometric Model for Boundary Completion, *International Journal of Computer Vision*, Vol. 46 Num. 3, pp. 201 – 221, 2002.

[35] A. Sarti, R. Malladi, and J. A. Sethian, Subjective Surfaces: A Method for Completing Missing Boundaries, *PNAS*, Vol. 12, Num. 97, pp. 6258 – 6263, 2000.

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

[37] R. D. Skeel, Numerical Linear Algebra, *Purdue University*, (2006), pp. 1–41. https://www.cs.purdue.edu/homes/skeel/CS515/1.pdf

[38] R. Špir, K. Mikula, and N. Peyriéras, Cell Lineage Tree Reconstruction from Time Series of 3D Images of Zebrafish Embryogenesis. In: Chen CS., Lu J., Ma KK. (eds) Computer Vision – ACCV 2016 Workshops. ACCV 2016. Lecture Notes in Computer Science, vol 10117. *Springer*, Cham (2017) pp. 539 – 554.

[39] M. O. Uba, K. Mikula, Z. Krivá, H. Nguyen, T. Savy, E. Kardash, and N. Peyriéras, 3D Cell Image Segmentation by Modified Subjective Surface Method, *Tatra Mountains Mathematical Publications*, (2020) 75, 147 – 162. doi: 10.2478/tmmp-2020-0010.

[40] M. O. Uba, K. Mikula, Z. Krivá, H. Nguyen, T. Savy, E. Kardash, and N. Peyriéras, Application of Modified Subjective Surface Method to 3D Cell Membrane Image Segmentation, *ALGORITMY 2020, Conference on Scientific Computing*, Podbanske, Slovakia, 2020, *Proceedings of contributed papers*, 51 – 60.

[41] L. Vincent, and P. Soille, Watersheds in Digital Spaces: an Efficient Algorithm Based on Immersion Simulations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583 – 598, 1991.

[42] P. Witkin, Scale-space filtering, *in Proc. 8th Int. Joint Conf. Art. Intel l.*, (Karlsruhe, West Germany), pp. 1019–1022, Aug. 1983.

[43] C. Zanella, M. Campana, B. Rizzi, C. Melani, G. Sanguinetti, P. Bourgine, K. Mikula, N. Peyrieras, and A. Sarti, Cells Segmentation from 3-D Confocal Images of Early Zebrafish Embryogenesis, *IEEE Transactions on Image Processing*, (2010) Vol.19, No.3 pp. 770 – 781.

[44] C. Zanella, B. Rizzi, C. Melani, M. Campana, P. Bourgine, K. Mikula, and N. Peyriéras, A.Sarti, Segmentation of Cells from 3-D Confocal Images of Live Zebrafish Embryo, *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, (2007) No.1, pp. 6028 – 31.

## List of publications

UBA, Markjoe Olunna — MIKULA, Karol — KRIVÁ, Zuzana — NGUYEN, Hanh — SAVY, Thierry — KARDASH, Eléna — PEYRIÉRAS, Nadine. 3D Cell Image Segmentation by Modified Subjective Surface Method, *Tatra Mountains Mathematical Publications*, (2020) 75, 147 – 162. DOI: 10.2478/tmmp-2020-0010, ISSN: 12103195, SCOPUS: 2-s2.0-85085765653.

UBA, Markjoe Olunna — MIKULA, Karol — KRIVÁ, Zuzana – NGUYEN, Hanh — SAVY, Thierry — KARDASH, Eléna — PEYRIÉRAS, Nadine. Application of Modified Subjective Surface Method to 3D Cell Membrane Image Segmentation, *ALGORITMY 2020, Conference on Scientific Computing*, Podbanske, Slovakia, 2020, *Proceedings of contributed papers*, 51 – 60, ISBN: 978-80-227-5032-5, WOS: 000621768800006.

KÓSA, Bálazs — MIKULA, Karol — UBA, Markjoe Olunna — WEBERLING, Antonia — CHRISTODOULOU, Neophytos — ZERNICKA-GOETZ, Magdalena. 3D Image Segmentation Supported by A Point Cloud, *Discrete and Continuous Dynamical Systems — Series S (DCDS-S)*, 2021, 14(3): 971 – 985, DOI: 10.3934/dcdss.2020351, ISBN: 1937-1632, WOS: 000608373600015.