Tatra Mt. Math. Publ. **91** (2025), 101–128
DOI: 10.2478/tmmp-2025-0019

TATRA MOUNTAINS
Mathematical Publications

SEGMENTATION OF MACROPHAGES ON A QUADTREE GRID USING THE P4EST LIBRARY

Zuzana Krivá

Department of Mathematics and Constructive Geometry, Faculty of Civil Engineering, Slovak University of Technology, Bratislava, SLOVAKIA

ABSTRACT. The paper proposes the numerical schemes applied in the macrophage segmentation process on a quadtree grid, generated adaptively with respect to intensity of the data. Because the data are sparse and, in general, distinctly rectangular images, to generate the mesh the library 'p4est' (parallel forest) has been selected. The library offers tools to connect multiple trees into a 'forest' ('4est'), enabling parallel processing ('p4est'). The segmentation methods used can be solved with PDEs commonly used in image processing, the linear heat equation and the modified SUBSURF model, for which we proposed explicit and semi-implicit schemes based on the finite volume space discretisation. The choice of numerical algorithms is adapted to the way the grid elements are iterated in the environment of the library.

In this paper we focus on a single quadtree. From the numerical point of view, the extension to the forest is straightforward. Description of the library's principles with respect to the grid generation, its elements iteration, refining, coarsening and balancing the grid with the help of so-called *callback* functions and parallelism can be found in more detail, e.g., in [7], [8].

1. Introduction

Macrophage is a type of white blood cell that, if working properly, is an important part of an immune system. In the case of malfunctioning, they can become pathogenic and contribute to inflammatory diseases and cancer [19]. Image segmentation of macrophage data can be the first step in analyzing

2020 Mathematics Subject Classification: 68U10, 35K61, 65M08.
Keywords: image processing, non-uniform grids, p4est, quadtree mesh, adaptive discretization, segmentation, adaptive thresholding, macrophages.
Supported by the Marie Sklodowska-Curie grant agreement ID 955576, APVV-23-0186, APVV-10-0460.


FIGURE 1. An example of the data from the source [21]: 75 time slices of the dimensions $3755 \times 683 = 2564665$ pixels (2.6 MB). In experiments described in [7], the adaptive grids corresponding to images in the data set have about 0.04 % elements: the median value being 3709, minimum 2713 and maximum 5017 squares.

the characteristics of macrophages. The macrophage changes its shape when interacting with surrounding objects as it moves toward a wound. The remarkable dynamic plasticity of macrophages causes extreme irregularity in their shapes and variability in image intensity within them. All of this makes segmentation a difficult task.

Quadtree grid. At Fig. 2 b) we see an example of a quadtree grid adapted to intensity. The basic quadtree grid is constructed over the data of the dimension $2^n \times 2^n$. However, the data in Fig. 1 are distinctly rectangular and the basic quadtree representation is not suitable. Looking at Fig. 1 we see that most pixels do not contain relevant information and it can be possible to use a grid that contains larger and smaller elements, which is finer in the vicinity of macrophages and coarser elsewhere. Such a representation can significantly reduce the number of elements. To process this kind of data, we decided to use the library **p4est**. The name is abbreviation for *parallel forest*. This library, written in C language, enables management and processing of a collection of quadtrees (2D) or oct-trees (3D) which are called forests. It supports MPI (Message Passing Interface), and thus from the beginning parallel processing of operations can be taken into account. It offers tools to create a forest determining the connectivity of quadtrees or oct-trees, creating the grid, iterating the forests, refining and coarsening the grid, balancing, etc. Its source code is available under GNU GPL v 2.0 license and the authors are Carsten Burstedde, Lucas C. Wilcox and Tobin Isaac [5, 1, 2].

Remark 1. The grid mentioned above will be called an adaptive grid with respect to the intensity, or simply an *adaptive grid*. On the other hand, when we mention the *adaptive thresholding method*, we will mean a method that assigns a particular threshold value to each element of the grid. For our problem, adaptive (local) thresholding methods outperformed methods working with a global threshold value.

The processed data in this paper contains 74 cuts of selected macrophages in the database of time slices in [21], rescaled to the dimensions 128×128.

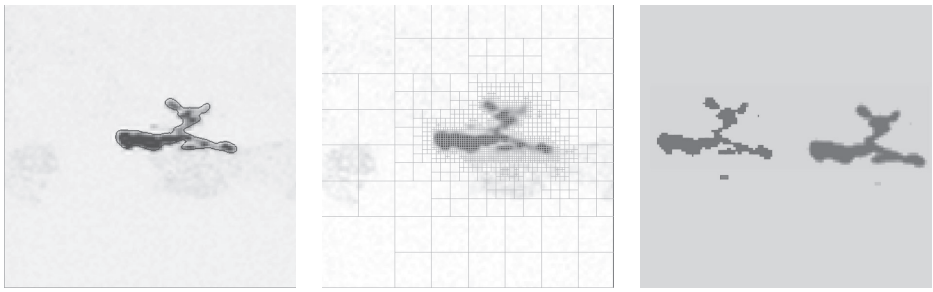


FIGURE 2. Left: the initial data and the isoline representing the result of the segmentation. Middle: grid after refinement procedures and balancing. Right: two results of the adaptive thresholding used as two inputs of the SUBSURF model: the 1st one (black values only) affects the weight of the curvature during isolines movement and the edge detection and the 2nd one – black + dark grey values – represents the segmentation function, isolines of which will be moving towards the boundaries of the macrophage to define its shape. On the right: the final state of the segmentation function, the representing isoline over the original data is on the left.

In Fig. 2 we see an example of a cut: irregularity and varying intensity can be observed as well as noise, often present in the vicinity of objects (here, the intensity is inverted to see the noise better). Using a simple thresholding method, i.e., the classical (global) Otsu's method, can result in the splitting of a macrophage into several fractions, as we see in Fig. 2 b), below the grid. Adaptive thresholding methods are often able to avoid fragmentation as in Fig. 2 c) on the right and produce an initial segmentation function covering the whole macrophage. The noise can be often successfully suppressed when building the grid: in b) we can see the grid placed on the original data with large squares on originally noisy data, their intensity is set to the average intensity of the noisy background below. This grid is balanced, i.e., the ratio of sides can be 1:1, 1:2 or 2:1 only.

The SUBSURF method [17] is applied to get a final segmentation curve representing the shape of the macrophage: it helps to suppress the remaining noise after grid construction (Fig. 10), to smooth the boundaries of the macrophages, and, first of all, it is able to connect fragmented parts of macrophages into one (as long as they are not too far apart), see (Fig. 8 and Fig. 9). The proposed segmentation method, explored in [18] for a uniform grid, combines adaptive thresholding methods with the SUBSURF approach that does not require cell nuclei centers or other reference information. The classical SUBSURF method has two inputs (see eq.(24)), the first is the object to be segmented, providing the information to detect the boundaries to stop the curvature motion of the isolines there and also to construct a vector field oriented toward the edges,

which draws the isolines in its reach closer to the boundary of the segmented object. The second input is the segmentation function itself, with a set of suitable isolines, and at the end, one of them will represent the result of the segmentation.
Adaptive thresholding methods. For the first input, it is important that it captures the shape of the macrophage as best as possible. In [14], global thresholding methods for SUBSURF have been explored, but in the end the adaptive thresholding methods [16], providing a specific threshold for each pixel, outperformed the former.

In [18], the Niblack-Bernsen method on a regular grid has been studied and tested to obtain a better connectivity of the thresholded data. In [15] and also[13], the local (adaptive) Otsu's method has been used successfully on a regular grid. The method employs the classical Otsu's method [12] within a local window centered at a pixel (i, j) to determine thresholds $T_{i,j}$ for this pixel. However, in the *p4est* environment, the implementation of a local window is not straightforward. It was the reason why we have instead selected the Niblack method [6]. It is based on a computation of a standard deviation and a mean in a local window, and if we replace them with a weighted mean and a weighted standard deviation, we can get the latter by solving the linear heat equation (Section (4.2)) for time(scale) corresponding to the radius of the window. The results of the work [18] confirmed the suitability of using this method to obtain both inputs for the modified SUBSURF method.
Adaptive grid. In Fig. 2b) we see an example of an adaptive grid based on the quadtree representation. The location of a macrophage is refined down to a pixel level, and towards the border the background data is represented by gradually enlarging quadrants. The example shows a discretization of an image with original size 128 x 128=16 384 pixels, reduced to about 800 elements. In the case of sparse data, as slices of macrophages often are, the reduction is larger, and paired with parallelism, the computational time can be reduced significantly.

2. General overview of p4est

2.1. The principle of management

The principle of management (control) is based on *callback* functions: by the *callback* function we mean an arbitrary used-defined function or procedure called by another function. In the framework of *p4est*, the input of these functions is prescribed, and after providing its address, the function is called internally. Application data is referred, in general, using (void) pointers, which must be casted. The elements of the grid, the quadrants, are assigned natural ordering by a traversal across all leaves. By the equivalence of tree nodes and quadrants, this one-dimensional sequence corresponds to storage in the order of Morton Z-curve, belonging to the *space filling curves* [20] and a particular quadrant

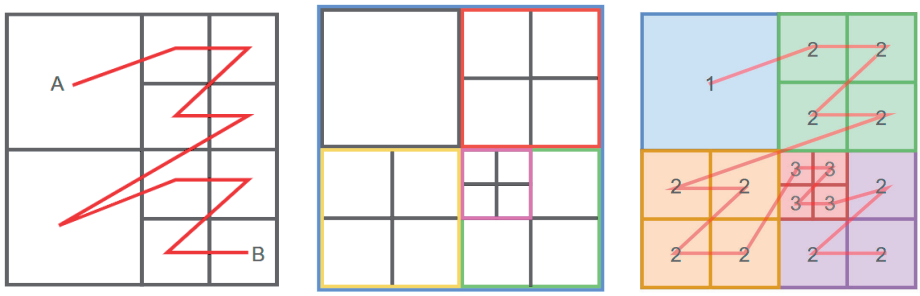


FIGURE 3. Left: Z-order of traversing quadrants [20]. Middle: tree representation of the quadtree on the right with white color denoting the inner nodes. Right: The colors of quadrants correspond to leaves in the tree, the numbers show the level of refinement [21].

is defined by the coordinates of the left lower corner, level of refinement, and a number of its tree in the forest. The representation of the forest by the Z curve (see Fig. 3) allows a very simple parallelization, because the whole problem is reduced to the partition of a 1-dimensional array into (not necessary) equally sized portions. The library supports export to *.pvtu* format for data on irregular grids stored in multiple files. This format is supported by visualization software *Paraview*. More details can be found in [5], [2] or [7].

Iterating the forest. To decide which algorithm and which method is suitable for the deployment of *p4est*, the way in which the forest is iterated is crucial. Let us restrict ourselves to 2D and to a simple quadtree. We can iterate quadrants, edges of quadrants, and corners of quadrants calling the *p4est_iterate* function. It is a function that expects individual types of *callback* functions and will be called for each iterable element. The *callback* functions for each type of elements are defined separately. In the following example, we can see how this function is called. The first argument is always a pointer to the forest, the second and the third are used for parallelization, which is not our case, and then tree pointers to *callback* functions for volumes, edges, and corners can follow.

```
p4est_iterate (p4est,    /* pointer to the forest */
        NULL,            /* the ghost layer */
        NULL,            /* the synchronized ghost data */
        set_volume,      /* e.g., setting value of the volume*/
        update_flux,     /* e.g., edge's contribution to flux */
        NULL);           /* no callback for the corners*/
```

The *callback* functions are called in the order specified by the Z-curve (Fig. 3). Often we call only one *callback* function, but in general, if there are more *callback* functions in the list of arguments like above, the *callback* function for surfaces (here quadrants) is called first. The *callback* function for the edges is called

the second, but only after the *callback* function for the surface is called on the corresponding quadrants that share the edge.

The *callback* function, which is called for a quadrant, does not provide any information about its neighbors, and when we implement algorithms we have to change our perspective a bit when compared to uniform discretizations, where we can access the adjacent quadrants just by changing the index. Therefore, if we want to work with neighbors, the operations must be performed in the edge *callback* functions, because all quadrants that share the given edge are available. Let us mention that any edge *callback* functions can be used only on a balanced quadtree (forest), i.e., there are two or three shared quadrants inside and one on the border.

For example, the calculation of edge flow depends on the edge across which the flow is calculated: from which side the edge is viewed and whether the edge is hanging or not, i.e., if the edge appears in the faces of adjacent elements with different refinement levels. All these cases must be treated. (Because each edge is iterated just once, we must often update information for all shared quadrants, thus we need to deal with internal numbering of sides in the quadrants.) If we want to calculate the volume flow, we need information about all the flows through all the edges. This operation cannot be performed within a single *p4est_iterate* call, because no edge *callback* functions have yet been called when the *callback* functions are called on the volume. For example, when we implement a linear heat equation to get an edge flux, we need just information stored in the quadrants sharing the edge, and this information is available. In these quadrants, we update the corresponding volume flux variables, and the process is finished when the edge iteration finishes. Then, to use the volume flux, we iterate once again with the volume *callback* function.

The problem can occur when we need to access more neighbors than mentioned above, for example when we evaluate a gradient over the edge. In this case, the *callback* functions would lead to a confusing code. Because of these limitations, there has been considerable effort to use algorithms that need to access neighboring quadrants as little as possible. We achieve this by using auxiliary variables in the data structure of a quadrant (e.g., for midpoints of the edges) that enable one to evaluate all volume information locally. All of these facts influenced the choice of our algorithm.

2.2. Using the forest for image processing

Our first goal is to create a simplified version of the image, with maximal refinement in the macrophage and in its vicinity, as displayed in Fig. 2b)and all operations are performed on this simplified adaptive (to intensity) grid. In general, first we must create the initial discretization of the forest, i.e., how many and how big quadtrees it is composed of, following some optimality criterion with respect to the number of redundant pixels. Then, we create an initial

adaptive forest with intensities of the quadrants derived from the input data. Here, the forest consists of only one quadtree.

In our application, while reading the image data, we evaluate a global threshold value using the Otsu's method to separate the macrophages from the background: the quadrants with intensity values exceeding this threshold are initially identified as macrophage regions; they are refined to the pixel level. However, due to variations in the quality of the input images, some macrophage portions might be incorrectly classified as background. Therefore, to address this problem, we implement a secondary refinement step. Quadrants smaller than a predefined size are further analyzed by comparing their maximum and minimum pixel intensities. In this way, we reduce the number of quadrants significantly. The value of quadrants covering more than one pixel is set to the pixels's mean intensity. In the end, the grid is balanced. To do this, we define *callback* functions for the refinement and balancing functions. For further details, refer to [8].

3. Methods and numerical schemes

One of the chosen models is a modified version of the SUBSURF model, as previously discussed in Section 1. The first input is not the image to be segmented, nor the combination of the initial image and thresholding as in [14]. Instead, it is a binary image generated by applying Niblack's adaptive thresholding, resulting in pixel values of 0 and 255. The second input — the segmentation function — is also derived from the adaptive thresholding resulting in values 0 and 255, but it requires a longer computation time for solving the linear heat equation. The segmentation function is obtained as a maximum of the first and second thresholds. The model then evolves the isolines of the segmentation function towards the edges defined by the first threshold; the curvature motion of these isolines is slowed down near these edges, while simultaneously being attracted towards them by a vector field derived from the first threshold. There are some possibilities of how to weight the motion of isolines by the curvature and their attraction to the edges, which allows for a trade-off between achieving more accurate macrophage shape representation and preventing the segmentation from fragmenting. The specific strategies will be detailed in the section *Numerical experiments*.

For image processing, the finite volume space discretization on balanced adaptive grids has been previously explored, e.g., in [11], [10] or [9]. These methods typically operate either on local neighborhoods or require access to the endpoints of finite volume edges, which can present a certain disadvantage as discussed earlier. Inspired by [4]'s approach for regular grids, we employ a different strategy. By introducing auxiliary points at the midpoints of edges, computations can be

performed locally, and the values at these auxiliary points are updated using conservation principles.

While [7] presents a method compatible with *p4est* edge iteration scheme, it only guarantees conservation principles for the linear heat equation and extending this for nonlinear models is not as straightforward as for the LHE or a regular grid. In this paper, we derive the scheme for the SUBSURF model that satisfies the conservation principles on the adaptive grid described above.

3.1. Outline of the method

The methodology comprises the following key steps:

(1) Grid Initialization: create an initial (unbalanced) quadtree grid using a combination of global Otsu and max-diff methods), assigning initial values and balancing the grid
(2) Adaptive Thresholding: apply Niblack's method with varying time scales to obtain two binary images (see (2)–(4)), numerically solved by the linear heat equation (subsection 3.3). The first one is utilized to construct the diffusion coefficients for the SUBSURF model, the initial condition for the SUBSURF model is derived by combining the first and second binary images.
(3) The SUBSURF is applied, solving the nonlinear model (24).

3.2. The Niblack adaptive thresholding method

Let (i, j) be the coordinates of a particular pixel in the intensity array I. The local threshold $Q(i, j)$ depends on a local mean of intensities $\mu_R(i, j)$ and a standard deviation $\sigma_R(i, j)$, both calculated in a window with a radius R, centered at the reference point (i, j). $Q(i, j)$ is given by the relationship

$$Q(i, j) = \mu_R(i, j) + \kappa\, \sigma_R(i, j), \tag{1}$$

where κ is a suitable predefined constant.

Now we rewrite the standard deviation given by (2) into (3). In the following formulas, $H_R(i, j)$ represents the set of pixel indices from the local neighborhood of the pixel (i, j), of the radius R (i.e., the window with a reference point (i, j) and the radius R.)

$$\sigma_R(i,j) = \sqrt{\frac{1}{N} \sum_{(u,v)\in H_R(i,j)} \big(I(u,v) - \mu_R(i,j)\big)^2}, \tag{2}$$

$$\sigma_R(i,j) = \sqrt{\frac{1}{N}\Big(\sum I(u,v)^2 - 2\sum I(u,v)\mu_R(i,j) + \sum \mu_R(i,j)^2\Big)},$$

$$\sigma_R(i,j) = \sqrt{\overline{I(i,j)^2} - 2\mu_R(i,j)\mu_R(i,j) + \frac{N\mu_R(i,j)^2}{N}},$$

$$\sigma_R(i,j) = \sqrt{\overline{I(i,j)^2} - \mu_R(i,j)^2}, \tag{3}$$

where $\overline{I(i,j)^2}$ is a mean value of *intensity squares* in the local neighborhood. Solving the linear heat equation (LHE) is equivalent to performing weighted averaging using a mask with coefficients derived from the Gaussian function by integration over the mask.The effective radius of this mask increases proportionally with time t to solve LHE. It follows that applying LHE to the input data, we get the values of weighted $\mu_R(i,j)$ and applying LHE to the intensity squares, we obtain values of weighted mean of the intensity squares, which replaces $\overline{I(i,j)^2}$. This approach avoids manipulations with local neighborhoods. Applying (4), we obtain a unique threshold for each pixel (i,j), enabling the classification of pixels as background or macrophage.

When using the Niblack method, regions with a constant or nearly constant intensity require careful consideration. In these regions, the standard deviation is minimal, leading to a threshold that closely approximates the local average intensity. This can result in an unstable classification of the pixel. To address this issue, we introduce a constant term $\pm d$ to equation (3). This ensures a minimal shift in the calculated threshold from the local average, especially in very dark or very bright areas. The shift is applied selectively: (+d) for small standard deviations on a dark background, and (-d) for small standard deviations on a bright object with minimal or no noise. The resulting relationship can be expressed as follows:

$$Q(i,j) = \mu_R(i,j) + \kappa \sqrt{\overline{I(i,j)^2} - \mu_R(i,j)^2} \pm d. \tag{4}$$

3.3. The linear heat equation

The linear heat equation is used to perform adaptive thresholding with the Niblack method using (4), it can also be used to smooth the data, e.g., before evaluation of gradients.

We solve the following problem:

$$\frac{\partial u(x,y,t)}{\partial t} - \Delta u(x,y,t) = 0, \quad (x,y,t) \in Q_T \equiv \Omega \times I, \tag{5}$$

$$\frac{\partial u(x,y,t)}{\partial n} = 0, \quad (x,y,t) \in \partial\Omega \times I, \tag{6}$$

$$u(x,y,0) = u^0(x,y), \quad (x,y) \in \Omega, u_0 \in L_\infty(\Omega). \tag{7}$$

Here, $u(x,t)$ is an unknown function defined in $\Omega \subset R^2$, where Ω is a domain in a two-dimensional space, and the time interval $I = [0,T]$. Let $\mathbf{n}$ denote a unit outer normal vector to $\partial\Omega$ and $u^0(x)$ is an initial condition.

We employ the method of auxiliary values, u_σ, defined at points x_σ, located at the center of each quadrant edge σ. The LHE can be solved effectively without utilizing the auxiliary points, but their inclusion provides a framework for understanding the nonlinear schemes introduced later in the paper. The explanation with auxiliary points offers a more intuitive and perhaps easier-to-grasp foundation for the subsequent development of more complex nonlinear methods.

3.4. The finite volume numerical scheme for (5)

Let σ denote the edge of the finite volume p and let $\mathbf{n}_\sigma$ represent the outward unit normal to σ.

Given the grid, we integrate the diffusion equation over the finite volume p and apply the divergence theorem to obtain:

$$\int_p \partial_t u \, dX - \int_{\partial p} \nabla u . \mathbf{n}_p \, dS = 0. \tag{8}$$

We replace the time derivative with a finite difference using the uniform time step $\tau = t^n - t^{n-1}$, where t^{n-1}, t^n represent the previous and current time steps, respectively. Let u^n denote the solution in the n^{th} time step and let u_p^n represent the constant solution over the finite volume p in the n^{th} time step. Having the integral form (8) of the problem (5), let us denote the flux through the boundary σ at the time step t^n by

$$f_\sigma^n = \int_\sigma \nabla u^n \cdot \mathbf{n}_\sigma \, dS, \tag{9}$$

where ∇u^n is the gradient of the solution at time step t^n. Then, the implicit scheme can be written in the following general form:

$$(u_p^n - u_p^{n-1}) \, |p| = \tau \sum_{\sigma \in \mathcal{E}_p} f_\sigma^n, \tag{10}$$

where
u_p^n denotes the constant solution within the finite volume p at the time step t^n, $|p|$ is the volume of p, and $\mathcal{E}_p$ denotes the set of edges of the finite volume p.

The flux f_σ^n involves the normal derivative, which can be approximated numerically as:

$$\nabla u^n \cdot \mathbf{n}_\sigma \approx \frac{\left(u_\sigma^n - u_p^n\right)}{d_{p\sigma}}, \tag{11}$$

where $d_{p\sigma}$ is the distance between the representative point of p (the center of p) and the midpoint of the edge σ, and

$$f_\sigma^n \approx F_\sigma^n = \frac{|\sigma|}{d_{p\sigma}}(u_\sigma^n - u_p^n). \tag{12}$$

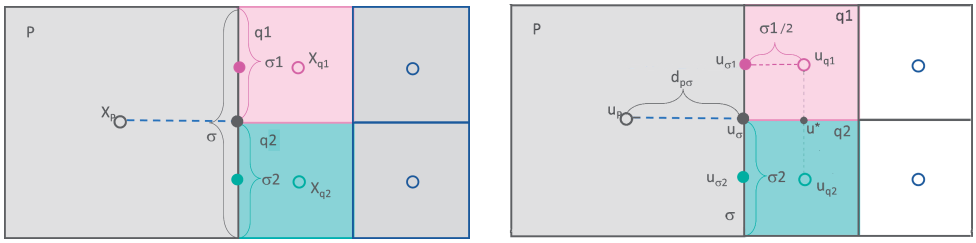
For a while, let us suppose that we have a uniform square grid. To apply conservative principles, we balance the fluxes for σ shared by p and q

$$\frac{|\sigma|}{d_{p\sigma}}(u_\sigma^n - u_p^n) = -\frac{|\sigma|}{d_{q\sigma}}(u_\sigma^n - u_q^n) \tag{13}$$

and get

$$u_\sigma^n = \frac{u_p^n + u_q^n}{2}. \tag{14}$$

3.5. Updating u_σ in a balanced adaptive grid



(a) Notation of quadrants and their representative points (empty circles) and auxiliary points x_σ, $x_{\sigma 1}$ and $x_{\sigma 2}$ (full circles).

(b) Notation of values. Quadrants: u_p, u_{q1}, u_{q2}. Auxiliary points on edges: u_σ, $u_{\sigma 1}$ and $u_{\sigma 2}$. New auxiliary point u^*.

FIGURE 4. Elements of the numerical scheme for the linear heat equation.

The geometric notation and the notation of the values are depicted in Fig. 4.

3.5.1. Updating values u_σ^n

In general, the method will be the following:

(1) On *nonhanging edges* the update of u_σ^n is straightforward, and the conservative principle is applied here ((13), (14)). In general, the values u_{q1} and u_{q2} are weighted by terms depending on the equation itself. In the case of the linear heat equation the weights are $\frac{1}{2}$. In the case of SUBSURF, the weights are obtained from nonlinear parameters determined by norms of gradients.

(2) *On hanging edges* we use the fact that one side of the edge is shared by two uniform quadrants and their mutual u_σ^n is updated as in the previous case. It is denoted by u^*, see Fig. 4b. The value u_σ belonging to the hanging edge σ (full black circle in Fig. 4b) is obtained by linear interpolation of u_p and u^*: $u_\sigma = \frac{1}{3}u_p + \frac{2}{3}u^*$.

(3) On hanging edges the discrete flux across σ contains

$$u_\sigma - u_p = \frac{1}{3}u_p + \frac{2}{3}u^* - u_p = \frac{2}{3}(u^* - u_p). \tag{15}$$

This flux for σ is subdivided into 2 parts using weights w_1 and w_2 (depending on the equation) and balanced with fluxes through σ_1 and σ_2, thus the corresponding u_{σ_1} and u_{σ_2} are evaluated.

3.5.2. Evaluating u_{σ_1}, u_{σ_2} and u_σ in the linear heat equation

Let us evaluate the flux through σ:

$$|\sigma|\frac{u_\sigma - u_p}{d_{p\sigma}} = |\sigma|\frac{u_\sigma - u_p}{\frac{|\sigma|}{2}} = 2\frac{2}{3}(u^* - u_p) = \frac{4}{3}\left(\frac{1}{2}(u_{q_1} + u_{q_2}) - u_p\right)$$
$$= \frac{4}{3}\left(\frac{1}{2}(u_{q_1} - u_p) + \frac{1}{2}(u_{q_2} - u_p)\right). \tag{16}$$

The weights subdividing the flux are $w_1 = w_2 = \frac{1}{2}$. We balance fluxes on σ_1 and $\frac{\sigma}{2}$:

$$\frac{4}{3}\frac{1}{2}(u_{q_1} - u_p) = 2(u_{q_1} - u_{\sigma_1}),$$
$$2u_{\sigma_1} = \frac{4}{3}u_{q_1} + \frac{2}{3}u_p$$

and finally

$$u_{\sigma_1} = \frac{1}{3}u_p + \frac{2}{3}u_{q_1} \tag{17}$$

and in a similar manner

$$u_{\sigma_2} = \frac{1}{3}u_p + \frac{2}{3}u_{q_2}. \tag{18}$$

Inserting the values u_{σ_1} and u_{σ_2} in (16) we obtain

$$u_\sigma = \frac{u_{q1} + u_{q2} + u_p}{3}. \tag{19}$$

3.6. Solving the linear system for the linear heat equation

Although the numerical schemes to solve the linear heat equation can be derived in a more straightforward way, we will stick to the method described in the previous paragraph and derive the implicit scheme. Because $\frac{|\sigma|}{d_{p\sigma}} = 2$

the time discretization of linear heat equation (10) can be written in the form

$$(u_p^n - u_p^{n-1}) = \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2(u_\sigma^n - u_p^n), \tag{20}$$

for each p we get an equation of the linear system in the form

$$\left(1 + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2\right) u_p^n - \frac{\tau}{m(p)} \sum_{\sigma \in \varepsilon_p} 2u_\sigma^n = u_p^{n-1}. \tag{21}$$

We use the Jacobi iterative method to solve it, where the $(l+1)^{\text{st}}$ iteration in the time step n will be evaluated as

$$u_p^{n(l+1)} = \frac{u_p^{n-1} + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2u_\sigma^{n(l)}}{\left(1 + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2\right)}, \tag{22}$$

followed by update of the u_σ values. The scheme stops when the tolerance of the residual vector is sufficiently small. One iteration of the scheme is performed using the *volume callback* function: all the necessary information is available and no access to neighbors is needed.

For this scheme, we show a simplified version of an implementation (without parallelism).

```
void LHEImplicit(p4est_t *p4est)
{
  p4est_iterate(NULL, NULL, NULL, SigmaInitialize, NULL);
  ...
  for(int i = 0; i < numOfSteps; ++i)
 {
  do
  {
   p4est_iterate(NULL,NULL,JacobiIterUpdate,NULL,NULL);//iter. vol.
   p4est_iterate(NULL,NULL,NULL,SigmaUpdate,NULL);      //iter edge
   p4est_iterate(NULL,NULL,Residuals,NULL,NULL);        //iter volume
  }
  while(residuals > TOL);
 }
p4est_iterate(Finalize, NULL, NULL);
}
```

Finally, we introduce a simplified example of the *edge callback* function *SigmaUpdate*

```
void SigmaUpdate}(p4est_iter_face_info_t *info)
{
 if (side_size == 1) sigma = quadData[0]->value;
                                  //border of the forest
 else if (side_size == 2)         //inner quadrant
 {
```

```
  ExtractData();
 ...
  if (numOfQuadrants == 2)        //nonhanging edge
  {
   EvaluateSigmaEqual();          //nonhanging edge, use(15)
  }
   else if (numOfQuadrants == 3)//hanging edge
   {
    EvaluateSigmaHanging();       //use (18)-(19)
   }
 }
}
```

The solution itself needs (except for initialization) to iterate with edge *callback* functions only when it updates the values u_σ. All necessary information about neighboring quadrants that share a processed edge is provided.

Remark 2. At the end of Section (3.3), we note that LHE can be efficiently solved without auxiliary points $u_{\sigma i}$. If these points were eliminated (achievable by combining equations (12), (14), (17), (18) and (19)), an explicit scheme would yield following relationship:

$$u_p^{n+1} = u_p^n + \frac{\tau}{|p|} \sum_{q \in N(p)} c_q (u_q^n - u_p^n), \tag{23}$$

where the constant c_q depends on the neighboring volumes q of the processed finite volume p and takes the values 1 or $\frac{1}{3}$. To implement this, we would introduce a variable for each volume to accumulate the sum of discrete fluxes. These fluxes would be computed for neighboring volume pairs (p and q) within an edge *callback* function. As each edge configuration is processed only once, the processing step would update two or three accumulated flux sums for the quadrants sharing the edge (or none on the boundary). Finally, the value u_p^{n+1} would be updated by calling a volume *callback* function in a straightforward manner. Advantage of this approach is that we avoid initialization of $u_{\sigma i}$.

3.7. Numerical scheme for the SUBSURF model

In the application, the subjective surface SUBSURF [17] aims to improve the segmentations resulting from the Niblack thresholding. It can close smaller holes, remove noise, and create smoother contours.

The classical SUBSURF method is defined by the following partial differential equation:

$$\frac{\partial u}{\partial t} - |\nabla u|\,\nabla \cdot \left(g(|\nabla I^0|)\frac{\nabla u}{|\nabla u|} \right) = 0, \tag{24}$$

$$\frac{\partial u(x,y,t)}{\partial n} = 0, \qquad (x,y,t) \in \partial\Omega \times I, \tag{25}$$

$$u(x,y,0) = u^0(x,y), \quad (x,y) \in \Omega, \;\; u_0 \in L_\infty(\Omega) \tag{26}$$

with g defined by

$$g(s) = \frac{1}{1+Ks^2}, \;\; K > 0, \;\; g(s) > 0. \tag{27}$$

where u is a level-set function that represents the segmentation and I^0 is the input image.

Rewriting the equation in the advection-diffusion form shows that the model consists of two parts, the advective part driving the isolines (curves) of u towards the boundaries of the object to be segmented (I^0) and this motion is combined with a curvature smoothing, regularizing the evolving segmentation. The vector field that attracts the segmentation curves towards the object boundaries is constructed from functions g and I^0 and is given by $-\nabla g(|\nabla I^0|)$.

3.7.1. Modification of the SUBSURF model

Due to the varying intensities within the macrophage, using the original macrophage image for I^0 directly can lead to a vector field that does not represent the boundaries of macrophages correctly, potentially resulting in 'tearing' of the macrophages in the segmentation. Following the approach in [18], we observed improved results when calculating the vector field using the thresholded image obtained by adaptive thresholding with the Niblack method. The initial segmentation function, u^0, is also derived from the Niblack method, but this time after applying LHE for a longer duration. We then obtain the maximum of both Niblack outputs, effectively creating an 'envelope' of the object. Computationally, getting I^0 and u^0 corresponds to obtaining the LHE solutions in two distinct time steps.

This modified approach, with improved I^0 and u^0, better captures contours of macrophages, giving a stronger vector field and a more suitable set of isolines within u^0. This modified model with comprehensive coverage of the macrophages increases the probability of obtaining complete and sufficiently accurate segmentation, minimizing the risk of fragmented results.

This is the basic algorithm; various modifications are possible (for example, how to obtain $u(x,y,0)$). In the following subsection, we keep the notation

$$I^0 = I(x,y,0) \quad \text{and} \quad u^0 = u(x,y,0),$$

but the inputs will be changed as described in this subsection.

3.7.2. Semi-implicit finite volume scheme

The time discretization is semi-implicit, giving the relationship (28).

$$\frac{u^n - u^{n-1}}{\tau} = |\nabla u^{n-1}|_\epsilon \nabla \cdot \left(g(|\nabla I^0|) \frac{\nabla u^n}{|\nabla u^{n-1}|_\epsilon} \right), \tag{28}$$

where $|\nabla u|$ in (24) is replaced by the Evans-Spruck regularization defined as

$$|\nabla u^n|_\epsilon \approx \sqrt{(\nabla u^n)^2 + \epsilon^2},$$

where ϵ is a nonnegative number close to zero.

Considering the space discretization, the magnitude of $|\nabla u_p^n|$ constant on p will be evaluated locally with help of auxiliary points u_σ as

$$N_p(u^n) = \sqrt{\frac{1}{|p|} \sum_{\sigma \in \varepsilon_p} \frac{|\sigma|}{d_{p\sigma}} (u_\sigma^n - u_p^n)^2} \tag{29}$$

and

$$|\nabla u_p^n|_\epsilon \approx f_p^n = \sqrt{\left(N_p(u^n)\right)^2 + \epsilon^2}, \qquad \text{see [4], [3].}$$

In the following, the term g_p^0 is evaluated just once at the beginning as $g(|\nabla I_p^0|)$. More details on the implementation can be found in Section 4. To get the weak formulation we first divide the equation (28) by $|\nabla u^{n-1}|_\epsilon$ and then we integrate it over p. The semi-implicit time discretization and finite volume space discretization lead to the linear system, with following equation for each p:

$$\frac{u_p^n - u_p^{n-1}}{\tau f_p^{n-1}} |p| - \sum_{\sigma \in \varepsilon_p} \frac{|\sigma|}{d_{p\sigma}} \frac{g_p^0}{f_p^{n-1}} (u_\sigma^n - u_p^n) = 0, \tag{30}$$

$$\frac{u_p^n - u_p^{n-1}}{\tau f_p^{n-1}} |p| - \sum_{\sigma \in \varepsilon_p} 2 \frac{g_p^0}{f_p^{n-1}} (u_\sigma^n - u_p^n) = 0, \tag{31}$$

$$u_p^n \left(1 + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2 g_p^0 \right) - \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2\, g_p^0\, u_\sigma^n = u_p^{n-1}. \tag{32}$$

Then the algorithm of the Jacobi iterative method is described by

$$u_p^{n(l+1)} = \frac{u_p^{n-1} + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2 g_p^0 u_\sigma^{n(l)}}{\left(1 + \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2 g_p^0 \right)} = \frac{u_p^{n-1} + \frac{2 g_p^0 \tau}{|p|} \sum_{\sigma \in \varepsilon_p} u_\sigma^{n(l)}}{\left(1 + \frac{8 \tau g_p^0}{|p|} \right)}. \tag{33}$$

Remark 3. From (30), the term f_p^{n-1} is eliminated. Each iteration must be followed by updating the value $u_\sigma^{n(l)}$ (see next subparagraph), where f_p^{n-1} is involved.

In the case of a uniform grid, let q be a neighbor of p from a set of its neighbors $N(p)$ and e_{pq} be an edge shared by p and q, then if we eliminate u_σ^n, given by

$$u_\sigma^n = \frac{g_p^0 f_q^{n-1} u_p^n + g_q^0 f_p^{n-1} u_q^n}{g_p^0 f_q^{n-1} + g_q^0 f_p^{n-1}}, \tag{34}$$

we get the following form of the linear system:

$$\frac{u_p^n - u_p^{n-1}}{\tau f_p^{n-1}}|p| - \sum_{q\in N(p)} \frac{|e_{pq}|}{d_{pq}} \frac{(u_q^n - u_p^n)}{\frac{1}{2}\frac{f_q^{n-1}}{g_q^0} + \frac{1}{2}\frac{f_p^{n-1}}{g_p^0}} = 0. \tag{35}$$

3.7.3. Updating u_σ, $u_{\sigma 1}$ and $u_{\sigma 2}$

Let us evaluate the flux on a hanging edge σ, shared by the quadrants p, $q1$ and $q2$. Using u_*^n on a non-hanging edge, for which we apply (34) and perform a set of steps similar to (3.5.2):

$$\begin{aligned}
&|\sigma|\frac{g_p^0}{f_p^{n-1}}\frac{u_\sigma^n - u_p^n}{d_{p\sigma}} = |\sigma|\frac{g_p^0}{f_p^{n-1}}\frac{u_\sigma - u_p}{\frac{|\sigma|}{2}} = \frac{4}{3}\frac{g_p^0}{f_p^{n-1}}(u_*^n - u_p^n)\\
&= \frac{4}{3}\frac{g_p^0}{f_p^{n-1}}\left(\frac{g_{q1}^0 f_{q_2}^{n-1} u_{q1}^n + g_{q2}^0 f_{q_1}^{n-1} u_{q2}^n}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}} - u_p^n\right)\\
&= \frac{4}{3}\frac{g_p^0}{f_p^{n-1}}\left(\frac{g_{q1}^0 f_{q_2}^{n-1} u_{q1}^n + g_{q2}^0 f_{q_1}^{n-1} u_{q2}^n}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}} + \frac{-g_{q1}^0 f_{q_2}^{n-1} u_p^n - g_{q2}^0 f_{q_1}^{n-1} u_p^n}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}}\right)\\
&= \frac{4}{3}\frac{g_p^0}{f_p^{n-1}}\left(\frac{g_{q1}^0 f_{q_2}^{n-1}}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}}(u_{q1}^n - u_p^n) + \frac{g_{q2}^0 f_{q_1}^{n-1}}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}}(u_{q2}^n - u_p^n)\right).
\end{aligned} \tag{36}$$

The discrete flux across σ will be split into two parts using two weights:

$$w_1 = \frac{g_{q1}^0 f_{q_2}^{n-1}}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}}, \tag{37}$$

$$w_2 = \frac{g_{q2}^0 f_{q_1}^{n-1}}{g_{q1}^0 f_{q_2}^{n-1} + g_{q2}^0 f_{q_1}^{n-1}}. \tag{38}$$

To get u_{σ_1}, we compare the first part of the flux (36) to flux through σ_1 (31):

$$\frac{4}{3}\frac{g_p^0}{f_p^{n-1}} w_1 (u_{q1}^n - u_p^n) = -2\frac{g_{q1}^0}{f_{q1}^{n-1}}(u_{\sigma_1}^n - u_{q1}^n), \tag{39}$$

$$u_{\sigma_1}^n = u_{q1}^n - \frac{2}{3}\frac{f_{q_1}^{n-1}}{g_{q_1}^0}\frac{g_p^0}{f_p^{n-1}} w_1 (u_{q_1}^n - u_p^n). \tag{40}$$

In (39) we now express the coefficient for $(u^n_{q_1} - u^n_p)$:

$$\frac{f^{n-1}_{q_1}}{g^0_{q_1}} \frac{g^0_p}{f^{n-1}_p} w_1 = \frac{f^{n-1}_{q_1}}{g^0_{q_1}} \frac{g^0_p}{f^{n-1}_p} \frac{g^0_{q_1} f^{n-1}_{q_2}}{g^0_{q_1} f^{n-1}_{q_2} + g^0_{q_2} f^{n-1}_{q_1}} \tag{41}$$

$$= \frac{g^0_p}{f^{n-1}_p} \frac{f^{n-1}_{q_1} f^{n-1}_{q_2}}{g^0_{q_1} f^{n-1}_{q_2} + g^0_{q_2} f^{n-1}_{q_1}} = \frac{\frac{g^0_p}{f^{n-1}_p}}{\frac{g^0_{q_1}}{f^{n-1}_{q_1}} + \frac{g^0_{q_2}}{f^{n-1}_{q_2}}} = A. \tag{42}$$

Similarly, we express the coefficient for $(u^n_{q_2} - u^n_p)$:

$$u^n_{\sigma_2} = u^n_{q2} - \frac{2}{3} \frac{f^{n-1}_{q_2}}{g^0_{q2}} \frac{g^0_p}{f^{n-1}_p} w_2 (u^n_{q_2} - u^n_p),$$

$$\frac{f^{n-1}_{q_2}}{g^0_{q_2}} \frac{g^0_p}{f^{n-1}_p} w_2 = \frac{f^{n-1}_{q_2}}{g^0_{q_2}} \frac{g^0_p}{f^{n-1}_p} \frac{g^0_{q2} f^{n-1}_{q_1}}{g^0_{q1} f^{n-1}_{q_2} + g^0_{q2} f^{n-1}_{q_1}} \tag{43}$$

$$= \frac{g^0_p}{f^{n-1}_p} \frac{f^{n-1}_{q_2} f^{n-1}_{q_1}}{g^0_{q1} f^{n-1}_{q_2} + g^0_{q2} f^{n-1}_{q_1}}$$

$$= \frac{\frac{g^0_p}{f^{n-1}_p}}{\frac{g^0_{q_1}}{f^{n-1}_{q_1}} + \frac{g^0_{q_2}}{f^{n-1}_{q_2}}} = A. \tag{44}$$

Rewriting (39) we get

$$u^n_{\sigma_i} = u^n_{q_i} - \frac{2}{3} A (u^n_{q_i} - u^n_p), \tag{45}$$

and finally,

$$u_{\sigma_1} = \left(1 - \frac{2}{3} A\right) u_{q_1} + \frac{2}{3} A u_p, \tag{46}$$

$$u_{\sigma_2} = \left(1 - \frac{2}{3} A\right) u_{q_2} + \frac{2}{3} A u_p. \tag{47}$$

In all our experiments, the value of $\frac{2}{3}A$ was below 0.67, so u_{σ_1} and u_{σ_2}, were weighted averages of u_{σ_i} and u_p.

At the end we summarize the semi-implicit scheme. First, we initialize u^0_σ values (see next section) and evaluate g^0_p and f^0_p. Then, in a loop, for each time step n we repeat:

- evaluate $u^{n(l+1)}_p$ (33) and evaluate residual vector to stop iterations
- if stopping condition is not satisfied, update values of auxiliary points $u^{n(l+1)}_\sigma$ and go to new iteration,

- when iterations finish with $l = l_{\text{stop}}$, update values of

$$u_p^{(n+1)0} = u_p^{n(l_{\text{stop}})}, \quad f_p^{n+1} \quad \text{using} \quad u_\sigma^{n(l_{\text{stop}})}$$

  and go to the next time step.

3.7.4. Explicit finite volume scheme

Using explicit time discretization, we get:

$$\frac{u_p^{n+1} - u_p^n}{\tau f_p^n}|p| - \sum_{\sigma \in \varepsilon_p} \frac{|\sigma|}{d_{p\sigma}} \frac{g_p^0}{f_p^n}(u_\sigma^n - u_p^n) = 0,$$

$$u_p^{n+1} - u_p^n - \frac{\tau}{|p|} \sum_{\sigma \in \varepsilon_p} 2g_p^0 \left(u_\sigma^n - u_p^n \right) = 0,$$

and finally,

$$u_p^{n+1} = \left(1 - \frac{8\tau g_p^0}{|p|} \right) u_p^n + \frac{2\tau g_p^0}{|p|} \sum_{\sigma \in \varepsilon_p} u_\sigma^n. \tag{48}$$

The stability condition is:

$$\tau < \frac{|p|}{8g_p^0} < \frac{|p|}{8} < \frac{1}{8}. \tag{49}$$

First, we initialize u_σ^0 values (see next section) and evaluate g_p^0 and f_p^0 and then, in a loop, repeat:

- evaluate u_p^{n+1} using (48),
- update values of auxiliary points u_σ^{n+1},
- update values of f_p^{n+1}.

4. Evaluating g_p^0

To initiate the numerical solution of the SUBSURF model, we must first set the values of $I_{\sigma_i}^0$ and $u_{\sigma_i}^0$ at the midpoints of the edges to subsequently evaluate $g(|\nabla I^0|)$ and $|\nabla u^0|$). To do this, we employ the notation in Fig. 5, which is used for this purpose. Iterating by edge *callback* function, we first set the values I_σ on the non-hanging edges to the mean value of u_{q1} and u_{q2} Then we process hanging edges: first, iterating the edges, we set values only for larger edges in the configurations, using (19). Afterwards, again iterating with edge *callback* functions, we set values for smaller edges in the configuration and stop, when all edges are set.

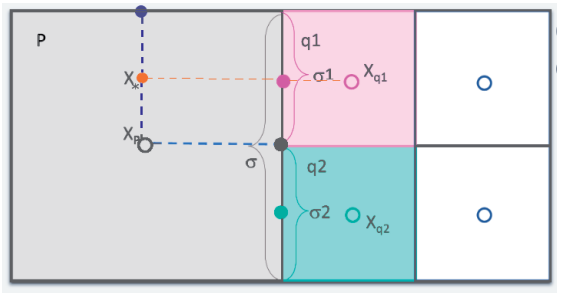


FIGURE 5. Notation for initialization of values at midpoints of the edges.

For example, let us evaluate the value u_σ at the magenta point. If the dark blue value is set (i.e., its edge is not a smaller edge in the hanging edge configuration), applying linear interpolation, we evaluate the X_* value (at the orange point) and then interpolate the value at the magenta point, otherwise we skip the evaluation. We repeat iteration until all points are set (in our examples twice). Then we evaluate $|\nabla I^0|_p$, using (29) and g_p^0 using (27), both with volume *callback* functions. These values do not change during evolution of the SUBSURF model.

Analogously, we initialize the values of u_σ^0 of the segmentation function, enabling evaluation of $|\nabla u^0|_p$ and f_p^0. Subsequently, in each time step, these values are updated by calculating $u_{\sigma_i}^n$ based on the conservation principle.

The numerical experiment dealing with the error takes three grids that have been subsequently refined four times (shown in Fig. 6). The function

$$u(x,y) = \frac{x^2+y^2}{2}$$

was used to initialize the (representative) values u_p in the adaptive grids. The values $u_{\sigma i}$ were calculated, and the numerically evaluated $|\nabla u|_p$ was compared with the exact value. The resulting EOC in all three cases was approximately 1.55.

5. Numerical experiments

Displaying segmentation results. They are visualized by drawing isolines or as thresholds of the final SUBSURF segmentation function. Many results will be displayed as in Fig. 7 a) — a blue or red isoline over the initial data, allowing easy comparison of the segmentation result with the original image. The cyan color denotes the result for the semi-implicit scheme, and the red color denotes the result for the explicit scheme. Because it is not possible to display isolines on a quadtree grid in the Paraview software, we transformed the data into a regular (structured) grid and changed the format to point data, for which

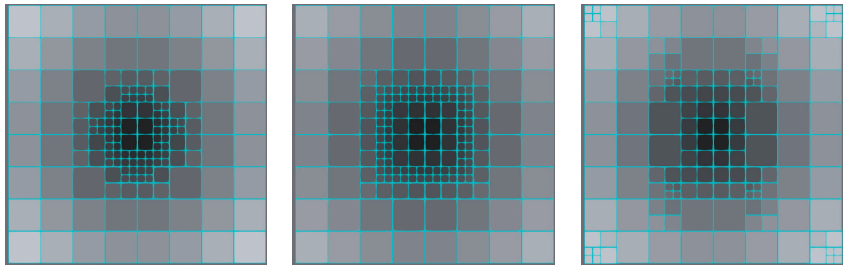


FIGURE 6. Evaluation of $|\nabla I^0|_p$ and EOC. The three balanced grids obtained by subsequent refinement of initial grids - each square was replaced by its four children. The function $\frac{x^2+y^2}{2}$ was used to initialize the values and to compare with exact values. The resulting EOC in all three cases was about 1.55.

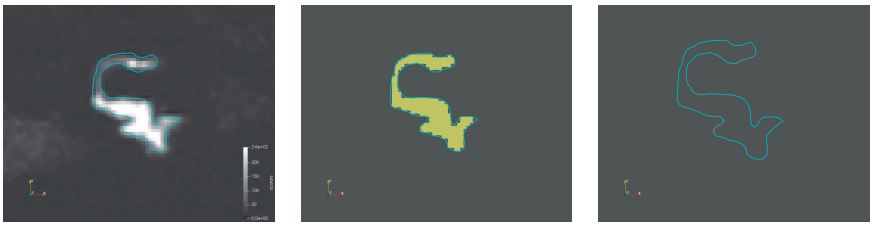


FIGURE 7. Left: the initial data and the isoline resulting from semi-implicit scheme. Middle: thresholding in Paraview and isoline of the semi-implicit method. Right: isolines of both semi-implicit and explicit (red) schemes, for this example almost identical.

(in Paraview) isolines can be displayed. Due to interpolation of the data, the displayed noise is not as strong as without it. Inverted images are often used to show the noise better, as, for exmaple, in Fig. 2.

When the results are used in the next step of the processing chain of biological experiments, the thresholded data are used. In Paraview, we can use thresholding, which is displayed in yellow, sometimes together with the isoline of the explicit or implicit schemes to compare. In Fig. 7 (c), two isolines corresponding to explicit and implicit schemes are displayed together to compare.

The problems encountered during segmentation are typically of the following types:

(1) Macrophages contain very thin parts, often exhibiting weak intensities (Fig. 8).
(2) Objects become fragmented after Niblack thresholding (Fig. 9).
(3) The background near the object contains noise with intensities similar to those within the object (Fig. 10).

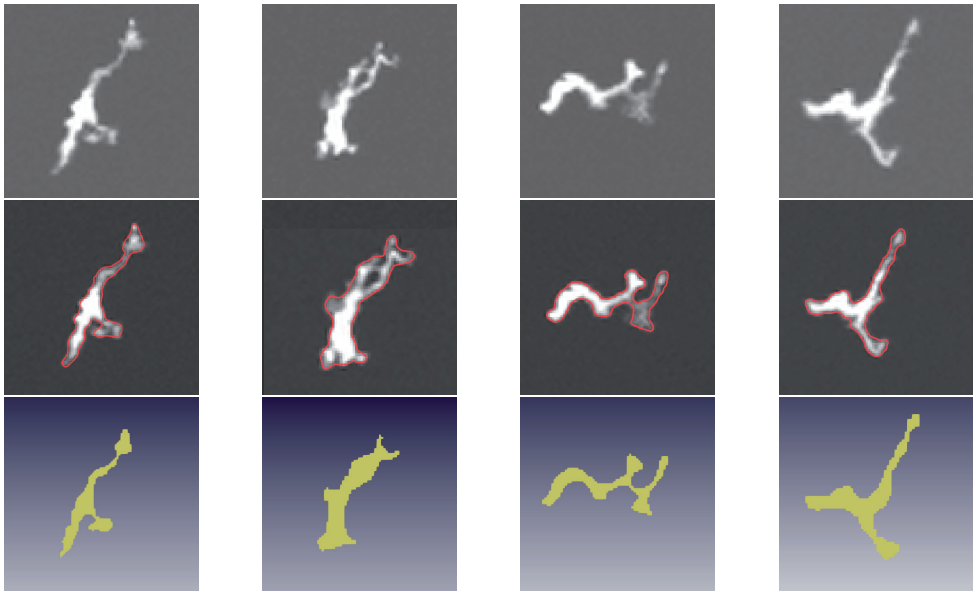
(4) Object parts are significantly separated by dark intensity and are distant — this method cannot process them without modification. Of the 76 objects in this data set, there were 5 macrophages which were split into two parts after segmentation. Object parts are significantly separated by dark intensity and are distant. This method cannot process them without modification. For example, of the 76 objects in this data set, 5 macrophages were split into two parts after segmentation.

When the results of the segmentation are used for applications such as cell tracking, it is crucial that the isolines are segmented as a single continuous piece. Multiple disconnected segments corresponding to a single macrophage complicate cell tracking and the evaluation of morphological characteristics (see subsections 5.1–5.3). To obtain a more precise boundary of the macrophage (e.g., to estimate morfological characteristics or to use them as data for machine learning), we use a different parameter setting. Such examples will be shown in Subsection 5.4.

The aforementioned problems will be discussed in the following subsections. Subsections 5.1–5.3 will use the same parameter setting for all macrophages in the data set.

5.1. 0bjects having thin parts with week intensities

As shown in Fig 8, the selected macrophages display thin parts characterized by weak intensities. To prevent fragmentation, initial thresholds used larger time values. It is important for the initial segmentation function (the second threshold) to cover the entire macrophage. Additionally, the first threshold, used for computing g_p^0, should fit the object as closely as possible — meaning it shouldn't be too "wide" and if fragmentation does occur, the distances between fragments should be minimal. In these experiments, the first threshold used 60 time steps of the explicit scheme for the linear heat equation with time step 0.2, the second threshold needed 90 time steps. The explicit scheme for SUBSURF needed 500 time steps with $\tau = 0.1$, $K = 0.0015$ and $\epsilon = 0.01$. The values of g_p^0 are evaluated from the first threshold — binary image with values 0 and 255. For $K = 0.0015$ we have low values of g_p^0 for nonzero $|u_p^0|$ (on the edges of the first threshold), on the other hand, in constant regions $|u_p^0| = 0$, $g_p^0(|\nabla u_p^0|) = 1$. Elevated vector field values exist only at macrophage boundaries. Thus, the segmentation function is attracted only if it is sufficiently close; otherwise, curvature motion prevail. This explains observations in some concave parts, where the segmentation function's isolines could not extend inward because the curvature was too strong; see Fig. 8 b) and d) and Fig. 8 c) and d). The holes in the object were filled (Fig. 8 b). In all experiments, the value used for isolines was always 130, and the threshold value used in Paraview was 115–255.



(a) cropT27 (b) cropT30 (c) cropT34 (d) cropT35

FIGURE 8. Macrophages containing thin parts with low signal intensity.

5.2. Objects thresholded as fragmented

In this experiment, we applied the semi-implicit scheme using the same parameter settings, with $\tau = 1$. We performed 50 time steps (corresponding to the same total time) and the tolerance of the linear system (referring to the convergence of residuals) was set to 0.05.

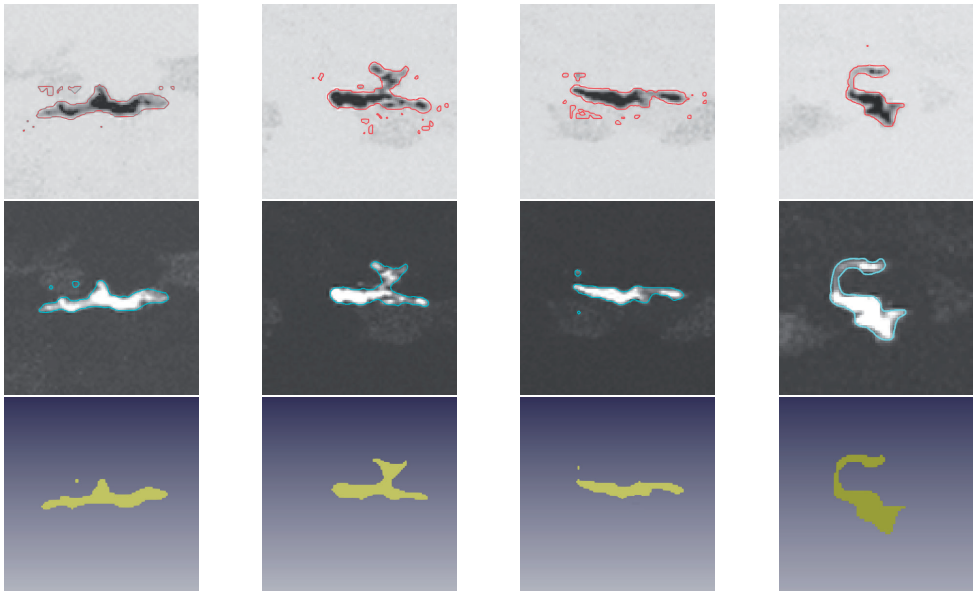
Figure 9 presents the results for selected macrophages. The second row displays the first thresholds, which are notably fragmented. However, SUBSURF successfully connected these separated parts, provided that they were not too distant, for reasons consistent with those discussed in the previous subsection.

5.3. Objects with noise

Figure 10 displays the results for noisy images using this parameter setting. The noise was either removed or partially removed during the grid construction (see the end of Section 3.3. Because this redundant noise is present in both the first threshold and the segmentation function (as the maximum of the first and second thresholds), its removal is slow. Although small dots could be eliminated during post-processing or segmentation function construction, this version does not include such a step.

(a) cropT53 (b) cropT54 (c) cropT60 (d) cropT64

FIGURE 9. Objects with missing parts and fragmented Threshold 1.



(a) cropT0 (b) cropT2 (c) cropT3 (d) cropT7

FIGURE 10. Suppression of the noise. The isolines of the inverted initial segmentation function and results shown as isolines and thresholds.

5.4. Getting isolines with better accuracy

Our aim is to obtain isolines that more accurately describe macrophage boundaries. The results for selected macrophages are presented in Fig 11. We modify the parameters as follows: first, the thresholds, particularly the first, are now closer to the object (20 and 40 time steps of the linear heat equation (LHE) with a 0.2 time step). Further, the first threshold is smoothed by LHE (typically at T=1). The smoothed vector field $-\nabla g(|\nabla I^0_\sigma|)$ that attracts isolines to the boundary is wider, and its greatest strength is now closer to the center of the boundary edge. Finally, by enlarging K, we increase the stopping diffusion coefficients, thereby reducing the influence of curvature motion at the boundaries. Looking at Fig. 11 we can see that the isolines can reach concave parts. For detailed information, see Fig. 12, which focuses on the macrophage labeled cropT66. The initial segmentation function is now located within the concave region. Although the isoline exhibits high curvature, the vector field is sufficiently close to attract and maintain it.

However, some macrophages previously segmented as a single piece may now be fragmented. The needed processing time is longer, because the motion of isolines is slower, and more noise can appear now at the beginning, due to shorter times of LHE for the thresholds.

For this experiment, both explicit and semi-implicit schemes have been tested with following setting of parameters:

for thresholds 20 and 40 iterations of LHE with

$$\tau = 0.2, \quad K = 0.02, \quad T = 160,$$

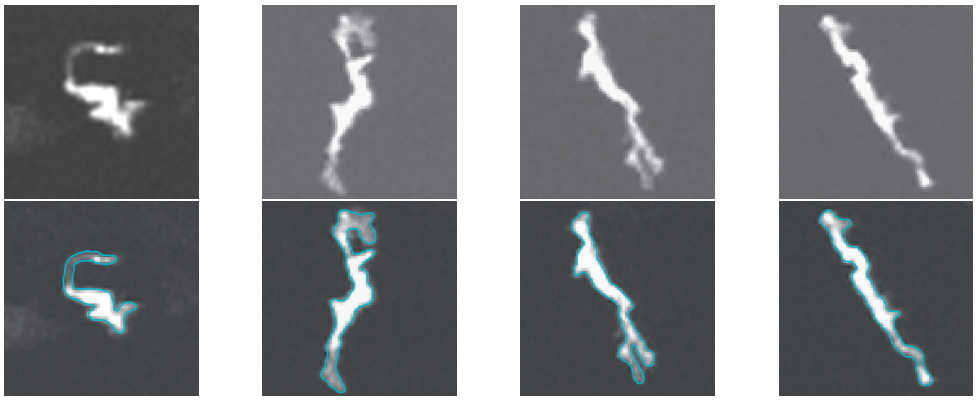
presmoothing of threshold 1 with

$$T = 1, \ \ \epsilon = 0.01,$$

for the linear system

$$\text{tol} = 0.05.$$

For this experiment, we also compared the CPU times of the explicit and semi-implicit schemes (for solving linear systems and time-step iterations). For the semi-implicit scheme, the initial number of iterations decreases rapidly, and at the end, eventually requiring only a few iterations. We did not observe significant differences in times (for cropT2, 1.96s for semi-implicit and 2.14s for the explicit schemes). Increasing τ resulted in more iterations and the times were again similar - but precision for higher time steps was lower. The times to run *p4est_iterate callback* function depend on the number of elements, so, in general, many factors influence the total CPU time. However, if there are more iterations of the linear system, such as for experiments with the first setting of parameters, the explicit scheme was (for selected macrophages with a similar number of grid elements) 3–5 times faster.

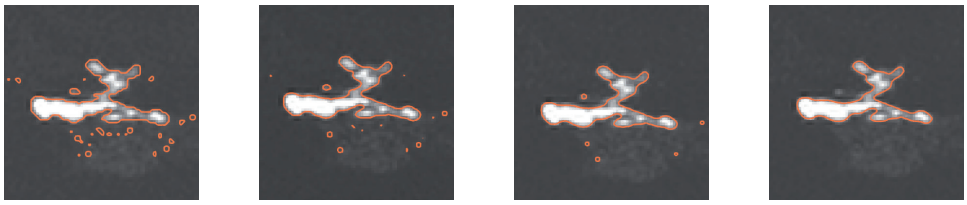


(a) cropT7 (b) cropT50 (c) cropT66 (d) cropT70

FIGURE 11. For selected macrophages, more accurate isolines are shown, concave parts are outlined successfully.

(a) threshold1 (b) threshold2 (c) cropT27 (d) final isoline

FIGURE 12. CropT66 — (a)smoothed input for g_p^0, (b) the initial segmentation function, (c) the segmentation function after evolution, (d) the result of the segmentation.



(a) cropT2, t=0 (b) t=40 (c) t=80 (d) t=160

FIGURE 13. Removing the noise. Because of shorter time for thresholds, there is more noise in the input. Evolution of cropT2 by the explicit scheme and isolines with the value 130 time t.

6. Conclusion

The aim of the paper is to present novel numerical schemes for the modified SUBSURF method, adapted to the environment and the built-in characteristics and tools, which the *p4est* library provides. We have derived the semi-implicit and explicit numerical schemes and demonstrated their effectiveness on groups of macrophages with selected characteristics. In this paper, our focus was on fragmentation. Processing all crops of macrophages in the explored data set has shown that there were only five macrophages for which we were unable to connect fragments that are too far apart. However, the paper works only with the basic method and does not incorporate statistical characteristics as, e.g., in ([13]). The presented numerical schemes can be implemented within the application [8], which supports rectangular data and offers built-in parallelism. The git repository contains sources of the library, source code of the application, implementation details and files with instructions on how to set up the developer environment of the *p4est* library, and many useful links.

Acknowledgement. The author thanks Tamara Sipka, Mai Nguyen-Chi, and Georges Lutfalla at the LPHI Laboratory of Pathogen Host Interaction, CNRS, University of Montpellier, for providing the macrophage data set. This work has been supported by grants APVV-19–0460, APVV-23–0186, VEGA 1/0249/24, and the European Union's Horizon 2020 under EXCELLENT SCIENCE — Marie Sklodowska-Curie Actions with grant agreement ID 955576. Also I would like to thank Angela Handlovičová and Juraj Kraňanský for precious help.

REFERENCES

[1] BURSTEDDE, C.—WILCOX, L. C.—GHATTAS, O.: *p4est: scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, SIAM J. Sci. Comput. **33** (2011), no. 3, 1103–1133.

[2] BURSTEDDE, C.: *Parallel tree algorithms for AMR and non-standard data access*, ACM Trans. Math. Software **46** (2020), no. 4, art. no. 32, 1–31.

[3] EYMARD, R.—GALLOUËT, T.—HERBIN, R.: *A finite volume for anisotropic diffusion problems*, C. R. Math. Acad. Sci. Paris **339** (2004), 299–302.

[4] EYMARD, R.—HANDLOVIČOVÁ A.—MIKULA, K.: *Study of a finite volume scheme for the regularized mean curvature flow level set equation* IMA J. Numer. Anal. **31** (2011), no. 3, 813–846.

[5] ISAAC, T.—BURSTEDDE, C.—WILCOX, L. C.—GHATTAS, O., : *Recursive algorithms for distributed forests of octrees*, SIAM J. Sci. Comput. **37** (2015), no. 5, 497–531.

[6] KHURSHID, K.—SIDDIQI, I.—FAURE, C.—VINCENT, N.: *Comparison of Niblack inspired binarization methods for ancient documents.* In: Proc. Document Recognition and Retrieval XVI. Vol. 7247 SPIE, 2009, pp. 267–275.

[7] KRASŇANSKÝ, J.: *Segmentácia Biologických Dát na Adaptívnych Sieťach, Diploma Thesis* (2021), pp 61.

[8] KRASŇANSKÝ, J.: *Repository to the Diploma Thesis* (2021), `https://bitbucket.org/Ligazetom/p4estthesis/src/master/`.

[9] KRIVÁ, Z.—MIKULA, K.: *Adaptive diamond cell finite volume method in image processing, Proceedings of contributed papers*, ALGORITMY (2009) (18th Conference on Scientific Computing, Vysoké Tatry-Podbanské, Slovakia, March 15–20, 2009), Publishing House of the Slovak University of Technology in Bratislava, (2009), pp. 121–133.

[10] KRIVÁ, Z.—HANDLOVIČOVÁ, A.: *Evaluation of gradient norms on a consistent quadtree grid in 2D*, Information Technology and Computational Physics **1** (2017), 143–164, DOI: 10.1007/978-3-319-44260-0_9.

[11] KRIVÁ, Z.: *Finite-volume level set method and its adaptive version in completing subjective contours* Kybernetika **43**(2007), no. 4, 509–522.

[12] OTSU, N.: *A threshold selection method from gray-level histograms*, IEEE transactions on systems, man, and cybernetics **9** (1979), no. 1, 62–66.

[13] PARK, A. S.—MIKULA, K.: *Automated completion of segmented fragments of macrophages using weighted dilation and erosion in 2D+time microscopy videos*, Proceedings of ALGORITMY (2024) (the 22nd Conference on Scientific Computing, Slovakia, 2024), pp. 11–22.

[14] PARK, S. A.—SIPKA, T.—KRIVÁ, Z.—AMBROZ, M.—KOLLÁR, M.— BALÁZS, K.––NGUYEN-CHI, M.— LUTFALLA, G.—MIKULA, K.: *Macrophage image segmentation by thresholding and subjective surface method*, Tatra Mt. Math. Publ. **75** (2020), 103–120.

[15] PARK, A. S.—SIPKA, T.—KRIVA, Z.—LUTFALLA, G,— NGUYEN-CHI, M.— MIKULA, K.: *Segmentation-based tracking of macrophages in 2D plus time microscopy movies inside a living animal*, Comput Biol Med. **153** (2023), DOI: 10.1016/j.compbiomed.2022.106499.

[16] SANKUR, B.—SEZGIN, M.: *Survey over image thresholding techniques and quantitative performance evaluation*, J. Electronic Imaging **13** (2004), 328–374. https://doi.org/10.1117/1.1631315

[17] SARTI, A.—MALLADI, R.—SETHIAN, J. A.: *Subjective surfaces: A method for completing missing boundaries*, Proc. Natl. Acad. Sci. USA **97** (2000), no. 12, pp. 6258–6263.

[18] SOMOROVSKÁ, M.—KRIVÁ, Z.: *Image segmentation of macrophages based on a local adaptive thresholding and a subjective surface method*, in: Advances in Architectural, Civil and Environmental Engineering. Spektrum STU, Bratislava 2022, pp. 36-42.

[19] WYNN, T. A.—CHAWLA, A.—POLLARD, J. W.: *Macrophage biology in development, homeostasis and disease*, Nature **496** (2013), 445–455; https://doi.org/10.1038/nature12034

[20] https://en.wikipedia.org/wiki/Space-filling_curve

[21] https://zenodo.org/record/3267228#.YBf1Xy2z1QI

Received August 8, 2024
Revised February 24, 2025
Accepted April 23, 2025
Publ. online October, 20, 2025

Department of Mathematics and Constructive Geometry
Faculty of Civil Engineering
Slovak University of Technology
Radlinského 11
SK 810 05 Bratislava
SLOVAKIA

E-mail: zuzana.kriva@stuba.sk