hochschule mannheim

Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment

Thorsten Mueller

Bachelor Thesis

for the acquisition of the academic degree Bachelor of Science (B.Sc.)

Course of Studies: Computer Science

Department of Computer Science

University of Applied Sciences Mannheim

11.11.2015

Tutors

Prof. Dr. Markus Gumbel, Hochschule Mannheim

Erika Mustermann, Paukenschlag GmbH

Mueller, Thorsten:
Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment / Thorsten Mueller. –
Bachelor Thesis, Mannheim: University of Applied Sciences Mannheim, 2018. 50 pages.

Mueller, Thorsten:
Einsatz eines Flux-Kompensators für Zeitreisen mit einer maximalen Höchstgeschwindigkeit von WARP 7 / Thorsten Mueller. –
Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2018. 50 Seiten.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 11.11.2015

Thorsten Mueller

Abstract

Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment

The bladder is one of many organs in humans and animals. It is coated by the urothelium. The urothelium ensures that the bladder contains only urine. With bladder cancer the urothelium is not able to ensures its barrier function.
Bladder cancer is one of the most common cancer types in men. Cancer is able to spread into the sourundings organs and muscles. In the case of bladder cancer, the cancer arises in the urothelium and spreads then into the neighboring organs as it grows. If bladder cancer spreads into the sourunding organs and muscles the affected people have almost no chance of healing. Therefore it is important to recognize bladder cancer early. To do so insights on how it arises are required. To receive these insights several simulations in two dimensions were done. With these simulation first insights of the rise of bladder cancer are revealed.
Because the urothelium is a three dimensional product, the simulation should now be extended by a third dimension. With this extension it is hoped to receive new insights on how bladder cancer arises.

Contents

Chapter 1

Introduction

1.1 Motivation

One of several requirements regarding complex life is cell adhesion. If cells do not stick to each other the only living things would be cells. In humans and animals, organs and epithels are made of several cells and several layers of cells. This is also the case for the urothelium, which is an epithelium, i.e. a membranous tissue which consists of one or several layers. For the urothelium it is necessary that the cells stick to each other. Otherwise the functions of the urothelium could not be executed and also it would not be able to grow.

There are two types of tumors. One is the benign and the other one is the malignant tumor [1]. The benign tumor is self limited. Thus, it does not invade surrounding tissues nor does it spread into other body parts [1]. The malignant tumor on the other hand is not limited in its growth and is able to invade other body parts [1].

Since bladder cancer is one of the most common cancer types among men it is important to understand how and why the cancer is able to grow.
Bladder cancer starts to grow in the urothelium. With its grow and spread, the structure of cells sticking together changes. In this case, the urothelium is no longer able to completely perform its tasks. In order to understand the urothelium, how and when bladder cancer appears, it is necessary to observe the epithel.

To understand the functionalities of organs and epithels, in general organisms, observations are essential. For the urothelium this is already done, as there are a lot of different in vitro experiments about the methodology of the urothelium. After an observation of an epithel or organ is complete, researches are able to predict how

the observed organism will react in different situations. To verify these predictions a simulation is necessary. A simulation is an abstract illustration of the reality, but it can also be used to change reality in a for the research specific way, to get more knowledge of the epithel, or an organism in general.

A simulation should always be as simple as possible but also not too simple. Otherwise the simulation does not represent the reality. There are several programs with different algorithms for cell simulation. A popular algorithm is the Glazier-Graner-Hogeweg (GGH) model. This model is popular because it is easy to describe how cells interact with each other and it is possible to define constraints for the volume and surface of each cell.
The program CompuCell3D (CC3D) is a simulation program, which uses the GGH algorithm in its simulation. In the moduro project CC3D is used, and with the program the GGH algorithm is used.

The target of the moduro project is to predict under which circumstances bladder cancer occurs and when it is able to grow. Therefore, 16 different morphogenesis models of the urothelium were created. An overview of these models is displayed in table 2.1 at page 12. So far, all 16 models were simulated in 2D for 720 days. The results reveal that some of the models are more realistic than others.

Because a cell is a three dimensional organism, a 2D simulation of the urothelium might not give as many aspects as a 3D simulation could do. Thus, the aim of this bachelor thesis is to create a 3D simulation of these 16 different models. With this 3D simulation it is hoped to receive new insights into the urothelium and how bladder cancer occurs.

1.2 Background

1.2.1 Biology of the Urothelium

Bladder cancer is the 4th most common cancer type in men according to everydayhealth.com [2], every 36st out of 100.000 men gets it. Bladder cancer usually starts with some cells in the bladder growing uncontrolled. From these cells, the tumor can spread further into surrounding areas [3]. The most common bladder cancer type is the urothelial carcinoma [3].

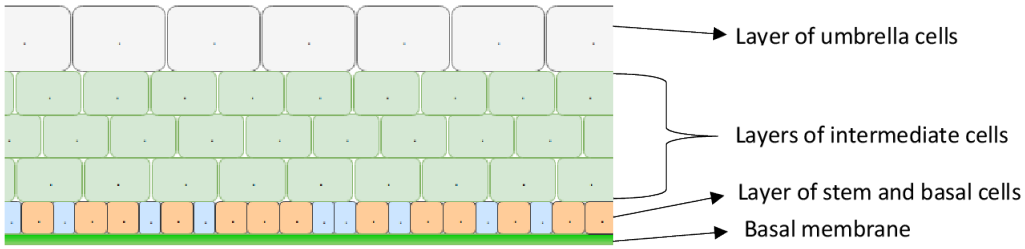


Figure 1.1: A simplified illustration of the urothelium. At the very bottom is the basal membrane. Above the membrane the stem and basal cells are displayed. The blue cells represent the stem cells, the red cells display the basal cells. Above the layer out of these two cell types, the urothelium contains several layers of intermediate cells. At the top of the urothelium is a layer of umbrella cells.

The bladder is located in the lower urinary tract and consists of several parts. The urothelium is one part of it and coats the bladder [4]. More specifically, it covers the bladder from the renal pelvis to the proximal urethra [5], [6].

Two important tasks of the bladder are the storage and release of urine. To do so the bladder will extend, during the storage, and then shrink again [7]. One task of the urothelium is to form a distensible barrier [4], [8]–[11], which prevents unregulated exchange of ions, solutes, and toxic metabolites between the bladder and the blood [4], [8]–[10]. The urothelium ensures its barrier function, through enlargement and downsizing of its size . This is done by the largest cells of the urothelium, the umbrella cells. Since the umbrella cells are in direct contact with the bladder, it is their task to change size and form during the growth and shrink process of the bladder. Birder described the urothelium as "... a responsive structure capable of detecting physiological and chemical stimuli and releasing a number of signaling mole-cules." [6]. Another task of the urothelium is to control the movement and passage of macromolecules, ions, water, toxic metabolites and solutes [8], [9]. If the urothelium is damaged, it rapidly generates new cells, to ensure full functionality [5], [8], [9].

To receive a better overview the different cell types are explained in the following paragraph. In figure 1.1 at page 3 a simplified illustration of the urothelium with its different cell types and cell layers is provided.
The umbrella cells, also called superficial cells, are connected directly with the bladder and have an average diameter of 25 up to $250\,\mu\mathrm{m}$ [5], [9].

Below these cells the intermediate cells are located. With an average diameter of 10 up to $20\,\mu\mathrm{m}$ [5], [9], they are smaller than the umbrella cells. There are at least three and up to five layers of the intermediate cells [9].

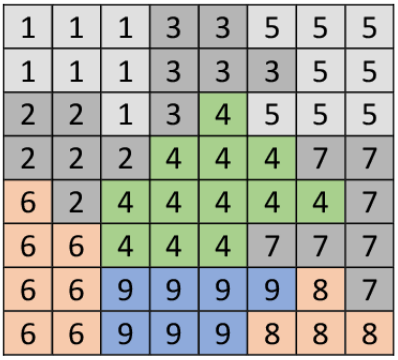


Figure 1.2: A square lattice in 2D. The same digits represent one cell $(\sigma(\vec{i}))$, whereas the different colors represent different cell types $\tau(\sigma)$.

The smallest and the most common cells in the urothelium are the basal and stem cells. Those cells have a diameter of up to $10\,\mu\mathrm{m}$ [4], [9].

The urothelium consits of several layers. In the first layer, there are the basal and stem cells. Above them, there are several layers of intermediate cells. On top of the epithelium is one layer of umbrella cells.

1.2.2 CompuCell3D

CC3D is an open-source program, which provides a simulation environment for multi- or single-cell-based modeling of tissues, organs and organisms [12]. To do so, CC3D uses the GGH model in its simulation. CC3D provides the possibility to create programs for the simulation, e.g. cell growth, mitosis, apoptosis or necrosis scripts, in python, C++ or in CC3DML, which is their own Markup Language. With such programs CC3D allows the user to modify the behavior of the simulation for a specific purpose. CC3D uses the GGH approach, explained in section 1.2.3 at page 5. It allows the user to choose between two cell-lattice types, i.e. a presentation of the pixels or voxels of a cell at a specific position in the simulation field. By default it uses a square-lattice of single pixels for each dimension, an example therefor is displayed in figure 1.2 at page 4. CC3D provides the possibility to use a hexagonal-lattice, where the pixels are hexagons in two dimensions, or rhombic dodecahedrons in three dimensions. Since the core of a GGH simulation is the effective energy [13],

CC3D tries to minimize this effective energy every Monte Carlo Step (MCS), i.e. a calculation step in the simulation. The basic form for the effective energy is:

$$\mathcal{H}_{boundary} = \sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})), (\tau(\sigma(\vec{j})))(1 - \delta(\sigma(\vec{i}), (\sigma(\vec{j}))) \tag{1.1}$$

This equation is a part of the equation of the GGH model. The formula of the GGH model, and with it this equation, is explained at section 1.2.3 at page 5.
There are two ways to extend this form, it is possible to either add a volume or a surface constraint for each cell. During each MCS an index-copy attempt takes place [13]. In this index-copy attempt a pixel is selected, and it is tried to overwrite a randomly chosen pixel, next to the current pixel. It succeeds and takes place if this index copy attempt decreases the effective energy [13]. Each MCS the program tries to minimize the effective energy with index copy attempts.

1.2.3 Glazier Graner Hogeweg Model

Glazier and Grander developed several formulas and models which are presented in this subsection with focus on the Cellular Potts Model (CPM) and GGH model.
The GGH model is widely used in biological simulations, since it provides a good flexibility, extensibility and usability [14]. Glazier and Graner developed the Extended Potts Model (EPM) as an extension of the large-q Potts model, which itself is an extension of the Ising Model. Nowadays, this model is called CPM [14]–[16]. Glazier and Graner extended the CPM in a way that also volume constraints are considered for the hamiltonian, see following form:

$$\begin{aligned}\mathcal{H}_{CPM} = &\sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})), (\tau(\sigma(\vec{j})))(1 - \delta(\sigma(\vec{i}), (\sigma(\vec{j}))) \\ &+ \sum_{\sigma} \lambda_{vol}((\tau)v(\sigma) - V_{target}(\tau(\sigma)))^2\end{aligned} \tag{1.2}$$

The hamiltonian of equation 1.2 describes the effective energy for the extension of the CPM model. The first sum describes J of all cells, the adhesion energy between different cells. Therefor, every cell has a specific cell type $\tau(\sigma)$ [15], [16]. Each cell is placed onto a lattice with a spin $(\sigma(\vec{i}))$ for every given dimension [14], [15]. The adhesion energy between cells is only considered if the kroenecker delta is 0. Thus, the surface energy between cells is considered if $\delta(\sigma, \sigma') = 0$ [14]–[19].

With the second sum over all cells the volume of each cell is now considered within the effective energy. The user is now able to set a target volume $V_{target}(\tau(\sigma))$ for each cell and a multiplier λ_{vol} for the deviation between the current volume $(\tau)v(\sigma)$ and the target volume. During the simulation this deviation is tried to be kept as small as possible for every cell in order to keep the effective energy as small as possible.

Together with Hogeweg Glazier and Graner further developed the created extension of the CPM. The further developed model is called GGH model. The main extension is that the user is now able to add surface area constraints [14]–[16] as well as to use a negative boundary energy [14]. With the surface constraint the equation for the effective energy of the GGH model is:

$$\begin{aligned}\mathcal{H}_{GGH} = &\sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})),(\tau(\sigma(\vec{j})))(1-\delta(\sigma(\vec{i}),(\sigma(\vec{j})))\\ &+\sum_{\sigma}\lambda_{vol}(\tau)v(\sigma)-V_{target}(\tau(\sigma)))^2\\ &+\sum_{\sigma}\lambda_{sur}(\tau)s(\sigma)-S_{target}(\tau(\sigma)))^2\end{aligned} \tag{1.3}$$

In addition to the hamiltonian of equation 1.2 is the surface constraint. It has the same principle as the volume constraint. Thus, the user is able to define a target surface $S_{target}(\tau(\sigma))$ for each cell and a multiplier λ_{sur} for the deviation between the target and the actual surface $(\tau)s(\sigma)$ of each cell. Since the volume and surface constraint are included in the effective energy, it should be possible to use these two parts of the effective energy to shape the cells.

Beside the surface constraint the new model allows the user to model (a): cell growth and proliferation (b): mitosis, i.e. cell division (c): fields, forces and diffusion and (d): chemotaxis and haptotaxis [14].
Glazier et. al. describe their model as:

> "GGH models define biological structure consisting of the configuration of a set of *generalized cells*, each represented on a *cell lattice* as a domain of lattice sites sharing the same cell index [...], a set of *internal cell states* for each cell [...], and a set of *auxiliary fields* ..." [14].

"Initial conditions emulating a particular biological configuration rather than random initial conditions." [14] brings the advantage that the GGH model now has now biologically motivated properties instead of physically motivated properties [14].

1.3 Objective

The aim of this bachelor thesis is to create a 3D morphogenesis simulation of the urothelium using CC3D. Because the simulation models and the program for a 2D simulation are given and the simulation is done by CC3D, the task is to modify the current application, of the 2D simulation, in a way that this program can be used for a 3D simulation.
The required changes are all in the program, because the models of the 2D simulation should be also used for the 3D simulation. Therefore, some parts of the program have to be modified whereas other parts have to be developed. Because a third dimension will be added to the simulation, it is possible to try to let the cell have a different shape than in the 2D simulation. The result of this bachelor thesis will be presented with an analysis of an 3D simulation. A model therefor is selected out of the models which were created earlier in the project.

1.4 Outline

In this chapter the necessary knowledge to understand this bachelor thesis is provided. Chapter 'State of the Art' provides the status at which the project was at the beginning of this bachelor thesis. Once the basic knowledge and the 'State of the Art' are explained, the modifications in the program during the journey are revealed in chapter 'Methods'. Then, in chapter 'Results' the outcome of this thesis is presented. After the results are presented, these are discussed of different perspectives in chapter 'Discussion'. In chapter 'Future Work' ideas for the project are presented. To round up this paper a conclusion is provided in chapter 'Conclusion'.

Chapter 2

State of the Art

In this chapter all information about the current project are revealed. At the beginning the properties of the project are explained and later in this chapter the simulation program is presented. The sections 2.5 to 2.9 are explained based on the program of the project, wheras the other sections are presented based on previous work of the project.

2.1 Moduro

In the project moduro stands for 'Modeling of the Urothelium with the GGH approach'.
The department of Medical Informatics at the University of Applied Sciences Mannheim and the Clinic of Urology in cooperation with the Medical Faculty Mannheim at the University of Heidelberg participate in this project. The aim is to predict how and when bladder cancer arises.

The current moduro project has a stable 2D simulation of 16 different models using CC3D. The simulations are performed by CC3D. During the simulation statistics about the current simulation are written in text files and can be read out by the moduro toolbox.
The project consists of a program and several models, both are written in python. The models include properties of cell behavior. Therefore, the adhesion energy between cells and the possibilities of the new cell types after mitosis is defined in the different models. The application modifies the cell behavior, for instace it checks when a mitosis takes place and how fast the cell will growth.

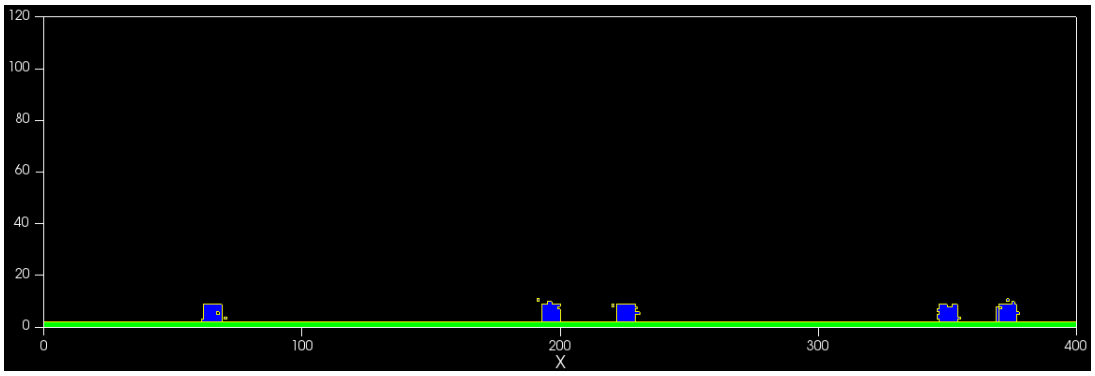


Figure 2.1: Initial state of a 2D simulation of the model SPA/BPCD/IPCD

2.2 Display and Simulation of the Urothelium

In the 2D simulation, the urothel is simulated with a size of $500\,\mu m$ for the x-axis and $150\,\mu m$ for the y-axis. Because, the voxel density is 0.8, the size of the urothelium is represented with 400 pixels at the x-axis and 120 pixels at the y-axis. The voxel denstity describes how many pixels are used to display $1\,\mu m$. CC3D allows the user to use any simulation size, as long as the required hardware is able to handle the simulation.

It is possible to use CC3D on several cores of the CPU. In the project one core for a simulation is used, because otherwise CC3D splits the simulation field into different grids. As a result race conditions can occur at the edges of these grids [13]. This means it is possible that a cell is in both grids. Thus, one half is in one grid and the other half is in another grid. During the simulation both grids calculate the volume of the cell and both apply the new volume of their calculation but it is not checked which result has to be used. Therefore, in order to reduce wrong results, it is necessary to avoid race conditions.

The 2D simulations of the urothelium covered 720 days. In the simulations one day is split up into 500 MCS. Therefore, the simulation covers 360000 MCS. The simulation duration can be set in the program. Thus, the user defines how many MCS simulate one day and how long the simulation runs. At the first calculation step, MCS 0, the simulation is initialized, i.e. the cells are drawn and placed on the basal membrane. An illustration of an initial simulation is displayed in figure 2.1. Since morphogenesis is simulated, the urothelium is proposed to growth and to proliferate in the given area. An illustration therefor is presented in figure 2.2.

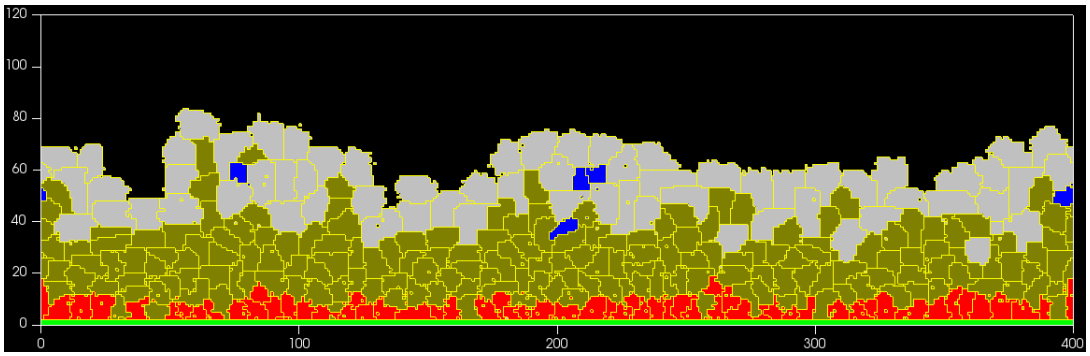


Figure 2.2: 2D Simulation of the model SPA/BPCD/IPCD after 33 days

2.3 Moduro Toolbox

The moduro toolbox is a software tool, which is able to visualize data in order to evaluate a simulation.
Every 12 hours the simulation is checked for its reality. This is done with the fitness functions of section 2.8. The results of these calculations are written in several text files, in a by CC3D created directory of the simulation. Moreover, the amount of cells of the different cell types and the overall amount of cells is saved as well. The moduro toolbox is able to read out the data of the text files and then visualize these. This visualization is done with charts and with a table.
CC3D saves some screenshots, of the simulation field, during the simulation. With these screenshots the moduro toolbox is able to create a video out of these screenshots. To create the video an extra program has to be installed which the toolbox would use.

2.4 Models

The 2D simulation of the urothel provides 16 different models. These models differ mostly in which cell types after the mitosis are created. The project divides the models into two domains, one has the identifier 'SSD' and means that every time a stem cell divides there will be one stem and one basal cell. The second domain is called 'SPA'. In the second domain the stem cell has a probability of 90% that it will be one stem and one basal cells after mitosis. There is also the chance with a probability of 5% that after mitosis of a stem cell there are two stem cells or two basal cells.
For the mitosis of basal cells there are 4 different models. The mitosis model 'BSD'

describes that each basal cell which undergoes mitosis will create one basal and one intermediate cell. The model 'BPA' describes that there is a 5% chance that the basal cell will become two basal or two intermediate cells. There is a 90% probability that the cell become one basal and one intermediate cell. The model 'BPCD', describes that always a basal cell becomes two basal cells during mitosis and if the basal cell is not on the basal membrane it transform into an intermediate cell. The model 'BCD' has the behavior that a basal cell immediately transforms into an intermediate cell if it is not at the basal membrane.
There are two different models how intermediate cells split. One is the 'IPCD', therefor in mitosis an intermediate cell becomes two intermediate cells during. Each MCS it is checked if there is a cell around the intermediate cell. If this is not the case it is transformed into an umbrella cell. The second model is the 'ICD'. In this one, only transformation of the intermediate cells into the umbrella cells happens if the intermediate cell is not enclosed by other cells.
With these mitosis concepts there are 16 different models in the project. These models were created by an earlier version of the project [20] and are displayed in table 2.1 at page 12.

2.5 Adhesion

During morphogenesis the cells not only growth, they also sort themselves. For cell sorting there have to be different adhesion values, i.e. how strong two different cells are holding on each other. In the project this is done with a matrix, which every of the 16 model has. Such a matrix was created in an earlier version of the project [20] and is displayed in table 2.2 at page 13.

2.6 Cell properties

In the project the physiology constraints of a cell are included. In an earlier version of the project these were evidenced [20] and are displayed in table 2.3 at page 13. In the program these constraints are converted into pixels and then applied.
In the simulation every cell has several attributes. Moreover, each cell has an cell dictionary, in which additional attributes are stored. The properties regarding the cell type are likely what every cell in general has, e.g. a min- max diameter, a min-

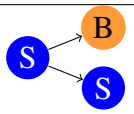
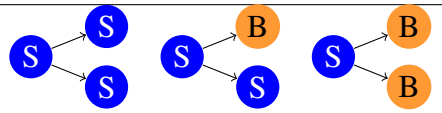
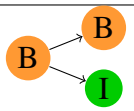
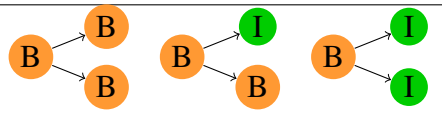
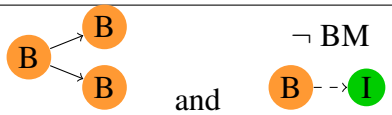
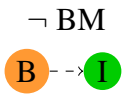
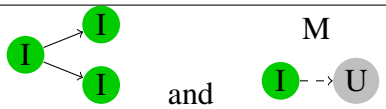
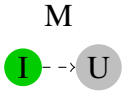
Type	ID	Description	Model
Stem cells	SSD	Stem cell-like division	
	SPA	Stem cell population asymmetry	 $p_s = 0.05$ $p_a = 0.90$ $p_s = 0.05$
Basal cells	BSD	Stem cell-like division in basal cell	
	BPA	Basal cell population asymmetry	 $p_s = 0.05$ $p_a = 0.90$ $p_s = 0.05$
	BPCD	Proliferation and contact differentiation of basal cells	
	BCD	Only contact differentiation of basal cells	
Intermediate cells	IPCD	Proliferation and contact differentiation of intermediate cells	
	ICD	Only contact differentiation of intermediate cells	

Table 2.1: 16 different models derived in the project as there are different ways of proliferation and mitosis are simulated for the different cell types.

Types		M	BM	S	B	I	U
Medium	M	0	14	14	14	14	4
Basal membrane	BM		-1	1	3	12	12
Stem cell	S			6	4	8	14
Basal cell	B				5	8	12
Intermediate cell	I					6	4
Umbrella cell	U						2

Table 2.2: Adhesion matrix for a model in the simulation. Smaller values refer to more adhesion and higher values mean less adhesion. The cell type M is the medium cell type, it is by CC3D a specific cell type which is every in the available space in the simulation, and BM is the basal membrane.

Cell type	V_{min}	d_{min}	V_{max}	d_{max}	Volume	Surface
Stem S	268	8	523	10	perfect	average
Basal B	381	9	523	10	important	average
Intermediate I	905	12	1767	15	important	poor
Umbrella U	1767	15	3591	19	important	poor

Table 2.3: Constraints of a cell. Volumes V in μm^3, diameters d in μm. The 'Volume' and 'Surface' column describe how the λ_{vol} and the λ_{sur} should be set for each cell type.

max volume, growth in µm per day or the time until apoptosis, i.e. cell death.
The attributes in the cell dictionary are mainly used for decision making. Some values are the current and expected live time, a flag for necrosis and if a cell is inhibited, i.e. enclosed by other cells.

2.7 Events in the simulation

In the simulation program there are several events modeled. These events can occur every calculation step or at a specific MCS in the simulation.
The events which are performed every MCS, except for MCS 0 and every MCS of factor 250, are a) cell growth and b) the check for mitosis c) cell death d) cell transformation and e) cell mutation. The event which occurs every 12 hours, every 250 MCS, is f) urination. These events are presented in detail in the following subsections.

2.7.1 Cell Growth

In the project the maximum possible growth of a cell is calculated and applied. This calculation is used for the relation volume and target volume as well as for the relation volume, surface and target surface. Because, the volume and the surface of a cell is calculated by CC3D and can only be read out, the target volume and target surface are calculated in the program and set.

2.7.2 Mitosis

The verification if a cell divides or growth further is done in every simulation step, except for these with a factor of 250. Therefore, it is possible to define a very specific calculation step at which a cell divides, as there a 500 MCS per day. In order that a cell divides it grows over its maximal size before it divides. The cells which are able to grow and as a result divide are a) stem cells b) basal cells and c) intermediate cells. The umbrella cells grow as well, but they do not split.

2.7.3 Necrosis

If a cell dies, necrosis takes place. In the program a flag in the cell dictionary is set. Every calculation step, except for the MCS of factor 250, the program checks if a cell dies or not. If so, the cell will shrink and as a result disappear.

2.7.4 Mutation

After 2 days, 1000MCS, it is possible that a cell mutates, i.e. it becomes malignant. We simulate the possibility for cells to mutate after 2 days, because otherwise there would not be much cells around the mutated cell. Each cell type has its own probability to become malignant. In the current simulation it is not considered that cells mutate since the probability for each cell type to mutate is 0%.
If a cell mutates, there is also the flag for necrosis set. Thus, the cell shrinks and disappears.

2.7.5 Transformation

A transformation can take place if a basal cell divides and at least one intermediate cell is created. If the intermediate cell is not enclosed by other cells, it immediately will be transformed to an umbrella cell.

2.7.6 Urination

Every 12 hours, every 500 MCS, a urination takes place. The first urination event takes place at MCS 375. This is simulated in a way that randomly 2% of the cells in direct contact with the bladder are washed out. In the program a flag for necrosis is set and the cells will disappear because of the necrosis event.

2.8 Fitness functions

In order to validate the simulated models, there are several functionalities which check if the model is realistic or not. These functions, i.e. part of a program, check every 12 hours, every 250 MCS, if the model is realistic or not. The result of these fitness functions is written into several files, and can be read out by the moduro toolbox.

2.8.1 Arrangement fitness function

The arrangement fitness function ensures that the strata of the simulated urothelium has the correct order [20], i.e. that the first layer on the basal membrane consists only of stem and basal cells **REFS** the next three to five layers consists only of intermediate cells **REFS** and that there is one layer of umbrella cells **REFS**

$$lib = \frac{L - lib}{L} \tag{2.1}$$

$$f_a = 1 - \frac{((1 - L_B) + (1 - L_U) + lib + (1 - L_O)}{4} \tag{2.2}$$

In equation 2.1 L refers to the amount of layers in the urothelium. lib describe the amount of layers between the stem and basal cell layer and the umbrella cell layer

at the top of the urotehlium, which include not only intermediate cells. Thus, if the layers between the first and last layer consist only of intermediate cells, the equation 2.1 has the result 0.
In equation 2.2 L_B and L_U are boolean values, i.e. they have the value 0 or 1 and represent a false or true value. They are 1 if the first layer of cells consists only of basal or stem cells and if the most upper layer consists only of umbrella cells, otherwise they will be 0. lib is calculated in equation 2.1. L_O presents the optimum amount of layers. It is 1, if the amount of layers in the simulated urothelium is between 3 and 7, otherwise it is 0. The result of equation 2.2 is between 0 and 1, where 0 refers to no reality in the simulation and 1 refers to a perfect simulated urothelium.

2.8.2 Volume fitness function

This function calculates the relative volume regarding the current volume of the different cell types in the urothel. The relative amount of the different cell types should be: stem and basal cell = 10%, intermediate cells = 67% and umbrella cells = 23% considering an average thickness of $85\,\mu\mathrm{m}$ [20]. Therefor the formula is:

$$f_{V_i} = \frac{1}{4(\frac{V_{Si} - V_{Ii}}{V_{Si}})^2 + 1} \tag{2.3}$$

V_{Si} and V_{Ii} describes the *should* and the actual *is* volume of a specific cell type i [20]. This calculation is done three times. One time for the stem and basal cells, one time for the intermediate cells and one time for the umbrella cells. The results of the three calculations are then further used to determine the relative volume overall.

$$f_V = \frac{f_{V_B} + f_{V_I} + f_{V_U}}{3} \tag{2.4}$$

In this equation f_{V_B} refers to the realtive volume of the stem and basal cells, f_{V_I} presents the relative volume of the intermediate cells and f_{V_U} include the realtive volume of the umbrella cells.
The result of all four calculations are written into a text file and can be read out by the moduro toolbox later.

2.8.3 Overall fitness function

The overall fitness function calculates the total fitness out of the volume and the arrangement fitness function. Therefor the average of both functions is calculated [20]. This calculation is done by the moduro toolbox only. For every calculation step, the arrangement and volume fitness function are calculated the overall fitness is calculated. Therefor the formula is:

$$f(t_i) = \frac{f_V(t_i) + f_a(t_i)}{2} \tag{2.5}$$

t_i describes a specific time point, in MCS, at which this calculation is done. At the end of the simulation the average of the overall fitness function is calculated to determine the reality of this simulation. Therefor, the formula is:

$$f = \frac{1}{e+1} + \sum_{i=0}^{e} f(t_i) \tag{2.6}$$

The result of the calculation is between 0 and 1. Where 0 describes no reality at all and 1 presents a perfect realistic simulated urothelium.

2.9 3D functionalaties

The functionalaties of the program are able to be used in 3D. Whenever a part of the program is required to be used in 2D as well as in 3D, it is checked if the simulation field has a third dimension or not. Examples for such functionalities are the fitness functions of the section before or the decision how many stem cells are placed on the basal membrane, which is modified and explained in section 3.4.
Because so far only the 2D simulations were progressed, it is important to keep the functionalaties of the 2D simulation in the program. Doing so it is possible to switch between a 3D and a 2D simulation, as well as the benefits of both simulation types can be observed.

Chapter 3

Methods

This section gives an overview over all changes in the project which were done during this bachelor thesis. These changes include some small improvements of the application as well as how to draw a sphere cell. First minor changes will be described briefly, further onwoards in this chapter the more complex changes will be explained.

3.1 Lambda multipliers

In the project the mulitplieres λ_{vol} and λ_{sur} are set if a cell is initialized and a second time in the necrosis event, if the program models that a cell dies.
In each cell object, there are two additional multipliers, one for the surface and one for the volume constraints. Because these values are not used in the program and they do not influence or set the λ_{vol} or λ_{sur} which CC3D uses, these two values are deleted. Moreover, in one place there were methods to calculate the lambda values. Because these methods are not used in the project and they had a multiplier itself to calculate the multiplier for the specific part of the effective energy, they are deleted as well.

In the project the multipliers λ_{vol} and λ_{sur} are now set each time when a cell is initialized. A second time the multiplier λ_{vol} is set, is if necrosis takes place. Now the unnecessary methods and values for the multipliers values are removed. Thus, no confusion about which constraint values in the program are used arises.

3.2 Abstract methods

The program has several classes with methods which were not used because of polymorphism. Polymorphism is a method out of Object Oriented Programming (OOP). Another technique out of OOP is the use of abstract classes and abstract methods. To explain polymorphism, abstract classes or abstract methods in detail, would take too much space out of this bachelor thesis. Thus, the change in the program is explained in the following.

In the project are several classes which inherit from each other. In these classes were the same methods implemented. It is not required that several classes have the same methods. Such methods are redundant methods, as long as they do not differ in their functionality. For these methods it is possible to declare them as an abstract method. An abstract method contains only the method construct, but no functionality. Due to polymorphism the class in which such an abstract method is implemented is used to call the method of the class in which the method is implemented, if one of the two class inherits from the other class.
For Python exits the library 'abc'. With this library it is possible to declare an abstract method. To do so every class with one or more abstract methods needs to initialize a special class variable. With this variable Python is able to recognize that there are abstract methods included.

```
class ModelConfig(object):
    __metaclass__ = ABCMeta
```

Listing 3.1: The initialization of a class variable which is required by Python in order to detect abstract methods and abstract classes.

The listing above displays how the class variable has to be initialized in order that abstract methods are recognized. If a class has at least one abstract method, it is an abstract class. In Python abstract classes can include implemented methods as well as abstract methods.

```
    @abstractmethod
    def _createExecConfig(self):
        pass
```

Listing 3.2: Declaration of an abstract method.

The listing above is an example for an abstract method in Python. Using abstract methods creates more structure and clarity within the project. Therefore, the abstract declarations in this project created a much better clarity of the structure of the project and of the methods, which were used in the classes.

3.3 Calculation steps until urination

To simulate the urination in the program it was checked if the current calculation step is larger than 250 and a factor of 125. Every 250 MCS the volume- and arrangement fitness functions are calculated and no other events in the simulation take place. Therefore, the urination was simulated every 12 hours. The check if the current MCS is a factor of 125 happens through the modulo operator. Thus, the current calculation step is divided by 125 and if the remainder of the result, as a whole number, is 0 it is a factor.
The check when the urination event takes place is modified. The urination event now takes place if the current calculation step is larger than 250 and if the current MCS modulo 125 equals 1. Therefore, the urination event is used every 6 hours, at MCS 376, 501, 626, 751, etc..

3.4 Amount of stem cells on the basal membrane

In an earlier version of the project it was evidenced that around 12% of the area of the basal membrane are required to be filled with stem cells in order to have an optimal proliferation during the morphogenesis of the cells [20]. In the project the calculation of the amount of stem cells for two dimensions were correct but without an mathematical evidence.
Since the y-axis is negligible in this calculation of an area, the calculation for two dimensions considers the x-axis and the calculation for three dimensions considers the x- and z-axis. Therefore, in two dimensions the area of the stem cells should be calculated by using the cell diameter. An illustration therefor is displayed in figure 3.1. For a 3D simulation it is possible to calculate the area of stem cells with a circle with the formula $A = \pi \cdot r^2$, because the y-axis is not considered in this calculation, and use this calculation to further determine the amount of stem cells on the basal membrane. An example for the basal membrane and a stem cell in three dimensions is displayed in figure 3.2.

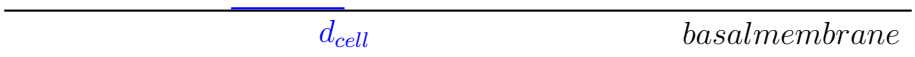


Figure 3.1: Considered area to spread the stem cells in 2D with an example of one stem cell placed on the basal membrane. Because only the x-axis is displayed we need to calculate the diameter of a cell.

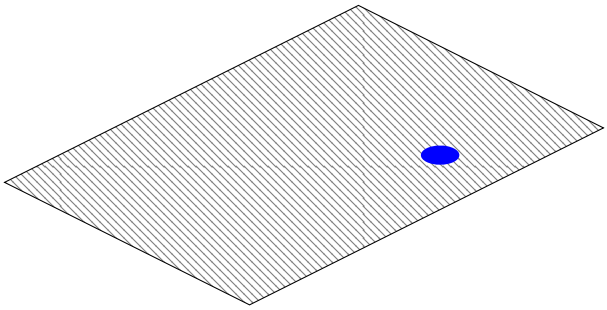


Figure 3.2: Considered area to spread the stem cells in 3D with an example of one stem cell placed on the basal membrane. The hatched area displays the basal membrane and the circle represents one stem cell. This stem cell is a circle, because in the calculation of the amount of stem cells the y-axis is negligible.

Table 3.1: Possible approximation error by not rounding the result of equation 3.2. The first column describes the length of the basal membrane, the values are in $\mu\mathrm{m}$. In the second column the result of equation 3.2 is displayed. In the third column the result of the second column is rounded up in the first row. In the second row this result is casted. The fourth column displays how much space the stem cells, in $\mu\mathrm{m}$, of the rounded or casted result of eqataion 3.2 require. The last column displays the physical space required in percentage to the basal membrane.

xLength of the basal membrane	Result of equation 3.2	Amount of stem cells	Area used of stem cells in µm	relative used area
200	~ 2.66	3	27	13.5%
200	~ 2.66	2	18	9%

With the following two equations it is possible to calculate the amount of stem cells in two dimensions. $A_{stemcells}$ refers to the area which can be used for stem cells. Therefore, it is 12% of the basal membrane. C is a constant, which describes 12% of an object. Thus, $c = 0.12$.

$$A_{stemcells} = xLength \cdot c \tag{3.1}$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{d_{cell}} \tag{3.2}$$

Equation 3.1 ensures that only 12% of the given area is used. Formula 3.2 then calculates the amount of stem cells on this given area. Because the result of the calculation is often not a whole number, the result is checked if the first decimal digit is larger or equal than 5 and then it is rounded up or casted, i.e. the decimal digits are cut off.

As table 3.2 displays, it is important to round the result. Otherwise there would be an approximation error and as a result a calculation error.
For three dimensions the equation 3.1 has to be extended. An illustration therefor is figure 3.2. Equation 3.2 has to be modified, because the area of an circle, instead of the diameter, is used for the calculation. Therefore, equations 3.3 and 3.4 are used to calculate the amount of stem in three dimensions are:

$$A_{stemcells} = (xLength \cdot zLength) \cdot c \tag{3.3}$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{\pi \cdot r^2} \tag{3.4}$$

The result of formula 3.4 has to be rounded as well, otherwise the program would include rounding errors. These calculations are now included in the program.

3.5 Adhesion Matrix

To have a realistic simulation it is necessary to have a correct adhesion matrix. The matrix which was used for the 2D simulation is displayed in 2.2 at page 13. It might be that the adhesion values are useful for the 2D simulation but not for the 3D simulations. Therefore, obsevations of different adhesion values between different cell types are done and explained in the following.
To find the correct adhesion values, the simulation program was modified in a way that only two cells in a small simulation field are place next to each other. Then it was observed how these cells interact with each other with different adhesion values during the growth process.

3.6 Target volume and target surface after mitosis

Mitosis of the cell is simulated by CC3D. In the simulation mitosis is simulated as the following: one cell splits into two cells. The cell which splits dies and then two new cells out of the dead cell are created.
In the program we specify and check if and when a cell splits. CC3D decides where the cell splits and calculates the volume and surface of the two new cells. In the program attributes of the new cells, e.g. the target volume or the target surface, are calculate and set.

The target volume is calculated by dividing the target volume of the cell before mitosis by 2. This value is applied for both new created cells. The problem with this technique is that it is possible that the cell might not split in the middle. In the case of mitosis, setting the target volume of the two created cells without the knowledge of the volume is a source of error. Because the target volume of the two new cells was set without considering the current volume of the cell, it occurred that one of the new cells, after mitosis, had a target volume which was smaller than the current volume.
After mitosis both cells are initialized. Thus, the target volume and target surface are calculated and set. After the initialization of the cells is done, the target volume of the new cells is set to the target volume, of the cell before mitosis, divided by 2.
Because the target volume is calculated out of the volume of the cell during the initialization process, the calculation of the target volume is removed out of the mitosis event. Now the target volume and surface after mitosis is set only during the initialization process of the new cells.
In the program, the target volume and target surface after mitosis is calculated and set dependent on the of CC3D given volume.

3.7 Approximation Error

The project includes conversions from μm into voxels. In CC3D the amount of pixels, e.g. for the surface of a cell, has to be set as data type integer, i.e. a whole number. Therefore, it is possible that in some places in the program there are calculation errors during a conversion of the units. In the project, the values of unit μm are saved with the data type float, i.e. a number with decimal digits. To set these values as a whole number the values are casted into the data type integer.

Whenever a conversion from μm into voxels is done the complete calculation is calculated with decimal digits. After the calculation is done it is verified if the first decimal digit is larger or equal to 5. If this condition is true the result is increased by 1, otherwise not. As a last step the result is casted. This technique has the advantage that calculation errors due to casting are removed, because the cast is the very last step. It is possible that the program still includes some rounding errors, but these can not be removed because CC3D requires the amount of voxels as a whole number and the calculations are done with decimal digits.

In the function displayed below, the cast and the rounding of the result are done in the last step of the calculation.

```
def calcVoxelVolumeFromVolume(self, volume):
    r = (3 * volume / (4.0 * PI)) ** (1.0 / 3.0) # Radius of a sphere with
        known volume.
    rDimension = r * self.voxelDensity
    if self.dimensions == 2:
        return int(self.__truncate(PI * (rDimension ** 2))) # Area of a circle.
    else:
        result = 4.0 / 3.0 * PI * (rDimension ** 3)
        if result % 1.0 >= 0.5:
            result += 1

        return int(result)
```

Listing 3.3: Function to calculate the volume of a sphere in voxels out of a given physical volume. First out of the given physical volume the radius is calculated. Then it is converted into the voxel unit. Next the volume of the voxel sphere is calculated and as last step the result is rounded and casted.

One use of the displayed function is to calculate the voxel volume out of the physical volume of a cell. In the simulation a minimum and a maximum volume for each cell type is calculated. These values are used in the calculation to determine if mitosis takes place or not.
For the basal cell the minimum volume is $381\,\mu m$ and the maximal volume is $523\,\mu m$. In table 2.3 at page 13 the constraints of the different cell types in μm are displayed. In table 3.2 three different possible calculations of the conversion of a volume in μm into a volume in voxels are presented.

Table 3.2: Three different ways to calculate the voxel volume out of a given physical volume. The first column describes the physical volume in μm. In the second column the radius in μm out of the volume is calculated. Next, the radius is used as it is, casted or rounded up. In the fourth column the exact result of the volume in voxel is presented and in the last column the rounded result of the voxel volume is displayed.

Volume in μm	radius in μm	radius used in further calculation	not rounded result in vx	rounded result in vx
381	4.49	4.497	1285.67	1286
381	4.49	4	904.77	905
381	4.49	5	1767.15	1767

In the table, the first row calculates the voxel volume as it is done in project now. Thus, rounding and casting is the last step in the calculation. In the second row the radius is calculated and casted. Then this casted radius is used for the further

calculation. The third row calculates the radius and rounds it immediately. This rounded radius is then used for the further calculation.
As the table displays, there is a significant difference in all three results. Because such a calculation is used to determine when a cell splits as well as it calculates the growth per MCS, it influences the result of a simulation. Thus, it is important to round and cast the result as very last step in the calculation.

3.8 Draw Sphere Cells

Since a sphere as a cell is an approximation to a cell in the urothelium, it is a possible technique to draw and further simulate a cell as a sphere.
To be able to draw a sphere out of voxels it is required that a cuboid lays around the sphere, as it is displayed in 3.3. The cuboid has to be at least as large as $2 \cdot radius$ of the sphere. In addition, the cuboid should not be larger than necessary, otherwise there would be unnecessary computable cost. The illustration in figure 3.4 at page 27 display the perfect size of a square and a circle. These 2D objects are chosen to display the boundaries of a circle in a square as well as the boundaries would be in three dimensions.

To be able to draw a cell as a sphere, CC3D has to allow the user to draw several different voxels in the simulation field, all containing one cell. Since CC3D allows the user to draw several pixels containing one cell, a solution for this problem is possible.

Because we are using the square lattice, the cuboid is filled with voxels. To be able to calculate every point within the sphere, the cuboid and the sphere are required to have the same center **REF** In this case the equation 3.5 provides a mechanism in which every point within the sphere can be calculated.

$$\sqrt{(x_r - x_0)^2 + (y_r - y_0)^2 + (z_r - z_0)^2} <= radius \tag{3.5}$$

In the equation above x_r, y_r and z_r describe the current point of each of the three axis and x_0, y_0 and z_0 describe the center of the sphere. If the distance of the current x, y, z coordinate is smaller or equal to the radius the current point is within the sphere, otherwise it is outside of the sphere. Because a voxel itself contains several points, this equation, in this case with cuboids, has the weakness that only one point is considered. Thus, only one point of the voxel is considered in the

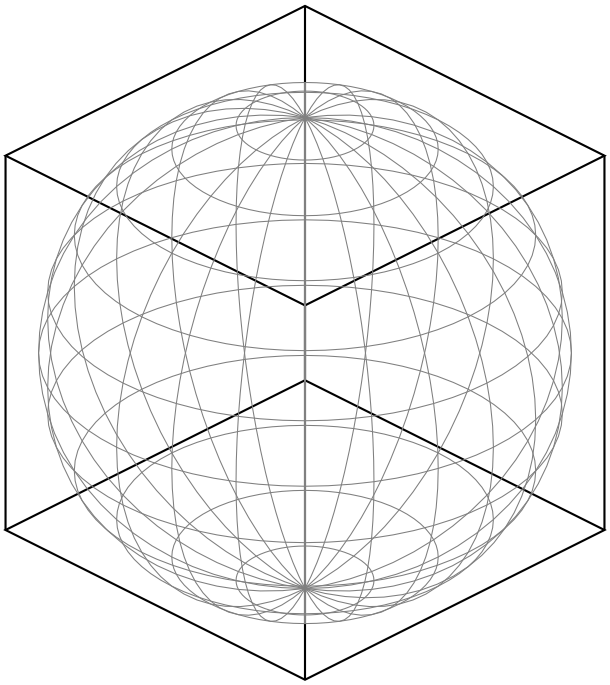


Figure 3.3: A cuboid layed around a sphere

decision whether the complete voxel is within the sphere or not. Since every voxel is a cuboid itself, the length of all corners are the same. This fact can be used to increase the accuracy of the equation above.

In the program it is possible to calculate the corner length of a voxel. Thus, it is possible to decide if the center of a voxel is at the inside or outside of a sphere. How the center of a voxel is calculated is explained in the following paragraph.
Since the radius of the sphere is known, whether it is known or it is calculated out of the volume or the area of a sphere, the diameter of the cuboid can be calculated. Since the voxel density is known in the simulation, the corner length of each voxel can be calculated as it is displayed in equation 3.6.

$$c = \frac{2 \cdot radius}{voxeldensity} \tag{3.6}$$

In this equation c describes the corner length of a voxel. With this corner length it is now possible to calculate the center of a voxel and use this center further to decide if the voxel is inside or outside of the sphere. To determine the center of a voxel the

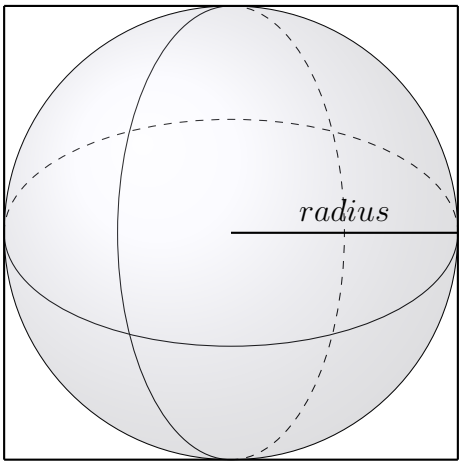


Figure 3.4: A cuboid with minimal size layed around a sphere

calculation has to be done for every of the three axes. The following three formulas in equation 3.7 display such calculations.

$$\begin{aligned} x_c &= x_r + \frac{x_{r+c} - x_r}{2} \\ y_c &= y_r + \frac{y_{r+c} - x_y}{2} \\ z_c &= z_r + \frac{z_{r+c} - x_z}{2} \end{aligned} \tag{3.7}$$

In this equation x_r, y_r and z_r describe the start point and x_{r+c}, y_{r+c} and z_{r+c} describe the end point of the voxel. With the calculations of equation 3.7 it is now possible to consider the center of a voxel in the decision if the voxel is in- or outside the sphere, as it is displayed in equation 3.8.

$$\sqrt{((x_c - x_0)^2 + (y_c - y_0)^2 + (z_c - z_0)^2} <= radius \tag{3.8}$$

To depict a cell as a sphere a new function had to be created. This function is named addSphereCell as it is displayed below. In the method all required points, the start- and end points as well as the centers, of all three axis are given in µm. Thus, these points as well as the radius, which is also given to the function, and the step-length, i.e. the corner length of a voxel, are converted into the voxel unit.
With all necessary points in the voxel unit it is now possible to iterate over all three axis and check for every voxel, if it is included in the sphere or not.

```
def _addSphereCell(self, typename, xPos, yPos, zPos, radius, steppable):
```

```
cell = steppable.newCell(typename)
xStart = self.execConfig.calcPixelFromMuMeter(xPos - radius)
x0 = self.execConfig.calcPixelFromMuMeter(xPos)
xEnd = self.execConfig.calcPixelFromMuMeter(xPos + radius)
yStart = self.execConfig.calcPixelFromMuMeter(yPos - radius)
y0 = self.execConfig.calcPixelFromMuMeter(yPos)
yEnd = self.execConfig.calcPixelFromMuMeter(yPos + radius)
zStart = self.execConfig.calcPixelFromMuMeter(zPos - radius)
z0 = self.execConfig.calcPixelFromMuMeter(zPos)
zEnd = self.execConfig.calcPixelFromMuMeter(zPos + radius)

radiusPx = self.execConfig.calcFloatPixel(radius)
stepLength = self.execConfig.calcFloatPixel(1)

# loop over the center of each pixel to determine boundaries of the circle
for xr in xrange(xStart, xEnd):
    for yr in xrange(yStart, yEnd):
        for zr in xrange(zStart, zEnd):
            rd = sqrt(
                ((xr+(((xr+stepLength) - xr)/2.)) - x0) ** 2 +
                ((yr+(((yr+stepLength) - yr)/2.)) - y0) ** 2 +
                ((zr+(((zr+stepLength) - zr)/2.)) - z0) ** 2)
            if (rd <= radiusPx):
                steppable.cellField[xr, yr, zr] = cell
```

Listing 3.4: Function to draw a cell as a sphere. First all required points for the calculation are converted into the voxel unit. Then over each axis is iterated. During these iterations for each voxel the distance to the center of the cuboid and sphere is calculated and then it is checked if the voxel is within the sphere or not. If the voxel is a part of the sphere it will be added to the sphere.

To draw the cell as a sphere is the first step to have sphere cells in the simulation. Two major parts of the simulation are the growth and the mitosis of the simulation. In order that the cells are able to stay as a sphere it is required to adjust the volume and surface calculation as well.

3.9 Growth of a sphere cell

The simulation of the morphogenesis of the urothelium requires the growth of cells. Ideally the cells keep their shape, which they become as they are depicted.
In the simulation the current volume and surface of a cell is calculated by CC3D. Every user is able to influence these values by setting the target volume and target surface and the proper multiplier, λ_{vol} and λ_{sur}, for the specific part of the effective energy. Because the goal of the simulation is to minimize the effective energy, the cell will change the current volume and surface in the direction of the set target volume and target surface.

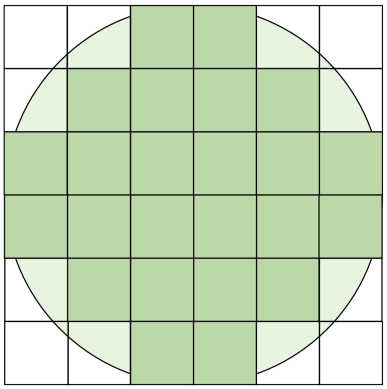


Figure 3.5: A circle in a possible pixel presentation. All small squares present pixels. The colored squares present the pixels, which are included in the pixel presentation of the circle.

To model the growth of a cell, the growth per day of the volume for the different cell types is set. Each calculation step the growth of one MCS is calculated and applied, 500 calculations steps are one day. After the growth of the volume is calculated and set as new target volume of the specific cell, a new target surface depending on the target volume is calculated and set.
In the program the target surface is calculated out of a real sphere and then multiplied by a factor. Since the surface of a sphere of voxels is larger than the surface of a real sphere, it is necessary to find the correct factor, by which the surface of a sphere has to be multiplied. This is necessary, in order to calculate the surface of a sphere of voxels.

For simplicity reasons, a first approximation of the factor is calculated in 2D. Thus, a circle and a square filled with pixels are used. An example therefor is displayed in figure 3.5.
Figure 3.6 displays an approximation, with which it is possible to calculate the factor for the surface.

In figure 3.6 the left square is considered as a pixel, it could be any pixel of the figure 3.5 through which the circle goes. Thus, the sides of the square are the same. The blue line represents the circle, in a way it could go through the pixel. An possible approximation is to insert a diagonal line. With this diagonal line an isosceles, right angular triangle is created as it is displayed at the right side of figure 3.6.
The following formula provides a way to calculate the sides of this triangle.

$$c^2 = 2 \cdot a^2 \tag{3.9}$$

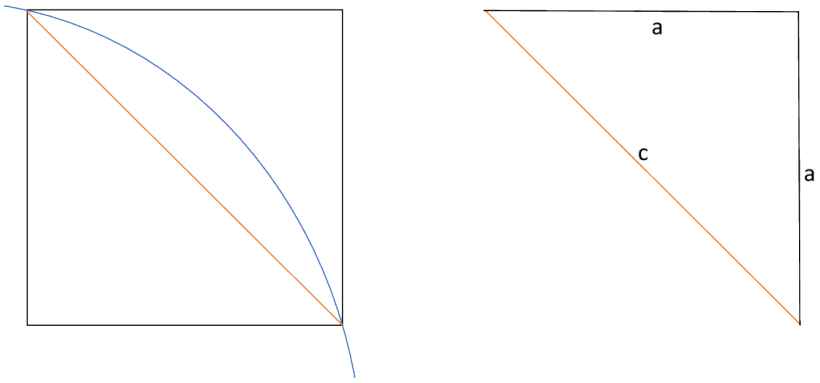


Figure 3.6: The left part of the illustration displays a pixel at the surface, in which the circle goes through. The blue line represents the circle. As an approximation to the surface line a diagonal line in the square is drawn. With this diagonal line an isosceles, right angular triangle appears and it is possible to calculate the additional surface of one pixel. This isosceles, right angular triangle is displayed at the right side of this figure.

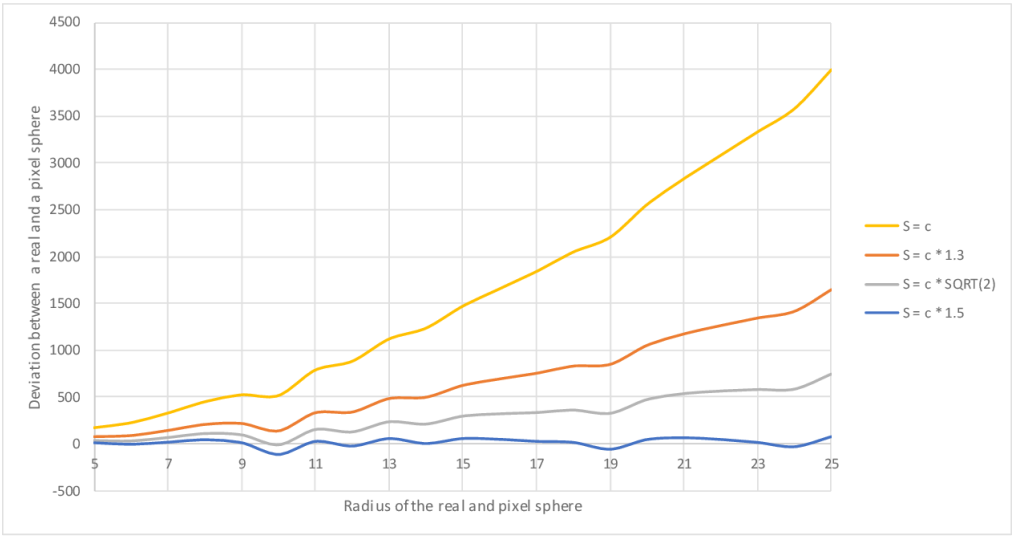


Figure 3.7: Deviation between the surface of the sphere of voxels and a real sphere. Each deviation is calculated with a different factor of the surface of the real sphere. S refers to the surface of the voxel sphere and c refers to the surface of a real sphere. Thus, $c = 4 \cdot \pi \cdot r^2$.
At the x-axis the radius of the sphere and the sphere of voxels is displayed. At the y-axis the deviation between both surfaces is represented.

This formula can be used to calculate the side a, as it is displayed in the following equation.

$$\begin{aligned} a &= \sqrt{\frac{c^2}{2}} \\ a &= \frac{c \cdot \sqrt{2}}{2} \end{aligned} \tag{3.10}$$

Since the surface of a pixel has two corner sides, the equation to calculate the surface of a pixel sphere is

$$\begin{aligned} S &= 2a \\ S &= c \cdot \sqrt{2} \end{aligned} \tag{3.11}$$

This formula applies to all pixels, which are at the surface of the pixel sphere, c is considered as the surface of the circle. Thus $c = \pi \cdot r^2$.

The calculated factor $\sqrt{2}$ was tested for a sphere surface. Because, this is an approximation to the surface of a pixel sphere, the tenth part beside this factor are also tested mathematically. As figure 3.7 displays the approximation with factor $\sqrt{2}$ still has some deviation. Almost no deviation between the surface of a sphere and a surface of a sphere of voxels appeared with a factor of 1.5. Thus, this factor is applied in the calculation of the target surface.

3.10 Calculation of the volume and surface sites of a voxel sphere

In order to determin the voxel volume and the surface sites of a voxels sphere without the start of a simulation, I created an algorithm for this problem. Otherwise a new simulation in CC3D had to be started for every measurement of the volume and surface of the sphere of voxels.
Since a sphere and a cuboid are both symmetrical, it is possible to split both into 8 pieces. Doing so only one eighth of the cuboid has to be calculated in order to determine the volume and the surface of the sphere.
To calculate which voxels are in the voxel sphere, the algorithm creates a x, z, y matrix. In the following calculations and decisions the center of each voxel is used. For every x,z voxel in the eighth of the cuboid the y constraint of the circle is calcu-

lated. Next, every voxel at the x,z coordinate is checked if the center of the voxel is within this constraint or not. Thus, every voxel at a x,z coordinate gets checked if its center is within the sphere or not. This process is repeated until every x,z coordinate of the eighth of the cuboid is calculated. Every time a voxel is within the sphere, it will be marked in the matrix.
To receive the volume of the voxel sphere all marked entries in the created matrix are count. Then, this result is mirrored for all three axes. Thus, the amount of marked voxels in the matrix is multiplied by 8, which equals to a single multiply by 2 for each axis. Therefore $V = ((Voxels \cdot 2) \cdot 2) \cdot 2$. To receive the surface sites of the voxels at the surface, every marked entry is checked if the voxel at x+1, z+1 or the y+1 is either out of bounce, i.e. if it is outside the cuboid, or outside the sphere. Every of the three conditions is checked for every voxel within the surface. If one condition is true, the amount of surface sites is increased by 1. At the end the result, like the volume, is multiplied by 8, in order to mirror the result for all three axis.

Chapter 4

Results

This chapter present the results of the bachelor thesis. Modifications which influence the structure of the program and the behavior of the simulation are presented at the beginning. Later the results of the desired cell as a sphere as well as the result of the created algorithm are revealed. In the last section evaluations of 3D simulations are shown.

4.1 Structure of the program

With the removement of unessesary methods and variables, which are not used, the program becomes more readable and is easier to understand. These changes decrease the probability of errors and confusion during code analysis, i.e. studying of the code. By declaring abstract methods the program becomes more structured. Because identical methods are not written again in classes which inherit from each other it is easier to understand the program as well to find errors.
The changes of section 3.1 and 3.2 modify the program in a way that it is more structured and understandable. This improves the time required for the code analysis, which is important for newcomers to the project, as well as to find errors.

4.2 Improvement of the program

During the bachelor thesis several adjustments in the program were made. All these changes are important for the simulation as they effect the simulation. These

changes were tested by using the output command to see the required information at the command line.
With the changes of section 3.4 the program now has a mathematical evidence in the calculation of the amount of stem cells on the basal membrane. This mathematical evidence has the advanges that changes in the calculation can be implemented fast as well as the part of the code becomes easier to analyze and understand. The new calculation is correct for every simulation size in 2D and 3D. This is important as the program should be able to have a correct implementation for two as well as for three dimensions.

Because the calculation until the urination event was modified in section 3.3 the urination takes place every 6 hours instead of every 12. This change is important for the simulation because with an urination every 12 hours the simulation is not as real as with the event every 6 hours. This change helps the simulation to not only be more realistic it is also important in order that there are not too many cells in the simulation field. As a result of too many cells in the simulation field the simulation stops. Thus, the change increases the reality and helps to prevent an overflow of cells.

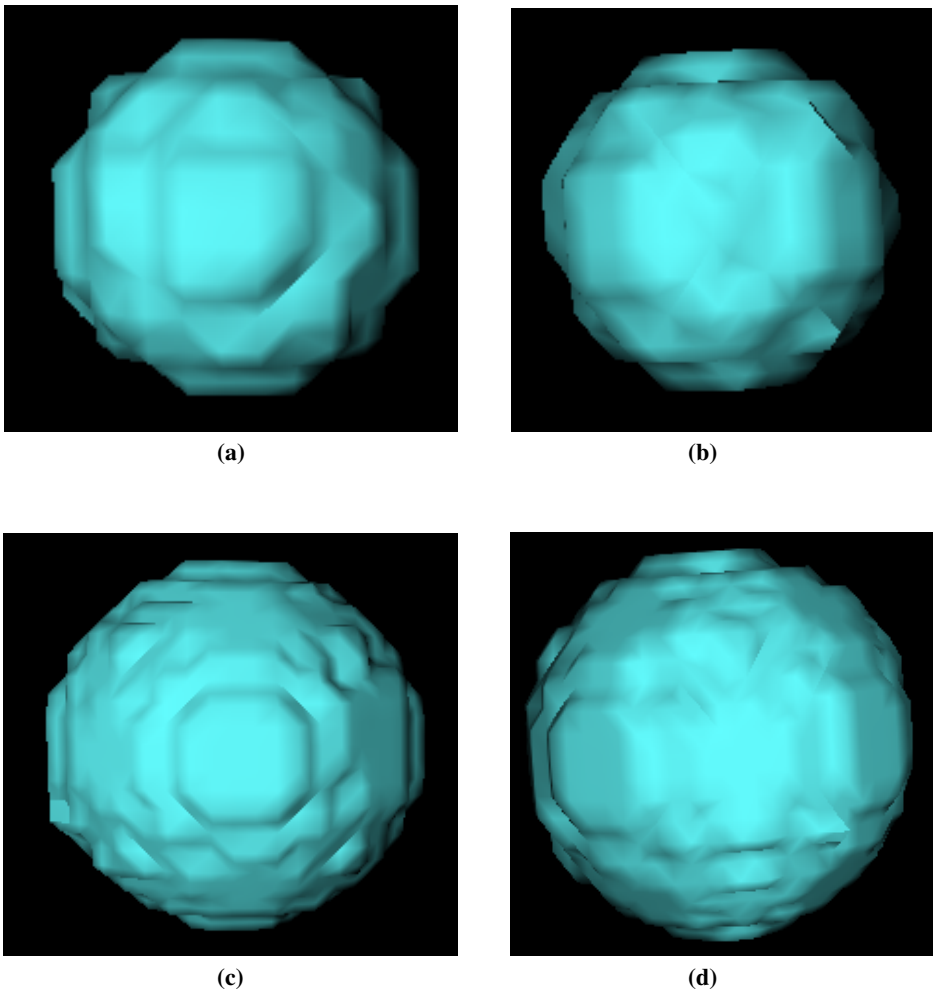
The modification of section 3.6 effects the simulation in a way that after mitosis both cells are able to grow immediately. Without the modification the cell with a too small target volume and target surface would not growth several calculation steps. Therefore, the cell does not grow several calculation steps until the target volume becomes larger than the current volume. Because the target volume of both created cells is now set dependent of the by CC3D given volume of each cell, it is not possible that one of the two cells has a smaller target volume and surface than the current volume and surface. With this modification both cells, created during mitosis, are now able to grow immediately.

The simulation is effected in a positive way because of the changes of section 3.7. The modifications in this section are crucial since now the calculation of volume constraints of each cell type is more precise. With this modification every convertion of $\mu\mathrm{m}$ into voxels is now free of calculation errors. Because the result of the calculations of the volume constraints of the different cell types is used to determine when mitosis takes place it is important to have them without an calculation error. Because the result is converted only at the end, simulations beside a voxel density of 1 have also the correct volume constraints of the cells. Moreover, this calculation is also used to calculate the growth per calculation step in the simulation, since

the growth of a cell is calculated in the physical unit and then converted into pixels. Because the calculation of the volume constraints of the different cell types and the calculation of the growth of each cell each MCS are two major parts of the simulation, this modification is important and effects the correctness of simulations.

4.3 Draw sphere cells

With the created method of section 3.8 the program is now able to draw a cell as sphere out of voxels, as it is displayed in figure 4.1 and 4.2.
Since voxels, cuboids, are used in the 3D simulation to draw a cell as a sphere, it is not possible to draw a perfectly round sphere. This problem is displayed with an circle and a square in picture 3.5 at page 29, since the circle and square are the 2D objects of an sphere and cuboid. The drawn cells are as spherish as possible in the simulation with the use of voxels.
Figure 4.1 displays two independent drawn cells with a radius of $5\,\mu\mathrm{m}$ and $9\,\mu\mathrm{m}$. These cells show that there are a lot of edges in the sphere. As the radius increases, the sphere shape of the cell gets more detailed, as it is displayed in figure 4.2. In this figure two cells are drawn idepently with a radius of $14\,\mu\mathrm{m}$ and $23\,\mu\mathrm{m}$. With the increase of the radius, the deviation of the surface increases as well, as it is shown in figure 3.7. This might be a result as the surface of a sphere and a cuboid, with the daimeter $2 \cdot r$, deviates more as r increases.



(a) **(b)**

(c) **(d)**

Figure 4.1: A single cell drawn into the simulation field. The radius of the cell of (a) and (b) is 5 and the radius of the cell of (c) and (d) is 9. Pictures (a) and (c) are with the front view, whereas the pictures (b) and (d) have an view angle of around 45 degree. The color of the cell is chosen in a way that more details are visible.

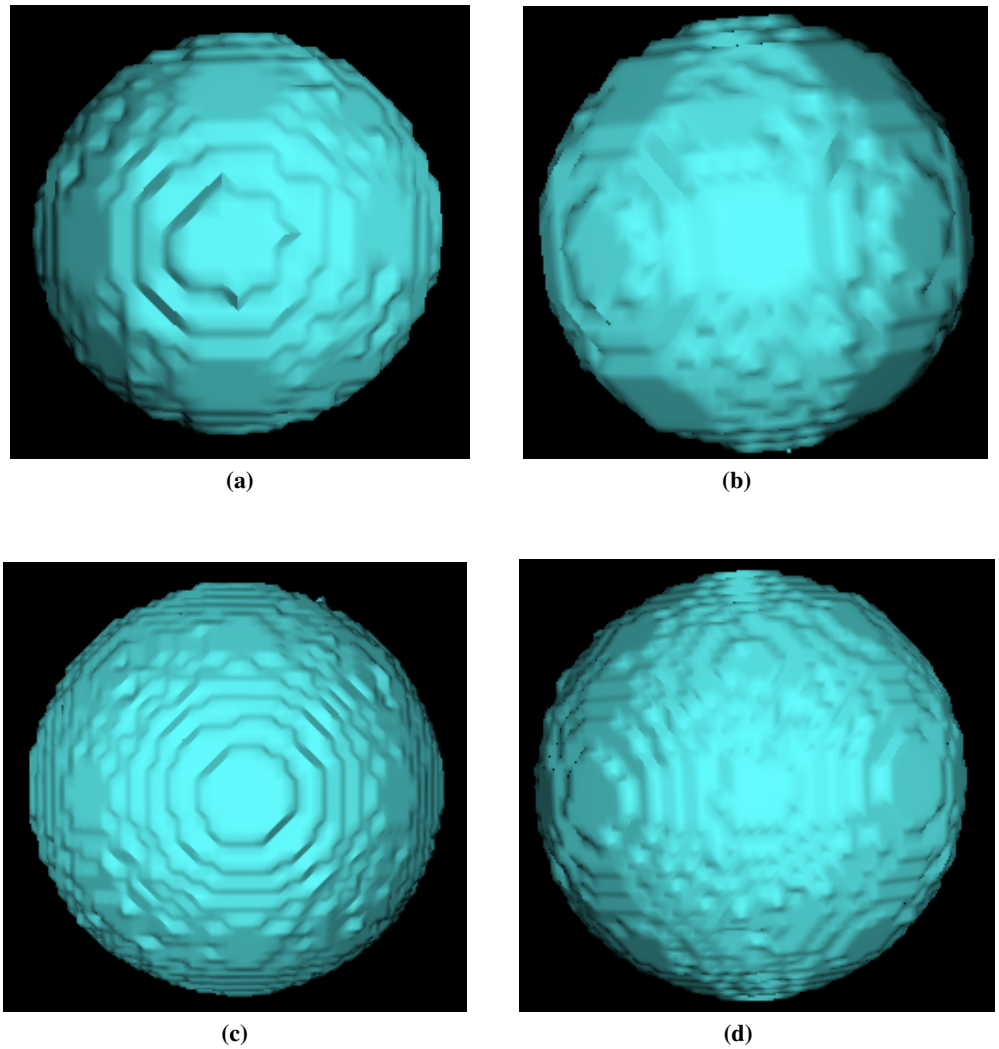


Figure 4.2: A single cell drawn into the simulation field. The radius of the cell of (a) and (b) is 14 and the radius of the cell of (c) and (d) is 23. Pictures (a) and (c) are with the front view, whereas the pictures (b) and (d) have an view angle of around 45 degree. The color of the cell is chosen in a way that more details are visible.

4.4 Calulate the voxel volume and surface sites of a sphere cell

The created algorithm of section 3.10 is able to calculate of a given radius the volume, of a sphere cell created with cuboids, in voxel as well as it is able to count the surface sites of the cell. Moreover, it is also possible to calculate the physical volume and surface of a sphere with a given radius. With this algorithm it is possible to calculate the volume and surface of a cell, drawn as a sphere of voxels, and become the same result as CC3D calculates.
The algorithm is useful but also has its weaknesses. For a voxel density beside 1 the results between the created algorithm and CC3D deviate. The results differ also if the radius is not a whole number. There are no insights how CC3D calculates the volume and surface of a drawn sphere cell. Moreover, for the tenths part of 5 to 9, for every radius with $.5$ until $.9$ it is possible to create the cube with an radius which is either rounded up or down. This means for a radius of 3.7 it is possible to use either 6 or 8 voxels to create the cuboid, as the diameter is best as $2 \cdot r$.
The algorithm helps to calculate the volume of a sphere cell, out of voxels, as well as to count the surface sites. It is able to be executed without an start of an simulaion. Thus, there is no need to adjust the coding of the project and to wait until the simulation started to recieve the volume and surface values of a cell drawn as a sphere out of voxels.

4.5 Grow sphere cells

In section 3.9 the factor for the calculation of the surface of the cell was evidenced best to be 1.5. With this factor it should be possible to let the sphere cell grow as a sphere. To test the growth of the cell, one single cell was placed in the simulation field. As the cell growed during the simulation the volume and surface values were rad out of the command line and compared to the documented values of a drawn cell.
To let a cell grow as a sphere does not work. Even the volume and surface values, calculated by CC3D, and the target volume and target surface values, calculated by the program, meet the values of drawn sphere cells with a small deviation *of an maximum deviation up to 50*.

Figures 4.3 and 4.4 display two examples of a single cell placed in the simulation field. It is observable that drawn sphere cells with a smaller radius become a cubish shape faster. In fiugre 4.3 the cell has a cubish form after 50 MCS where the cell in figure 4.4 becomes cubish after 750 MCS. This might be a result that the larger sphere cell has more detail than the cell with a smaller radius.

4.6 Simulations

Since no simulations are completed with three dimensions, several 3D simulations are presented. The models, which are simulated, are the most realistic models which were evidenced in an earlier version of the project [20]. The specific models are...

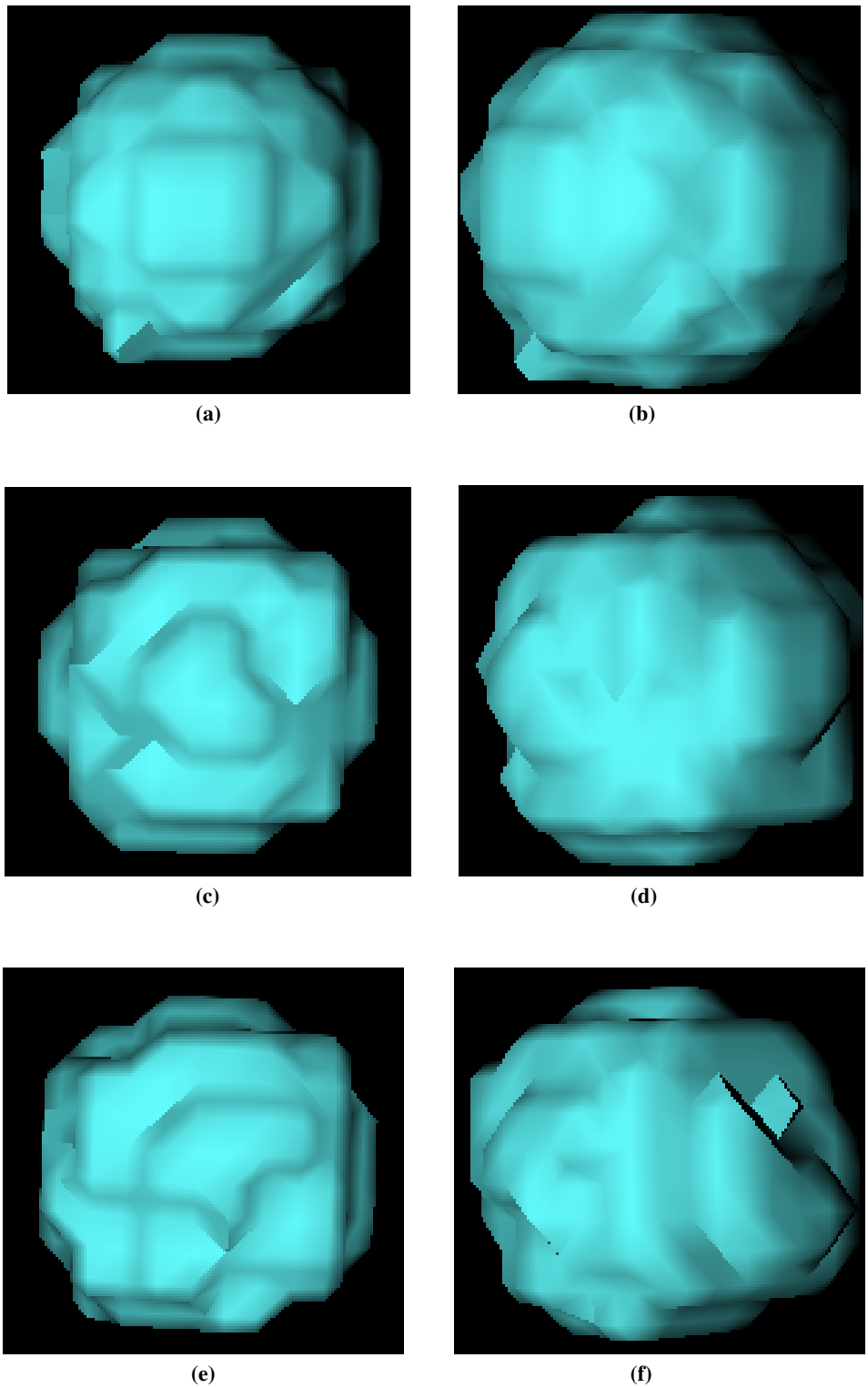


Figure 4.3: A sphere cell, with a radius of $5\,\mu\mathrm{m}$ and a voxel density of 1, as it grows. Images (a), (c) and (e) are the front view of the cell and figures (b), (d) and(f) have around a 45 degree angle of the front. Figure (a) and (b) are at MCS 0, images (c) and (d) at calculation step 50 and figures (e) and (f) present MCS 250.

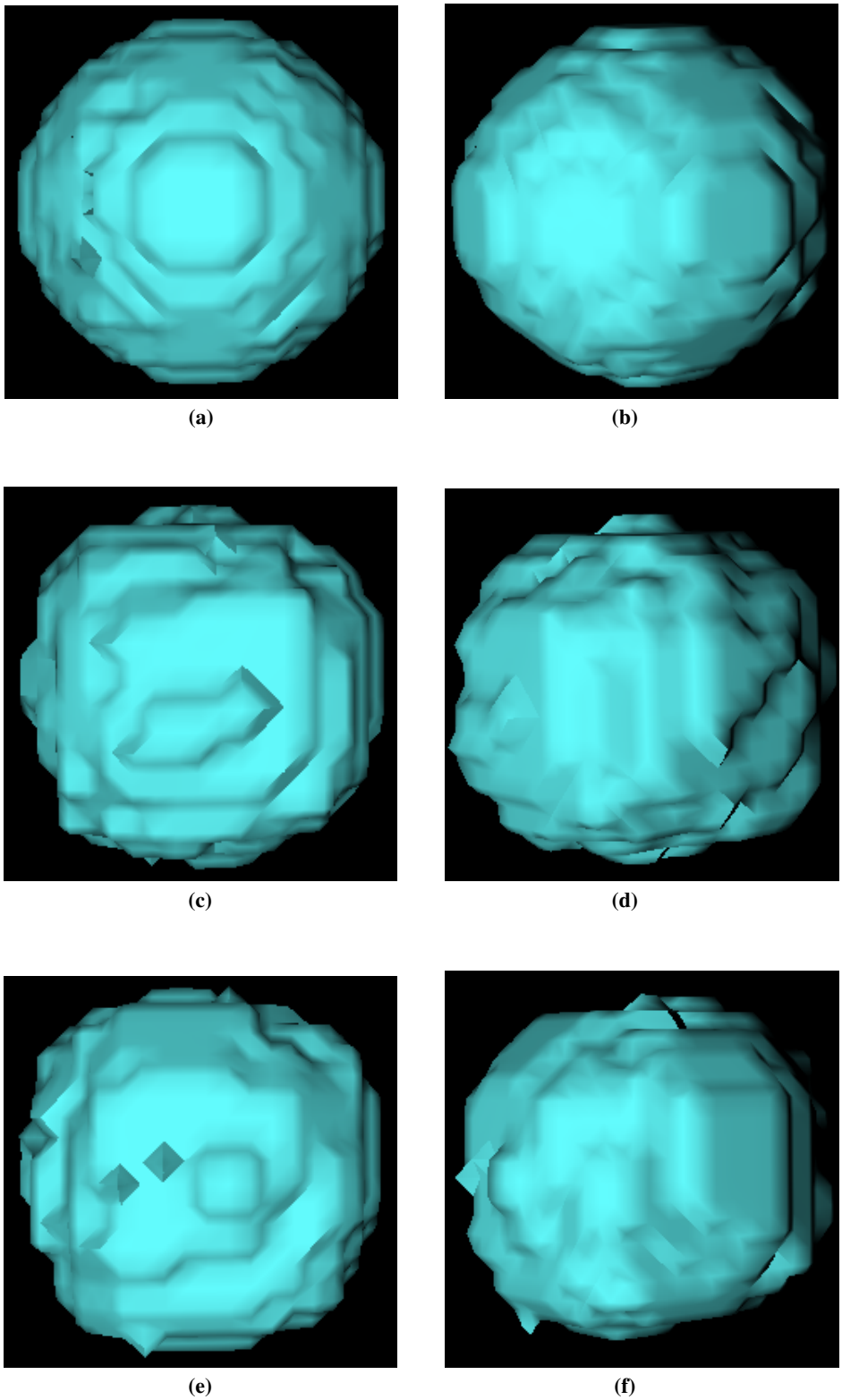


Figure 4.4: A sphere cell, with a radius of $9\,\mu\mathrm{m}$ and a voxel density of 1, as it grows. Images (a), (c) and (e) are the front view of the cell and figures (b), (d) and(f) have around a 45 degree angle of the front. Figure (a) and (b) are at MCS 0, images (c) and (d) at calculation step 750 and figures (e) and (f) present MCS 1250.

Chapter 5

Discussion

This chapter servers to discuss the developed solutions and the results from different aspects. Problems, which occured during the bachlor thesis, are presented as well.

5.1 Cell as a sphere

To draw a cell as sphere out of voxels is one way to get the desired spherish shape of the cell. Sadly to keep the shape of the cell during the growth process does not work. Because for a given volume and a given surface there are all kinds of shapes CC3D does not know which shape the cell should have. It it suprising that every time the cells get a cubish form during the growth process. It might be a result of the square lattice, which is used in the simulation. There are several research papers, which use CC3D for their simulation, sadly it is never explained how the desired cell form is achieved neither which lattice type is used. So far, it is not possible to know if there is an mistake in the simulation or if the cells get a cubish form due to the square lattice.
It is possible that a cell reaches a desired volume and surface, by only setting the terms for the volume and surface of the effective energy. The desired values can be reached faster if a high multiplier, λ_{vol} and λ_{sur}, is set. This has the disadvantage that a volume or an surface constraint might weigh more than the term of the adhesion in the effective energy. If the specific multiplier has a small value, it might be possible that it takes a long time until the desired shape is reached. In addition to the volume and surface constraint comes the adhesion. Because it was one the tasks to find a useful adhesion between cell types for the simulation it has it own

paragraph.
Other types of cell growth - Angelo tested several

5.2 Adhesion

Adhesion influences the shape of two or more cells, as it describes how strong the stick to each other. If the adhesion is set too high the cells infiltrate each other but if it is too low the cells may not stick to each other as they change their shape. During the simulations it was observable that some cells of different cell types want to infiltrate each other. This is a result of to high adhesion.

5.3 Voxel Density

In the simulations the voxel density should be set to 1. In this case, in CC3D one $1\,\mu\mathrm{m}$ is presented by one voxel. If the voxel density is higher than one voxel represents less $\mu\mathrm{m}$. Because the simulations requires a lot of more time if the voxel density is high and the deviation between the volume and surface values of a cell as a sphere out of voxels and a real sphere is larger, it should be tried to keep the voxel density as small as possible but also not to small. If the voxel density is too small than less details can be displayed and it is much more likely that the cells do not look like spheres. Why the deviation of a cell and a real sphere grows so much more for a higher voxel density is a puzzle.
On one hand the deviation grows with the radius, with more voxels. Since with a higher voxel density there are also more voxels to display it might is a result that CC3D uses more voxels to display the cell. On the other hand the volume and surface values of the drawn cell are that high that for the surface of a real sphere a factor of 6 would give an approximation to the surface of the drawn cell.

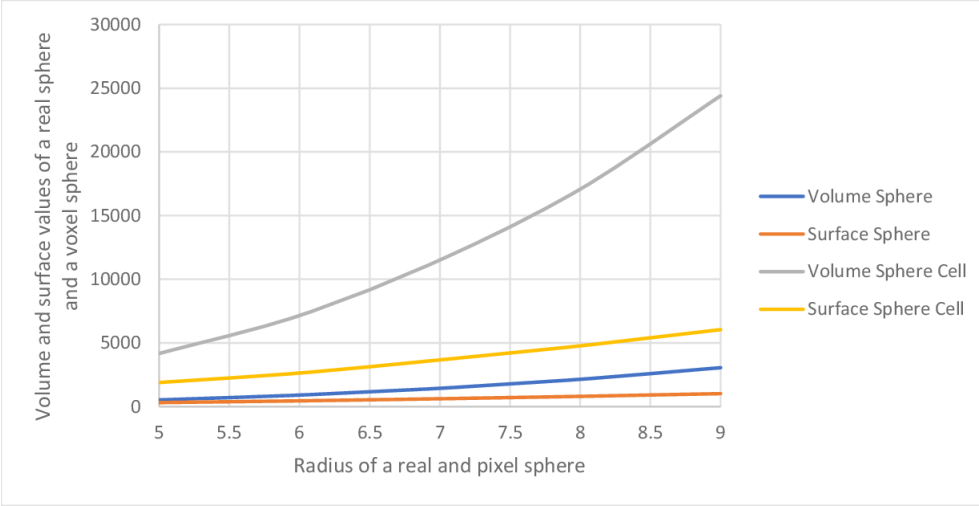


Figure 5.1

5.4 Deviation of Volume and Surface between a sphere and a cell as sphere

As it is displayed in figure 5.1 the volume of a real sphere and of a drawn sphere cell in CC3D, drawn in a simulation with a voxel density of 2, is way to far apart as this voxel density could be used for a realistic simulation of the urothelium. It is interesting to observe that the volume of a sphere cell increases this fast, whereas the surface of a sphere grows in an almost linear way for an increasing radius. Since in CC3D in a 3D simulation the volume refers to a physical volume and the surface refers to a physical surface it might be that one voxel still represent $1\,\mu\mathrm{m}$, even it should be only $0.5\,\mu\mathrm{m}$. This might explain the results, as they are similiar to the results of $2 \cdot r$ if the voxel densitiy equals 1.

5.5 Surface approximation

The approximation of the surface of a cell to a surface of a real sphere is done with voxels, because these are also used in the simulation field. There are other techniques to approximate the surface of a sphere. It is possible to use pyramids for this approximation. Since in the current simulation the square lattice is used it might is difficult to find an approximation technique which suits the square lattice better. Moreover, the volumes of the cells and of a sphere are almost identical for the same radius, if the voxel density is 1. Because the volume of a cell is more important than

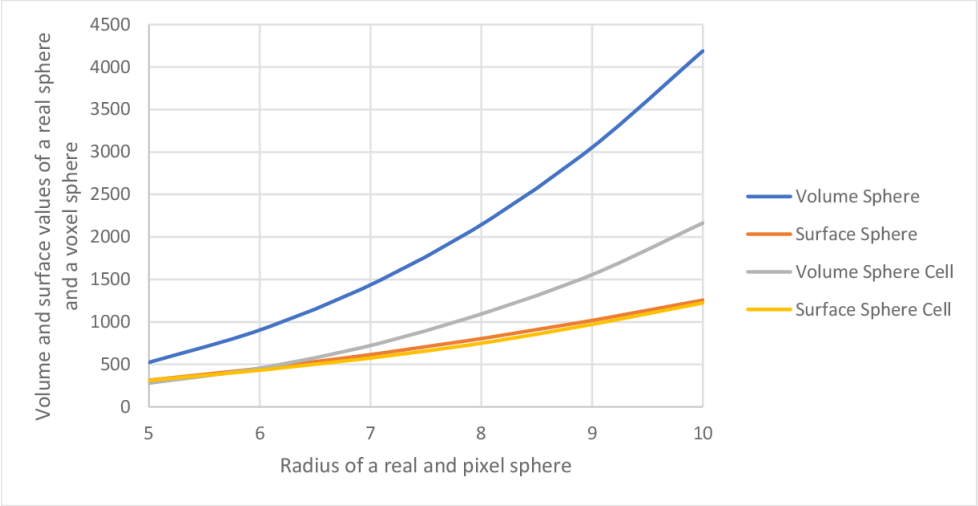


Figure 5.2

the correct surface ,as it is listed in table 2.3 at page 13, a new approximatino has to be checked against its volume as well as its surface. The use of pyramids might be in general a great way to approximate a sphere but for the square lattice it might be not as suited as the use of cuboids.

5.6 Surface Variation Pixels to Sphere

The created algorithm of section 3.10 has its benefits and also its weaknesses. There are several ways to design the algorithm. One is to use the same method as it is used to draw the sphere cell. With this method every voxel in the cuboid has to be checked if it is inside or outside the algorithm. Therefore, a lot of calculations are required just to create the matrix from which the volume and surface sites can be calculated. This amount of calculations is r^3, where the radius of the cuboid is $2 \cdot r$, as it is explained in section 3.8. Thus, the amount of calculations to create the algorithm is $(2 \cdot r)^3$, where r is the radius of the sphere. Because this approach has a high computational cost it should be only used if it is necessary.
The next two approaches are similiar. Both split the cuboid and the circle up into 8 pieces, then the calculations are done with one of the 8 cuboids and as last step this result gets mirrored. This is possible because a cuboid split up into an odd amount of parts creates several smaller cuboids. With this small cuboid it is possible to calculate if every cuboid, in the small cuboid, is inside the sphere. Then the matrix is created. For the calculation of the volume and surface the result of the one eigth

of the cuboid is mirrored for all three axes. This approach has less computations than the approach before. The amount of computations is now $(\frac{r}{2})^3$. Because only the half of the radius of the sphere is used, this is an improvement over the approach before.
The approach with the least computational cost is similiar to the one before. After the cuboid around the sphere is split up into 8 pices, the point in which the circle is at a given x,z coordinate is calculated. Then it is checked which cuboids at this given x,z coordinate are within the circle or not. Based on this approach there are even less calculations. Because of the eigth of the cuboid only the height of the circle for a given x,z coordinate is calculated. The required amount of calculations is now $(\frac{r}{2})^2)$. Because one eigth of the cuboid is used, the half of the radius of the sphere is required. Moreover, only every x,z coordinate is needed for the calculations, instead of every x,y,z coordinate of the cuboids. Thus, the approach of the algorithm has the lowest computational cost. Even the algorithm has its weakness, it might be further developed.

5.7 CC3D

Several simulation programs have been tested in an earlier master thesis of the project **??**, it was evidenced that CC3D is the best suited program. During this bachelor thesis I realized several weaknesses of CC3D which are explained in the following paragraph.
To develop a program an debugger should be available. An debugger allows the developer to execute the program command wise and to observe the values of the variables. CC3D do not offer an debugger. Therefore every observation of a variable or the order of the function calls has to be printed to the command line. This has signifacant disadvantages, as it requires more time to understand the structure of the code as well as the print commands first have to be written. Moreover, the developer has to be very precise at the print commands and she / he needs to know which output at the command line belongs to which print command.
A second disadvantage is that the memory after an simulation is not released. Every simulation requires an specific amount of memory dependent of the simulation field size. This memory should be released after the simulation is finished. If the memory would be released after the simulation other programs could use it. Moreover, by starting a simulation after another simulation is finished, CC3D reserves

the memory for the new simulation in addition to the required simulation before. Therefore, by starting several simulations without closing CC3D, the program has a high probability to crash because there is not enough memory free for the simulation.
Because the required memory for an simulation increases as the size of the simulation field increases it is good to know where the constraints of the size of the simulation field are. The size constraints was tested on an Virtual Machine (VM) with the distribution 'windows 7' as an 64 bit version. The VM has a memory of 4GB. For this memory the constraint of the size of the simulation field is around by 400 voxels for each axis. Therefore, around 107171875 voxels are allowed in the simulation. For this size of the simulation field CC3D became very slow, which is a result of the amount of memory. This constraint is different for different distributions as well as for different sizes of the memory.

5.8 GGH Model

The GGH model is widely used because it is easy to use. Even it is easy to use some parameters of the approach are not explained in detailed. As example it is possible to set a temperature for the simulation. This temperature effects how much a cell fluctuates as it changes its volume or surface. That the fluctuations changes with a different temperature is observable. Because it is almost nowhere explained it is questionable if this is the only effect of the temperature in the simulation or are the other effects with it.
For the scenario with the sphere cell it is a disadvantage that it is not possible to set a constraint of the shape of a cell. The use of one radius or one diameter would already be enough to limit the amount of possible shapes of a cell for a given volume and surface. Because such a constraint can not be set and it is not explained in the literautre how to receive a specific shape of a cell a lot of experiments has to be done to reach the desired shape of the cell.

Chapter 6

Future work

In this chapter ideas for the future of the project are presented.

As the result in section 4.5 reveals to keep a sphere shape is not possible. Because it might be a result of the square lattice it should be tested if the hexagonal lattice type suits this task better. With the hexagonal lattice the cells are hexagons in 2D and rhombic dodecahedrons in 3D. Because rhombic dodecahedron is in its form already similiar to a sphere it might work to keep the sphere shape of a cell with these objects.
In addition to the different lattice type it should be tested if another approximation to the surface and volume of a sphere is possible. An possible approximation is to use pyramids. With the use of the hexagonal lattice type the cells are made of rhombic dodecahedrons, in three dimensions. The rhombic dodecahedron consits of twelve polygons, where each of the twelve polygons is an pyramid itself. In case of a dodecahedron all parts have the same area as well as the same shape [21]. An explanation of the rhombic dodecahedron and some basic calculations can be found at Wolfram MathWorld [22].

As an extension of the project malignant cells could be simulated. Because these cells behave different from the non-malignant cells it will be challenge to implement these cells. This could increase the reality of the simulation a lot. First the hexagonal lattice should be tried to get sphere cells and a solid 3D simulation. It might is an useful extension if the 3D simulation works properly.

CC3D showed some weaknesses during the bachelor thesis. It is used for several years in the project and therefore there is a good knowledge how to handle the weaknesses. Beside the weaknesses CC3D has also its strengths. It allows to develop a

simulation program in several programming languages. This brings freedom to the developer as they can choose if a more object oriented or a script oriented program is required. It would be very useful, especially for complex programs, if a debug during the simulation would be possible. Another strength of CC3D is that it has a very active community. In the community a lot of questions are asked and members of the project as well as other community members help as much as possible. Therefore, it is possible to ask the community and get a feedback regarding the problem soon.

CC3D is not the only program which could be used. In a masters thesis of a former student it was evidenced that it is the best suited program for the simulation of the urothelium [23] at this time. This evaluation is almost 4 years in the past. Therefore, it might be possible that there are other programs as well as other approaches which could be used. One program which could be used in Morpheus. It is deveoped at the technical university in Dresden. In the most recent paper, in which the program is used, a 3D multi-scale model of fluids in the liver were simulated [24]. Another program for the simulation is Biocellion. It uses a parallel developing approach. This means the hardware sources of the computer can be used more as it is possible to let the cores of a CPU do several tasks [25]. It might is suited for this problem because a cell has a sphere shape by default [25]. These two programs might be an alternative to CC3D.

An approach which uses parallelization is the Random Walker (RW) algortihm [26]. This approach was created an uses the CPM algorithm. With the created RW algorithm a speedup of 3 to 10 times in comparison the the CPM is possible [26]. It is also mentioned that it is not enough to have a fast algorithm which allows parallelization, the developer has also to know how to use the algorithm in order to develop a faster simulation.

These programs and approaches might be an alternative to CC3D. To determine if CC3D is the best suited for the morphogenesis simulation of the urothelium, it might be useful to do a new evaluation of the different algorithms and programs for cell simulation at the market.

Chapter 7

Conclusion

After this journey with some ups and downs the 3D simulation of the urothelium is possible. During this bachelor thesis several important changes at the program and crucial observations were made. With these modifications and observations it is now possible to use the 3D simulation in order to receive new insights on the question how bladder cancer arises.

List of Abbreviations

GGH Glazier-Graner-Hogeweg

CC3D CompuCell3D

MCS Monte Carlo Step

CPM Cellular Potts Model

EPM Extended Potts Model

OOP Object Oriented Programming

RW Random Walker

VM Virtual Machine

List of Tables

List of Figures

Listings

Bibliography

[1] N. J. Poplawski, U. Agero, J. S. Gens, M. Swat, J. A. Glazier, and A. R. A. Anderson, "Front instabilities and invasiveness of simulated avascular tumors", *Bulletin of Mathematical Biology*, vol. 71, no. 5, pp. 1189–1227, Jul. 2009. DOI: 10.1007/s11538-009-9399-5. [Online]. Available: https://doi.org/10.1007/s11538-009-9399-5.

[2] [Online]. Available: https://www.everydayhealth.com/bladder-cancer/guide/.

[3] [Online]. Available: https://www.cancer.org/cancer/bladder-cancer/about/what-is-bladder-cancer.html.

[4] M. Lazzeri, "The physiological function of the urothelium–more than a simple barrier", *Urologia internationalis*, vol. 76, no. 4, pp. 289–295, 2006. DOI: https://doi.org/10.1159/000092049.

[5] T. Yamany, J. Van Batavia, and C. Mendelsohn, "Formation and regeneration of the urothelium", *Curr Opin Organ Transplant*, vol. 19, no. 3, pp. 323–330, Jun. 2014. DOI: 10.1097/MOT.0000000000000084.

[6] L. A. Birder, "More than just a barrier: Urothelium as a drug target for urinary bladder pain", *American Journal of Physiology-Renal Physiology*, vol. 289, no. 3, F489–F495, 2005, PMID: 16093424. DOI: 10.1152/ajprenal.00467.2004. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00467.2004. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00467.2004.

[7] A. A. Karl-Erik Andersson, "Urinary bladder contraction and relaxation: Physiology and pathophysiology", *Physiological Reviews*, vol. 84, no. 3, pp. 935–986, 2004, PMID: 15269341. DOI: 10.1152/physrev.00038.2003. eprint: http://www.physiology.org/doi/pdf/10.1152/physrev.00038.2003. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/physrev.00038.2003.

[8] G. Apodaca, “The uroepithelium: Not just a passive barrier”, *Traffic*, vol. 5, no. 3, pp. 117–128, 2004. DOI: 10.1046/j.1600-0854.2003.00156.x. [Online]. Available: http://dx.doi.org/10.1046/j.1600-0854.2003.00156.x.

[9] S. N. A. Puneet Khandelwal and G. Apodaca, “Cell biology and physiology of the uroepithelium”, *American Journal of Physiology-Renal Physiology*, vol. 297, no. 6, F1477–F1501, 2009, PMID: 19587142. DOI: 10.1152/ajprenal.00327.2009. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00327.2009. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00327.2009.

[10] S. A. Lewis, “Everything you wanted to know about the bladder epithelium but were afraid to ask”, *American Journal of Physiology-Renal Physiology*, vol. 278, no. 6, F867–F874, 2000, PMID: 10836974. DOI: 10.1152/ajprenal.2000.278.6.F867. eprint: https://doi.org/10.1152/ajprenal.2000.278.6.F867. [Online]. Available: https://doi.org/10.1152/ajprenal.2000.278.6.F867.

[11] H. J. L. W. R. Cross I. Eardley and J. Southgate, “A biomimetic tissue from cultured normal human urothelial cells: Analysis of physiological function”, *American Journal of Physiology-Renal Physiology*, vol. 289, no. 2, F459–F468, 2005, PMID: 15784840. DOI: 10.1152/ajprenal.00040.2005. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00040.2005. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00040.2005.

[12] [Online]. Available: http://www.compucell3d.org/.

[13] M. H. Swat, J. Belmonte, R. W. Heiland, B. L. Zaitlen, J. A. Glazier, and A. Shirinifard, *Introduction to compucell3d v3.7.4*, 2017. [Online]. Available: www.compucell3d.org/BinDoc/cc3 d_binaries/Manuals/Introduction_To_CompuCell3D_v.3.7.4.pdf.

[14] J. A. Glazier, A. Balter, and N. Poplawski, “Magnetization to morphogenesis: A brief history of the glazier-graner-hogeweg model”, *Single-Cell-Based Models in Biology and Medicine*, p. 79, 2007. [Online]. Available: https://www.researchgate.net/profile/Ariel_Balter/publication/227073495 _Magnetization_to_Morphogenesis_A_Brief_History_of_the_Glazier-Graner-Hogeweg_Model/links/00b7d52d79e94eadc7000000.pdf.

[15] F. Graner and J. A. Glazier, “Simulation of biological cell sorting using a two-dimensional extended potts model”, *Phys. Rev. Lett.*, vol. 69, pp. 2013–2016, 13 Sep. 1992. DOI: 10.1103/PhysRevLett.69.2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.69.2013.

[16] J. A. Glazier and F. Graner, "Simulation of the differential adhesion driven rearrangement of biological cells", *Phys. Rev. E*, vol. 47, pp. 2128–2154, 3 Mar. 1993. DOI: 10.1103/PhysRevE.47.2128. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.47.2128.

[17] E. Stott, N. Britton, J. Glazier, and M. Zajac, "Stochastic simulation of benign avascular tumour growth using the potts model", *Mathematical and Computer Modelling*, vol. 30, no. 5, pp. 183–198, 1999. DOI: https://doi.org/10.1016/S0895-7177(99)00156-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717799001569.

[18] N. Chen, J. A. Glazier, J. A. Izaguirre, and M. S. Alber, "A parallel implementation of the cellular potts model for simulation of cell-based morphogenesis", *Computer Physics Communications*, vol. 176, no. 11, pp. 670–681, 2007. DOI: https://doi.org/10.1016/j.cpc.2007.03.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465507002044.

[19] T. M. Cickovski, C. Huang, R. Chaturvedi, T. Glimm, H. G. E. Hentschel, M. S. Alber, J. A. Glazier, S. A. Newman, and J. A. Izaguirre, "A framework for three-dimensional simulation of morphogenesis", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 4, pp. 273–288, Oct. 2005. DOI: 10.1109/TCBB.2005.46.

[20] A. Torelli, P. Erben, J. Debatin, and M. Gumbel, "Proliferation and regeneration of the healthy human urothelium: A multi-scale simulation approach with 16 hypotheses of cell differentiation".

[21] B. K. P. Horn, "Extended gaussian images", *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1671–1686, Dec. 1984. DOI: 10.1109/PROC.1984.13073.

[22] [Online]. Available: http://mathworld.wolfram.com/RhombicDodecahedron.html.

[23] A. Torelli, "Computer simulation of urothelial cells", Master's thesis, 2014.

[24] K. Meyer, O. Ostrenko, G. Bourantas, H. Morales-Navarrete, N. Porat-Shliom, F. Segovia-Miranda, H. Nonaka, A. Ghaemi, J.-M. Verbavatz, L. Brusch, I. Sbalzarini, Y. Kalaidzidis, R. Weigert, and M. Zerial, "A predictive 3d multi-scale model of biliary fluid dynamics in the liver lobule", *Cell Systems*, vol. 4, no. 3, 277–290.e9, 2017/03/22 2017. DOI: 10.1016/j.cels.2017.02.008. [Online]. Available: http://dx.doi.org/10.1016/j.cels.2017.02.008.

[25] S. Kang, S. Kahan, J. McDermott, N. Flann, and I. Shmulevich, "Biocellion : Accelerating computer simulation of multicellular biological system

models", *Bioinformatics*, vol. 30, no. 21, pp. 3101–3108, 2014. DOI: 10.1093/bioinformatics/btu498. eprint: /oup/backfile/content_public/journal/bioinformatics/30/21/10.1093_bioinformatics_btu498/2/btu498.pdf. [Online]. Available: +%20http://dx.doi.org/10.1093/bioinformatics/btu498.

[26] F. P. Cercato, J. C. M. Mombach, and G. G. H. Cavalheiro, "High performance simulations of the cellular potts model", in *20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment (HPCS'06)*, May 2006, pp. 28–28. DOI: 10.1109/HPCS.2006.28.