

THE OHIO STATE UNIVERSITY

Epithelium

The Lightweight, Customizable, Epithelial Tissue Simulator

Author:

Melvyn Ian DRAG, B.A.

Advisors:

Dr. Marcos Manuel SOTOMAYOR

Dr. Edward OVERMAN

Presented in partial fulfillment of the requirements for the degree

Master of Mathematical Sciences

in the Graduate School of The Ohio State University

April 21, 2015

Copyright by
Melvyn Ian Drag
2015

Abstract

Epithelial tissue performs many important functions in animals, such as preventing contamination, transporting gases and nutrients, and fluid secretion. Macroscopically, epithelial tissue can be thought of as the layer of an animal that separates it from the exterior world. The geometrical and topological features of epithelial tissue make it amenable to computational modeling. There are several simulation codes in existence which reproduce certain aspects of epithelial tissue morphogenesis, wound healing, and equilibration, but to the best of our knowledge only one of them is freely available to the public. Unfortunately, installation and use of this software requires expertise in a unix-like operating system and advanced knowledge of several programming languages. With this in mind, I have developed *Epithelium*, a lightweight epithelial tissue simulator which compiles easily on any unix-like system, and which can also be distributed as precompiled binaries. The code has very few dependencies, and these dependencies are likely already satisfied by the default packages installed on a Linux or Mac computer. For users with access to NVIDIA GPUs, *Epithelium* comes in stable and beta parallel versions. In *Epithelium* simulations are fairly easy to design and run via several configuration files, the source code is highly modularized, and the algorithms used therein are extensively documented. As such, this code is useful for reproducing previous results, and for quickly designing new computational biology experiments of epithelial tissues.

a mis escuincles Alfie and Andy

Acknowledgements

First and foremost, I would like to thank Dr. M. Sotomayor for his all around support and encouragement. Dr. Sotomayor provided me with a wonderful computer equipped with two fancy monitors, an expensive NVIDIA GPU, a powerful CPU, and the OS of my choice with which I was able to do some great work. He also helped fund my travel expenses and wrote some fantastic letters of recommendation that got me accepted to some conferences, and helped me secure a fellowship. In his lab I was able to present my work regularly and received great feedback from him and from the other lab members about how to make my presentations more appealing to a variety of different audiences. In general, I couldn't have had an advisor who was more energetic, more encouraging, and more dedicated to providing me with all the tools I needed to produce great work.

I also have to thank the members of the Sotomayor lab for attending my presentations, and for being such great company during the last two years!

I am also indebted to Dr. E. Overman. Dr. Overman wrote me wonderful letters of recommendation to get my travel expenses covered, and to make sure that I received a fellowship. Dr. Overman was my professor for the *second* hardest class I ever took, and taught me to not pull my hair out when my homework had me stressed. I would have been bald. Even after watching me drown in his class, he still accepted me as his student and has since been a source of great mathematical, programming, and stylistic advice.

I have to thank the Mathematics Department for their financial support. I received a generous fellowship during my first year at OSU, got to TA a great class during my third semester, and was then granted another ample fellowship for my last term.

Vita

June 2006	Bayonne High School
June 2012	New Jersey City University

Contents

1	Epithelial Tissue and Vertex Dynamics	1
1.1	About Epithelial Tissue	1
1.2	Modeling Epithelial Tissue	3
1.3	The Nagai-Honda Model	6
1.3.1	How the Vertices Move	6
1.3.2	Topological Changes to the Mesh	8
1.3.3	Selection of Parameters	10
1.4	Further Remarks About Epithelial Tissue Modeling	11
1.4.1	Three dimensional models: Honda-Nagai and Okuda.	11
1.4.2	Similar Models of Potential Energy	13
1.4.3	The T3 Swap	13
1.4.4	The Euler Characteristic and Its Implications	14
2	Epithelium	17
2.1	About Epithelium	17
2.2	Sample Configuration Files	17
2.2.1	Simulation Settings, the Initial Mesh, and config.txt	17
2.2.2	Global Cell Parmeters	19
2.2.3	Local Cell Parameters	19
2.3	Monitoring the Output of Epithelium	20
2.4	The Design of <i>Epithelium</i>	22
2.4.1	[Highly Simplified] Pseudocode	22
2.4.2	Classes	23
2.5	Executive Decisions About Parameter Implementations	25
2.6	A Relational Database	26

2.7	Initial Mesh Design	27
2.8	Moving the Vertices	28
2.9	Error Tolerance of the Algorithm	29
2.10	Embarassing Parallelism and CUDA	30
2.11	Computing Topological changes.	32
2.11.1	The T2 swap.	32
2.11.2	The T1 Swap	32
2.12	Dependencies	33
2.13	Complexity	35
3	The Future of <i>Epithelium</i>	37
3.1	Rosette Models	37
3.2	A New Potential, A New Force	38
3.3	Improvements in Imaging	38
3.4	Further Parallelization	39
3.5	Voronoi Tesselations	40
3.6	Visualization Software	41
A	Getting, Running, and Modifying the Code	47
A.1	GitHub	47

List of Figures

1.1	The Types of Epithelial Tissue [14].	1
1.2	Various Models of Epithelial Tissue	2
1.3	The Weliky-Oster Force.	4
1.4	The Giant's Causeway. Adapted from [19].	5
1.5	A T1 Swap.	9
1.6	A T2 Swap	9
1.7	Distribution of Cell Shapes	10
1.8	The HOI swap.	11
1.9	A non-smooth energy curve.	12
1.10	The T3 Swap.	14
2.1	Cells before and after dynamics	18
2.2	Energy graphs.	20
2.3	Equilibrium area graphs	20
2.4	Equilibrium area graphs	21
2.5	Equilibrium distribution of cell shapes.	21
2.6	An example of a relational database.	27
2.7	Data structure used for storing data in a non-redundant form.	28
2.8	A 5x5 Hexagonal Mesh.	29
2.10	A rotated mesh for error analysis.	30
2.9	Getting Cells in Order	31
2.11	Implementation of a T1 swap.	34
2.12	The runtime does not scale as the number of vertices.	35
2.13	The runtime does not scale as $ V ^2$	36
3.1	Square cells in quail epithelium. Image courtesy of [59].	37

Chapter 1: Epithelial Tissue and Vertex Dynamics

1.1 About Epithelial Tissue

Epithelial tissue covers the interior and exterior surfaces of our bodies. Skin, the lining of the esophagus and intestines, the urethra, the lining of the lungs and the bronchioles are all made up of epithelial tissue. In this way, we can think of epithelial tissue as being the envelope in which our contents are packaged [26]; epithelial tissue is our interface with the outside world.

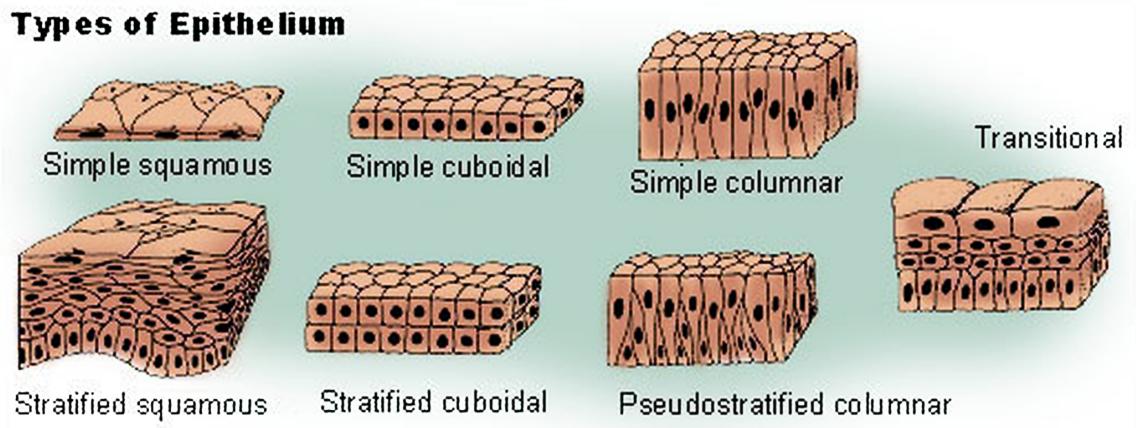


Figure 1.1: The Types of Epithelial Tissue [14].

As Figure 1.1 shows, there are many types of epithelial tissue in animals which vary in their number of layers and how the cells are shaped. Each of these types of cells are found in a different region of the body where they perform a specific function. For example, the simple squamous epithelium is no more than one layer of cells thick, and the cells are much flatter than they are wide. Because these cells are well suited to allow diffusion across themselves, simple squamous tissue is found in the walls of blood vessels and in the alveoli in the lungs, where the diffusion of oxygen occurs. On the other hand, columnar cells are much taller than they are wide, and are thus well suited to absorption. These cells are found in the intestines where they absorb nutrients from passing food. Stratified squamous epithelia are several layers thick and line the esophagus and mouth and protect against

abrasion.

What all of these tissues have in common, however, is how amenable they are to computational modeling. The most easily modeled tissues are simple epithelia, which typically have near-uniform height, and very little difference in appearance between their apical and basal faces. This means that the cells can easily be approximated by a two dimensional mesh in which the surfaces where two cells touch are approximated by a line. Examples of 2D epithelial tissue simulations are presented in Figure 1.2a,b. For an example of a 2D epithelial simulation on a three dimensional surface, see Figure 1.2d. The 3D simulation of stratified tissue is more difficult to implement than a two dimensional model¹. For an example of one successful 3D model, see Figure 1.2c.

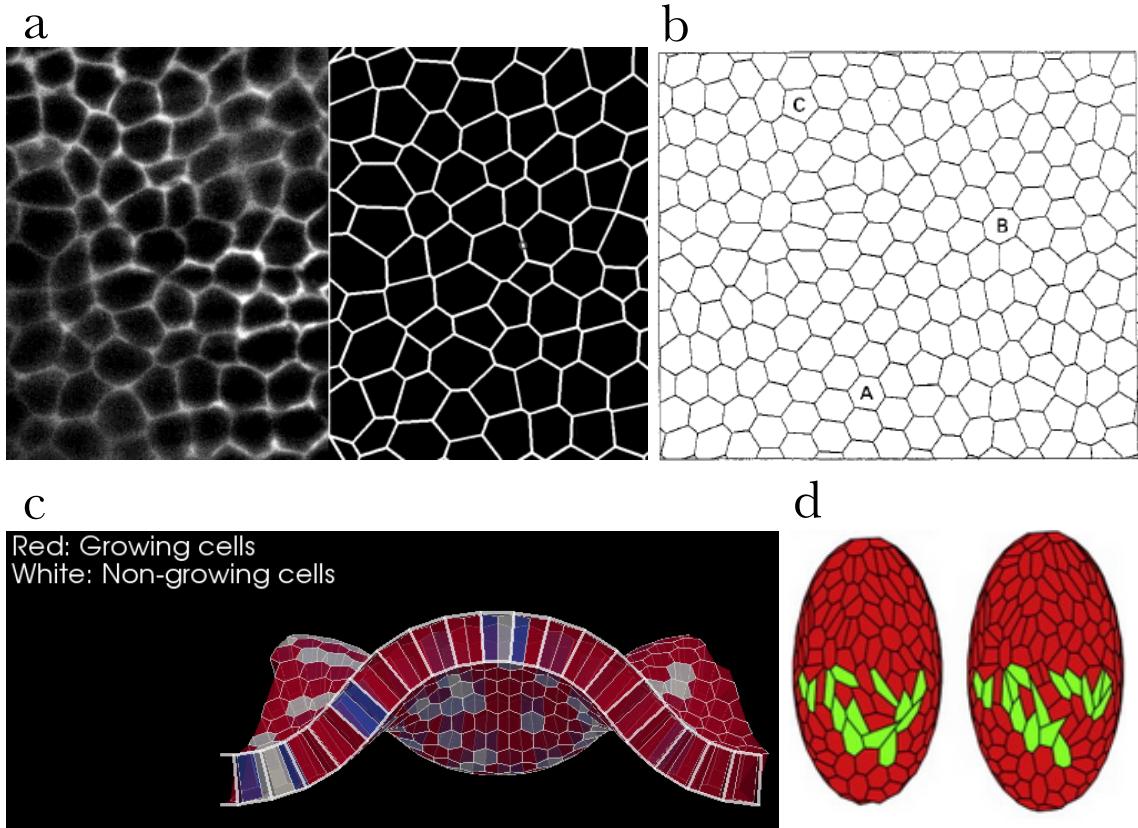


Figure 1.2: Several screenshots illustrating the variety of epithelial tissue models. (a) A comparison of living tissue (left) to a simulation (right) [4]. (b) A diagram from the original Honda-Nagai paper [37]. (c) A three-dimensional mesh of cells [44]. (d) A tissue developing on a surface [16].

Current modeling is producing great results in the field of epithelial tissue morphogenesis,

¹Even a leading epithelial tissue simulator, Chaste [17], still does not have stable 3D modeling capabilities.

equilibration, and wound healing. The Honda-Nagai model, which we will discuss in great detail below, successfully reproduced the wound healing of cats' corneas [36]. This model has also been able to reproduce all of the essential dynamics of epithelial tissue [37]. Current imaging tools have enabled the recording of epithelial tissue dynamics *in vivo* [53, 58], providing a wealth of experimental data which can serve as either initial conditions for simulations, or as benchmarks to measure the accuracy of computational predictions. In turn, models of epithelial dynamics can provide insights into the physical parameters that govern tissue development, maintenance, and malady.

Other modeling communities share advanced, free, and parallel simulation codes. For example, consider LAMMPS [30] for simulating atomistic materials, and CHARMM [9], Amber [1], and NAMD [41] for the molecular dynamics simulation of biomolecules. Unfortunately, there are only a handful of codes in use for the simulation of epithelial tissue, and to the best of our knowledge only one of them is freely available [17]. In this thesis I will present the basic ideas of **vertex dynamics models** of epithelial tissue, and then describe the implementation of one of them as a freely available modeling tool for the community.

1.2 Modeling Epithelial Tissue

A two dimensional **vertex dynamics model** of epithelial tissue is made up of vertices and edges [23]. The vertex dynamics model presupposes that the movement of cells in epithelial tissue can be approximated by the movement of edges and vertices. Some force is then proposed to guide epithelial cell movement according to an equation of motion that is solved via some numerical method.

Epithelial vertex dynamics has been a lively field of research since the 1970s because of several heartening results. Some researchers have had success modeling the morphogenesis of *Drosophila* wing growth [15], whereas others have accurately reproduced the dynamics of corneal wound healing [36]. In other research, simulations have faithfully captured the effects of laser perturbations to epithelial cell junctions [4], and others have quantified parameters which are important in describing the formation of the epithelial envelope in *Drosophila* [53]. Unfortunately, these results have not come from one standard model of epithelial tissue development, but from a variety of different, often irreconcilable, models. Two different approaches to epithelial simulation described below will clearly illustrate the variety of techniques in practice.

In the model developed by M. Weliky and G. Oster, forces due to osmotic pressure and contractile tension describe how vertices move [48]. This model also allows for certain forces external to the tissue to be applied at each node. In the end, the force applied to each vertex in the mesh is given by

$$F_i = F_{ext} + \sum_{n=1}^N (T_{i-1}^n - T_{i+1}^n + P^n)$$

where n is the index of the n^{th} cell which touches vertex i . The force applied to vertex i coming from cell n is seen graphically in Figure 1.3.

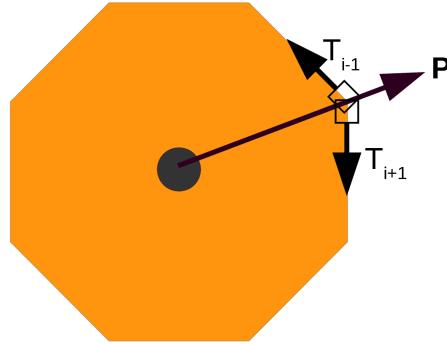


Figure 1.3: The Weliky-Oster Force.

In contrast, the model developed by H. Honda and T. Nagai takes an approach to modeling epithelial tissues rooted in the study of cellular structures ² [38]. In the fantastic review paper *Soap, Cells, and Statistics* [57], D. Weaire and N. Rivier argue for the existence of some natural mechanism underlying the development of epithelial tissue, columnar basalt formations, soap froths, grain growths, and other cellular structures, as they exhibit a great deal of similarity. For example, consider the images of epithelial tissue presented throughout this paper juxtaposed with the image of The Giant's Causeway in Northern Ireland in Figure 1.4. The equilibrium states of these structures all contain primarily hexagonal cells, and three cells typically meet at any junction. There are some differences in the exact distribution of cell shapes, the presence of chemicals in biological tissues versus the absence of growth inducing chemicals in geological structures, and the active migration of biological cells versus the entirely passive movement of soap froths; still, the authors conjecture that

²Including living and non-living structures.

the dominant principle behind all cellular dynamics is the principle of maximum entropy, by which the structures seek a state with minimal potential energy.



Figure 1.4: The Giant’s Causeway. Adapted from [19].

A very basic result from physics is the relationship between force and potential energy:

$$\vec{F} = (F_x, F_y, F_z) \quad (1.1)$$

$$W = -\Delta U(\vec{x}) = \int_{x_0}^x F_x dx + \int_{y_0}^y F_y dy + \int_{z_0}^z F_z dz \quad (1.2)$$

$$\nabla(-\Delta U(\vec{x})) = \nabla \left(\int_{x_0}^x F_x dx + \int_{y_0}^y F_y dy + \int_{z_0}^z F_z dz \right) \quad (1.3)$$

$$-\nabla U(\vec{x}) = \vec{F} \quad (1.4)$$

In the Honda-Nagai model, the authors posit that dynamics of epithelial cell packing is dominated by their seeking a state with minimal potential energy. They describe several types of potential energy in a tissue, take a gradient of the energy function as described above, and then apply the resulting force to the vertices in the epithelial mesh [37].

While both the Honda-Nagai and the Weliky-Oster models successfully reproduce the topological and geometric properties of epithelial tissue, I have chosen to focus my efforts on the Nagai-Honda model. This was the original vertex dynamics model, it still enjoys considerable use by other researchers, and is of a form quite similar to that used by others [15].

1.3 The Nagai-Honda Model

1.3.1 How the Vertices Move

In 1989, K. Kawasaki showed that the dynamics of grain growth can be reduced to a first order system given by:

$$\eta \frac{dr_i}{dt} = F_i \quad (1.5)$$

where F_i denotes the force applied to vertex i , r_i denotes the position of the i^{th} vertex, and the left hand side is the velocity of the vertex multiplied by a positive drag coefficient, η [29]. This is equivalent to $m_i \frac{d^2 r_i}{dt^2} + \eta \frac{dr_i}{dt} = F_i$, in which the inertial term is considered to be much smaller than the drag term.

Based upon the notion that biological cells move in a way quite similar to high temperature crystallites³, the Honda-Nagai model has equation(1.5) as its basis. The force on the right hand side of the equation is in turn defined as the gradient of an energy function, since the model presupposes that the tendency towards a state of lower potential energy is the guiding principle behind epithelial tissue equilibria. The energy function is composed of three terms which reflect the properties of biological cells.

The first two potential energy terms come from the assumption that the cell is elastic, and that the cell wants to return to a target shape. Therefore, the first two energy terms are of the harmonic form:

$$C(x - x_0)^2 \quad (1.6)$$

where x is some physical quantity, and C is some constant. The plot of this energy is therefore a parabola with a minimum at $x = x_0$, and the farther the quantity x strays from the equilibrium, the steeper its gradient will be, and the more forcefully it will want to return to equilibrium.

The third energy term is an adhesion energy, which is proportional to the amount of interfacial surface area between a cell and its neighbor. There is a successful theory in biology called the **differential adhesion hypothesis** which attempts to account for certain cellular distribution phenomena through their adhesive binding tendencies. The theory essentially says that certain cells tend to bond more tightly to cells of type A than to cells of type B due to the presence or absence of different adhesion proteins in the

³Often referred to as grain growth.

membranes of these cells [18]. In proliferating tissues, this difference in binding energies leads to cell segregation and the formation of structured tissues and organs.

The precise definitions of the potential energies are:

1. The deformation energy term U_D is given by

$$U_D = \lambda(A - A_0)^2 \quad (1.7)$$

where A is the cell area, A_0 is the target cell area, and λ is some positive constant.

2. The membrane surface energy term U_S is given by

$$U_S = \beta(P - P_0)^2 \quad (1.8)$$

where P is the cell perimiter, P_0 is a target perimeter, and β is some positive constant.

3. The cell-cell adhesion energy U_A is given by

$$U_A = \sum_{j=1}^n \gamma_j d_j \quad (1.9)$$

where n is the number of vertices in the cell, γ is some constant for the boundary in question between one cell and another, and d is the distance between one vertex and the next in a counter clockwise fashion. Note that in two dimensions the boundary is a distance d , but in three dimensions it would have to be the area of a cell face.

In total, the potential energy in a sheet of N cells is given as:

$$U = \sum_{c=1}^N \left(\lambda_c (A_c - A_{0c})^2 + \beta_c (P_c - P_{0c})^2 + \sum_{edges \in c} \gamma_{edge} d_{edge} \right)$$

As seen in [17], the negative gradient of this potential energy is:

$$\begin{aligned} F_i = - \sum_{l \in N_i} & (2\lambda(A_l - A_{0l}) \nabla_i A_l + 2\beta(C_l - C_{0l}) (\nabla_i d_{l,I_l-1} + \nabla_i d_{l,I_l}) + \\ & \gamma_{l,I_l-1} \nabla_i d_{l,I_l-1} + \gamma_{l,I_l} \nabla_i d_{l,I_l}) \end{aligned} \quad (1.10)$$

where l is the l^{th} cell containing vertex i , given a counter clockwise orientation. I_l is the local index of node i in element l . A detailed derivation of the force follows, as in [17].

The area of a convex cell made up of n vertices is given by Gauss's shoelace formula:

$$A = \frac{1}{2} \sum_{i=1}^n \left(x_i y_{(i+1) \text{mod}(n)} - x_{(i+1) \text{mod}(n)} y_i \right) \quad (1.11)$$

Therefore, the gradient is given by:

$$\nabla_i A_l = \frac{1}{2} \left(y_{I_l+1} - y_{I_l-1}, \ x_{I_l-1} - x_{I_l+1} \right) \quad (1.12)$$

where the subscripts l denote that x, y are in cell l . The subscripts are local indices in the cell l , and the orientation of vertices is counterclockwise. The circumference is given by:

$$P = \sum_{j=1}^N d_j = \sum_{j=1}^N \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \quad (1.13)$$

Therefore

$$\nabla_i P = \nabla_i d_{i-1} + \nabla_i d_i \quad (1.14)$$

and

$$\nabla_i d_{l,j} = \frac{1}{d_{l,j}} \left(x_{j+1} - x_j, \ y_{j+1} - y_j \right) \quad (1.15)$$

Substituting the above values into the equation:

$$-\nabla_i U = -\nabla_i (U_D + U_S + U_A) = F_i \quad (1.16)$$

gives the force described in equation(1.10).

1.3.2 Topological Changes to the Mesh

There is empirical evidence that nearly all vertices in a sheet of epithelial tissue have coordination number three (most vertices have three incident edges) [39]. This observation has led many researchers in the field of cellular structures to consider what sort of topological changes can occur in meshes of cells without changing their connectivity [57]. As it turns out there are three possible changes called the T1, T2 and T3 swaps, and the original Honda-Nagai Model implements the first two.

The T1 swap, illustrated in Figure 1.5, is also called a “neighbor exchanging swap” because two cells that were adjacent cease to be neighbors and two cells that were not adjacent become neighbors. The T1 swap occurs when two vertices become critically close to each other, and instead of allowing the vertices to collide we rotate the offending edge.

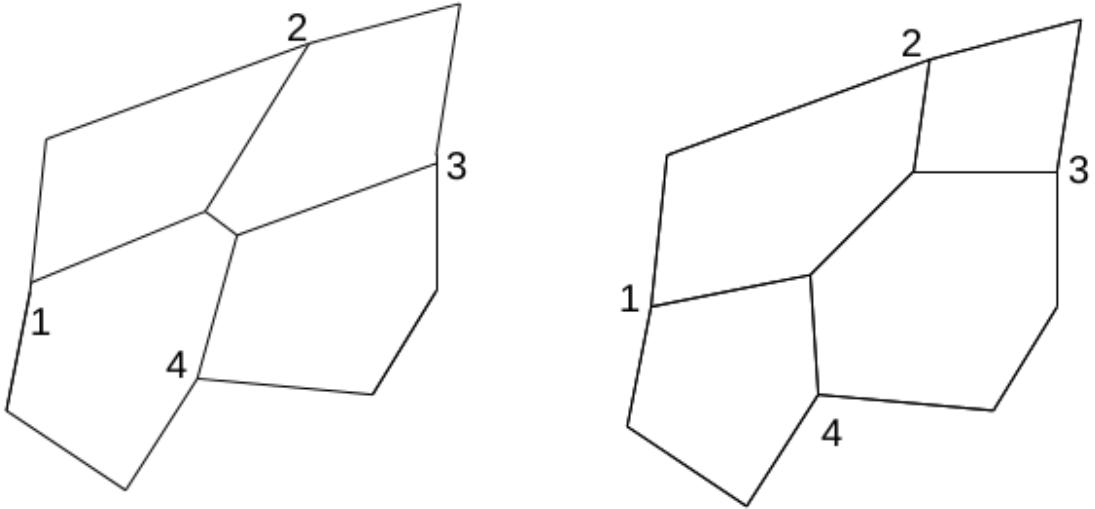


Figure 1.5: A T1 Swap.

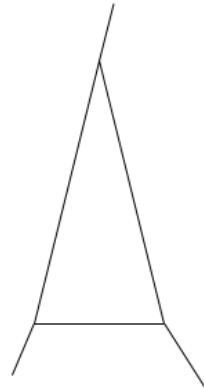


Figure 1.6: A T2 Swap.

There is no specification in the literature about how to rotate the edge, but the natural choice is to turn the edge by 90 degrees. In nature this should correspond to two vertices getting very close, colliding, and then flattening out into an edge. The Honda-Nagai model performs this action discretely as a simplifying measure to avoid having to handle the momentary degree four vertex.

The second topological change is the T2 swap, which is also known as “cell removal”. A T2 swap occurs when an edge of a triangular cell becomes too small and the cell is deleted and replaced by a single vertex located at the centroid of the cell (Figure 1.6).

1.3.3 Selection of Parameters

The basic vertex dynamics model requires the user to specify the A_0 , P_0 , and γ_{edge} parameters for each cell, as well as a value for the drag coefficient η and the integration timestep dt . The equations in this model are dimensionless. I will not undertake a discussion of how to derive the dimensionless model from the dimensional model, but for the curious reader this is all laid out in [37]. Typically, one would not choose the values of the aforementioned parameters, but would instead have some dimensional biological data and go through the necessary conversion steps to use them in the simulation [45].

Interestingly, I have found very few explicit statements of the parameters used in simulation (exceptions are [36, 17, 45]). As Honda has said, “We do not have accurate values for the cell parameters at present” [25]. Very recently, new imaging techniques have permitted the *in vivo* observation of epithelial tissue morphogenesis⁴ [53, 58] and this will likely open new doors for a more accurate parameterization of current models, or perhaps even for the reformulation of the expressions for forces and potential energies.

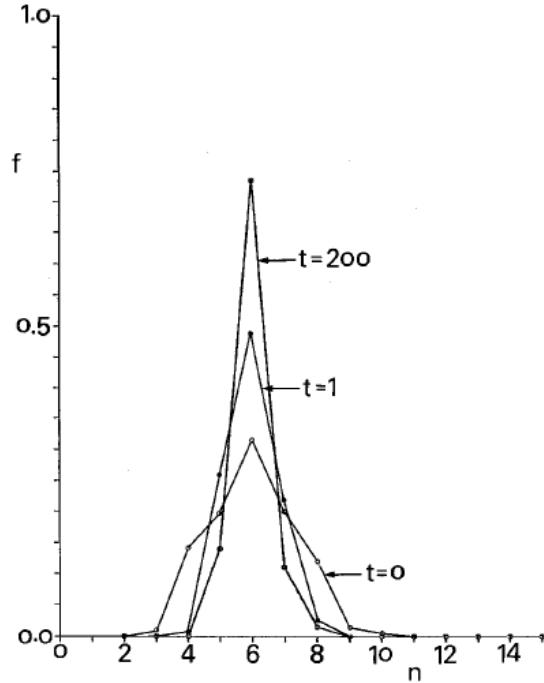


Figure 1.7: The distribution of cell shapes as a function of time in the original Honda-Nagai Model. Adapted from [37]

⁴Morphogenesis is the development of shape in an organ or organism.

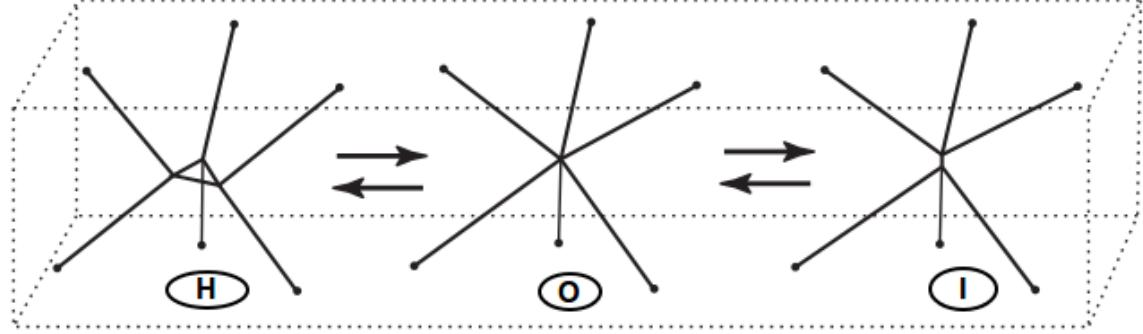


Figure 1.8: The HOI swap. H configuration(left), O configutation (center), I configuration (right). Adapted from citeHonda3D.

In the case of the Honda-Nagai Model, however, there is little difference between equilibrium states attributed to various parameter choices (See Chapter 2). One of the parameter-independent defining characteristics of the Honda-Nagai model is the strong tendency toward six-sided cells in equilibrium (Figure 1.7). Yet, it has been shown that different parameter values coupled with other mesh changing operations (such as oriented cell division) can cause drastically different types of morphogenesis [20]. For example, drosophila wings, with their highly oriented divisions, have been shown to contain approximately 80% hexagonal cells whereas simulations of tissues with purely stochastic divisions converge to approximately 47% hexagons [39]. While all epithelial tissue has a strong tendency towards achieving an equilibrium dominated by hexagons, the width of the distribution of cell shapes differs by cellular structure and, hence, by parameter choices [57].

1.4 Further Remarks About Epithelial Tissue Modeling

Over the years, various modifications and improvements have been made to the Honda-Nagai model. These changes involve new ways of specifying the potential energy, adding new cell dynamics, and changing the connectivity of the mesh. In this section I will discuss some of these advances, as well as present some important mathematical theory underlying epithelial tissue models.

1.4.1 Three dimensional models: Honda-Nagai and Okuda.

Honda and Nagai also implemented their model in three dimensions and used the same equation of motion for the tissue [25]. However, while the vertices in two dimensions all

meet at the junction of three edges, in three dimensions the vertices meet where four edges are coincident, and each edge is touched by four polygonal faces. A new type of topological change handles the new connectivity, as the T1 and T2 swaps work only on two-dimensional meshes. The so-called HOI swap (Figure 1.8) looks at all of the edges in the mesh and finds the edges measuring less than δ ; then, it proceeds on a case by case basis. If the edge lies on a triangle, then it is of the *H* form, and the swap goes from left to right (as indicated in Figure 1.8). Otherwise, the edge is of the *I* form, and the transformation goes from right to left [25]. The Honda-Nagai model in three dimensions has been successful in describing how embryonic epithelial cells grow in a plane at the expense of not proliferating in the orthogonal direction, but is not the only 3D epithelial growth model.

The 3D vertex dynamics models recently developed by Okuda and co-workers take a slightly different approach than Honda and Nagai. The new models of epithelial tissue morphogenesis feature an altered cell reconnection model (Reversible Network Reconnection, or, RNR) which allows new tissue dynamics to develop [44], and include a new viscosity term in the equation of motion which gives the model more physical plausibility [46].

The proposed reconnection scheme contrasts with the Honda-Nagai model in that there are no large inconsistencies in the energy output (Figure 1.9 and Figure 2.2).

The principal idea behind the RNR is that a triangular element ought not be reduced to a point unless *all* of its edges are critically small, whereas in the two-dimensional Honda-Nagai model a T2 swap (and in the three-dimensional model, an HIO swap) occurs when only one edge of a triangular face is smaller than δ . This can lead to jumps in potential energy, as well as generate artificial tissue dynamics which only occur because edges which are larger than the approximated zero length (δ) are forced to zero. Due to the fact that a triangle is forced to a vertex, the occurrence of an oscillation through several T2/HOI swaps

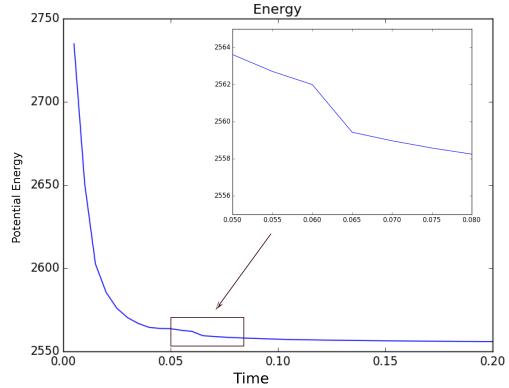


Figure 1.9: A jump in the potential energy. The reconnection scheme of the Honda-Nagai model can result in curves which are not smooth under certain circumstances.

at a vertex is not seen by the model. The two drawbacks to the RNR model are that it may not produce quality results when simulating tissues in which topological irreversibility is a defining feature [?], and that the model can sometimes produce topologies which are not computable by the software implementing it, i.e. when one edge of a triangle becomes critically small, while the other two remain long, and the triangle becomes a double edge between two vertices [46]).

Okuda's other innovation was the introduction of a new viscosity term into the equation of motion for a vertex, giving:

$$F_i = \eta_i \left(\frac{dr_i}{dt} - \vec{v}_i \right) \quad (1.17)$$

where the \vec{v}_i is a vector which describes the viscous force acting on a vertex as a result of its local interactions with neighboring vertices. Observations of growing tissues show that the viscosity of a growing tissue is inhomogeneous and depends upon the cells types in a region as well as the membranes interacting with the cells, yet prior vertex models did not take this into account. For this reason, the Honda-Nagai 3D model was unable to capture the morphogenetic dynamics seen in [35], while the Okuda model was.

1.4.2 Similar Models of Potential Energy

Interestingly, as mentioned in section 1.2, the Honda-Nagai form for the energy in a vertex is quite similar to the form developed by Farhadifar [15]. The Farhadifar formulation is:

$$E_i = \sum_{cells} \frac{K}{2} (A - A_0)^2 + \sum_{edge} \gamma_{edge} d_{edge} + \sum_{\alpha} \frac{\beta}{2} P_{\alpha}^2 \quad (1.18)$$

Remember the formulation of the Honda-Nagai energy:

$$U = \sum_{cells} \left(\lambda (A_c - A_{0c})^2 + \beta (P_c - P_{0c})^2 + \sum_{edges_c} \gamma_{edge} d_{edge} \right)$$

The equations are nearly identical, except that the Farhadifar model asserts that the cell perimeter persistently tries to collapse the cell ($P_0 = 0$), while the potential energy due internal pressure ($A_0 \neq 0$) resists this tendency. Nevertheless, the topological results of this model are not wildly different from the Honda-Nagai results [15].

1.4.3 The T3 Swap

The T3 swap is also known as “mitosis” or “cell division”. Cell division was not a part of the original Honda-Nagai Model [37] that dealt with the equilibration of a fixed number

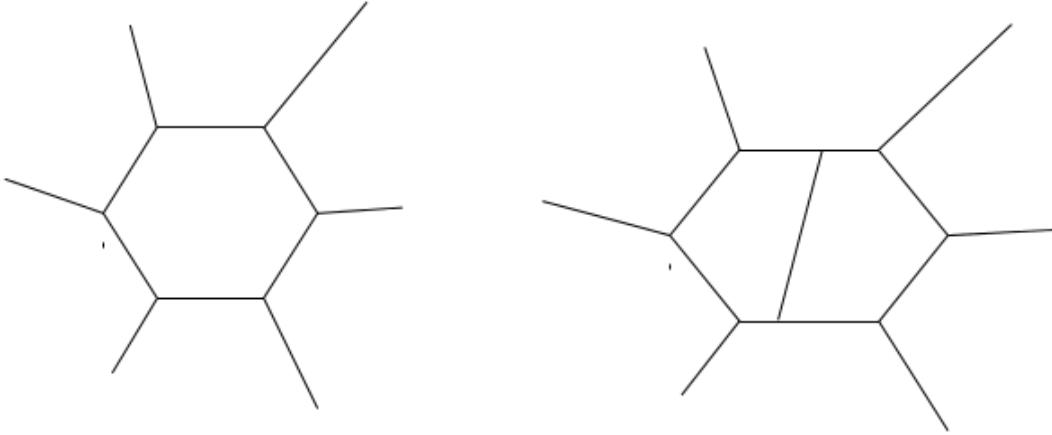


Figure 1.10: The T3 Swap.

of cells. However, during proliferation cells divide, and computational models need to take into account tissues with varying numbers of cells. The challenge with implementing the T3 swap is that there are infinitely (within the bounds of floating point arithmetic) many choices about where to divide a cell, and there are several competing opinions (though no unanimously accepted theory) about how the division is oriented. Some cells divide along their longer axis, which is known as the ‘Hertwig’s Long Axis Rule’, but global tissue stress and local cell geometry are also thought to affect the orientation of mitosis [51, 21]. The computational realization of a T3 swap is trivial, as the swap occurs by placing two new vertices along the edges of a cell and joining them by a new edge. The trouble is that there is no specification about which edges ought to have vertices implanted, or where to insert these vertices. The choice of where to divide a cell in a proliferating tissue can have profound effects upon the geometric appearance of a tissue - indeed, improperly oriented cell divisions are an indicator of cancer [39, 50].

1.4.4 The Euler Characteristic and Its Implications

The majority of current vertex dynamics models assume that all vertices have a coordination number of 3, since empirical evidence shows that the vast majority of cells have this property [39, 20]. In this section I will expand upon some observations made in [57] that deal with this phenomenon. **Euler’s Formula** is an equation which relates the number of edges, faces, and vertices in a graph or polyhedron. An invariant χ relates the faces, edges, and vertices as follows:

$$\chi = V - F + E \quad (1.19)$$

The invariant depends upon the graph or polyhedron in question. We will ignore the exact value of χ and simply use the fact that it is a constant. We know that each vertex connects to exactly three other vertices. Then we notice that all edges have two vertices, and that all vertices are connected to three edges. Initially, our intuition tells us that there should be three times as many edges as vertices, which leads us to the incorrect formula:

$$3V = E \quad (1.20)$$

But then we notice that if we consider all of the vertices in the mesh, we count each edge twice, so we divide by two and then simplify to get:

$$3V = 2E \quad (1.21)$$

Similarly, if we consider how to relate the number of edges to the faces in the mesh, we conjecture that the number of edges in the mesh is equal to the sum of the products of cell shapes by the sides, k , per shape. More clearly, we might guess the following:

$$\sum_{k=3}^N kF_k = E \quad (1.22)$$

where N is the highest number of edges in any cell in the mesh. But in this way we have again counted all of the edges twice, so the true number of edges must be the summation above divided by 2. We simplify the equation to get

$$\sum_{k=3}^N kF_k = 2E \quad (1.23)$$

Now, we are able to reduce Euler's Formula to one variable using the relationships given above.

$$V - F + E = \chi \quad (1.24)$$

$$\frac{2E}{3} - F + E = \chi \quad (1.25)$$

$$\frac{5E}{3} - F = \chi \quad (1.26)$$

$$\frac{\sum_{k=3}^N kF_k}{6} - F = \chi \quad (1.27)$$

$$\left(\frac{\sum_{k=3}^N kF_k}{F} - 6 \right) F = 6\chi \quad (1.28)$$

Biological cells are very small, and an epithelial tissue is composed of many cells, so we assume that $F \rightarrow \infty$ and then immediately notice that the expression in parentheses must tend to zero as F goes to infinity, or else the left hand side of the above equation will not approach the constant 6χ . From here it is easy to see that the introduction of a finite number of vertices with coordination number higher than three will not affect this result in the limiting case. Of course we have no reason at this point to assume that there is even one cell in the mesh with exactly six edges. It is feasible that the tissue is entirely be made up of five and seven sided cells. Nevertheless, empirical evidence shows a strong central tendency in the distribution of cell shapes. Whenever a cell tries to stray from the average, there are computational means (such as the T3 swap) of recentering the distribution at 6 sides.

The choice to impose degree three on each vertex is not one hundred percent consistent with nature, but has been a part of most models of epithelial tissue. It is a simplifying assumption that *rosettes* (epithelial cells organized radially about one vertex which has degree greater than or equal to 4) do not change the global dynamics of the development of an epithelial tissue. Recent computer vision developments [31] have made it easier to detect rosettes in epithelial tissue samples and may in the future provide information about the number of these formations, or evidence that rosettes are an important feature of epithelial tissue. Models would need to be revised to handle the introduction of vertices of higher degree.

Chapter 2: Epithelium

2.1 About Epithelium

Here I present my implementation of the Honda-Nagai Model for epithelial tissue development as the simulation software called *Epithelium*. The software is easy to install, takes up less than 50Mb, comes in parallel and non-parallel versions (Version 2.0.0, Version 1.0.0) and has very few dependencies. *Epithelium* allows users to specify all parameters of interest in easily modifiable text configuration files, and can generate data for animations of epithelial tissue development as well as useful plots of important variables.

The source code is highly modularized and allows for ambitious users to easily extend it to meet their needs. For example, alternate numerical integrators can easily replace the existing one, new mesh generators can replace the square mesh I have developed, and all data is output in space-separated formats which users with scripting language experience can transform to serve as input into the graphical utilities of their choice. In addition, the cell and coordinate classes are well documented and can be extended to output new data. Figure 2.1 provides a taste of what the most basic installation of *Epithelium* can do, showing a mesh of cells before and after equilibration.

2.2 Sample Configuration Files

The typical user will not want to modify source code, but would prefer to have a simple interface for changing simulation parameters. In this section we will explore in depth what the user interface looks like in terms of the three main configuration files, `config.txt`, `parameters.txt` and `change_mesh.txt`.

2.2.1 Simulation Settings, the Initial Mesh, and config.txt

In this file, the user can specify some global properties about the mesh, and some important quantities for how the simulation will proceed. The `dimension` of the mesh refers to the number of cells along a given axis in a square mesh, and the `max_step_size` is the largest

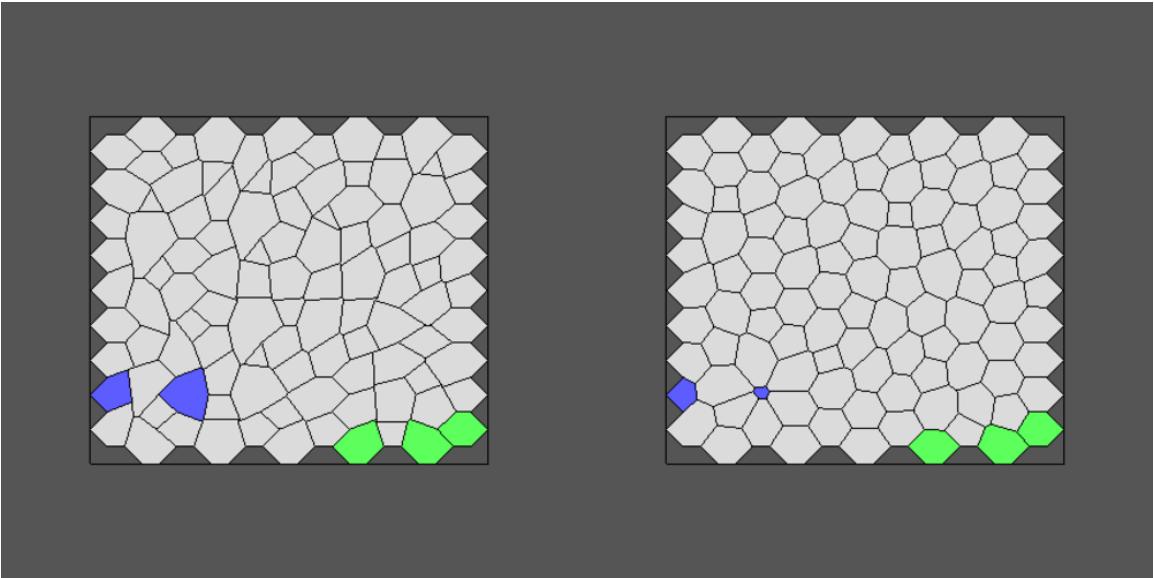


Figure 2.1: Cells before (left) and after (right) dynamics. The colored cells highlight cells with parameters different from the globally assigned values.

step the adaptive time stepping of the numerical integrator may take. OFF is the file format given to the plotting program **geomview** to plot the mesh and the user can specify how often the code prints an image with the **frequency** parameter, but it must be a multiple of ten. More closely spaced images can be generated by using a smaller integration step size. The **delta** parameter on line 5 specifies how close two vertices must be to force a T1 swap to occur.

The default setting of *Epithelium* creates initial conditions by drawing an equilibrium mesh of hexagons, and then perturbing it. The **swap length**, **upper bound**, and **max swaps** parameters are used for performing these random transformations, and if all are set to zero, the simulation will not alter the tissue. The **swap length** is the edge length below which a swap is performed, and the **max swaps** parameter defines the maximum number of random perturbations the code will make to the mesh before starting the simulation. The **upper bound** is an integer which is the upper bound of the range for a random number generator. A random number generator produces an integer in the range [1:**upper bound**], and a random T1 swap is performed to some edge of some cell when the random integer is ‘1’.

```
13 # Dimension of mesh MUST BE ODD!!!!
0.01 # Maximum step size
1000 # Number of iterations
```

```

10 # frequency of OFF file output. Must be a multiple 10!
.1 # delta minimum vertex separation.
1000 # max swaps
1.5 # swap length
2 # upper bound random number generator
1 # Make energy and shape plots? [1/0]
1 # Make a movie in the end? [1/0]

```

2.2.2 Global Cell Parameters

In the `parameters.txt` file, the user can specify the parameters discussed in Chapter 1. These are the default parameters for all of the cells in the tissue.

```

beta = 3;
lambda = 55;
t_gamma = 1;
t_area = 4.0;

```

2.2.3 Local Cell Parameters

While the parameters file allows global control of the mesh, the `change_mesh.txt` file allows users to change local properties of the mesh. The first line of the file is for specifying how many cells will have their γ , A_0 , λ , and β parameters changed. Note that P_0 cannot be changed, as P_0 is a function of A_0 . The subsequent lines are for specifying the cell index and new cell parameter value for each one of these modifications. As can be seen in Figure 2.1, cells can be color coded by parameter value to highlight these changes to the default settings.

```

2 3 1 1 # num gamma, num area , num lambda , num beta
16 4.0 # gamma
17 5.1 # gamma
3 1.0 # area
4 1.0 # area
10 3.7 # area
1 10 # lambda
90 100 # beta

```

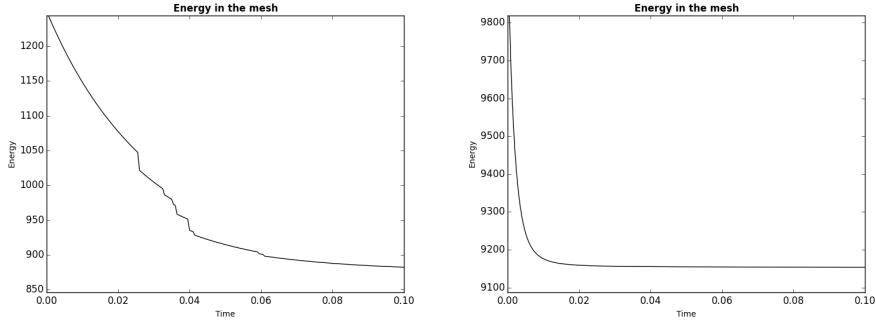


Figure 2.2: Energy graphs. Showing the decreasing energy in a mesh for two parameterizations. (left) $A_0 = 4.0; \beta = 0.0; \gamma = 1.0; \lambda = 5.0$. Many jumps due to inconsistencies that result from the T1 swaps. (right) $A_0 = 4.0; \beta = 5.0; \gamma = 1.0; \lambda = 1.0$. Energy decreases smoothly.

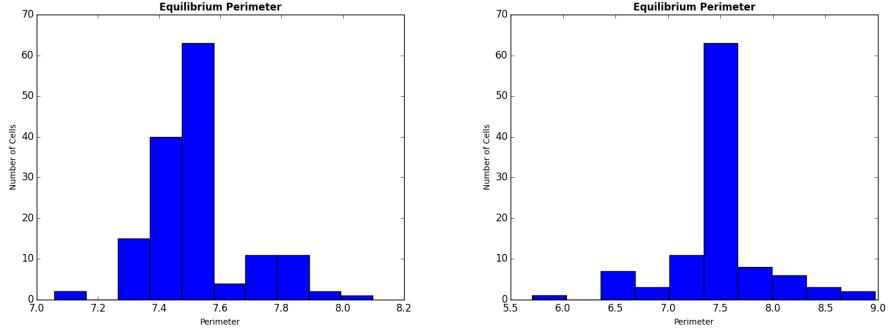


Figure 2.3: Equilibrium distribution of cell perimeters after dynamics. (left) $A_0 = 3.0; \beta = 1.0; \gamma = 3.0; \lambda = 3.0$. (right) $A_0 = 4.0; \beta = 5.0; \gamma = 1.0; \lambda = 1.0$.

2.3 Monitoring the Output of Epithelium

Model parameters are easy to change in *Epithelium*, and a variety of simulations are possible with very minimal effort on part of the user. Figure 2.2- 2.4 show the output from *Epithelium* for several parameterizations, including changes to the target area, and the β , γ , and λ values, and can be compared to the output of the original Honda-Nagai model in Figure 1.2. Each of the plots show the decreasing energy in the mesh over time and the equilibrium distributions of cell areas, perimeters, and shapes. There is very little variation between the plots - this is an essential property of the Honda-Nagai Model [37] that the introduction of cell proliferation and an unbounded mesh could change.

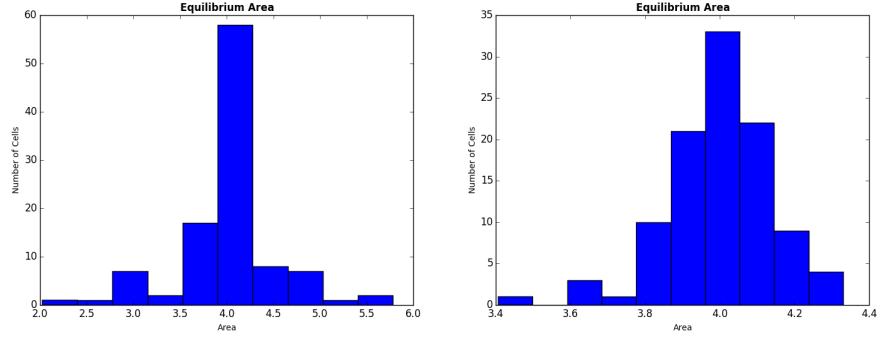


Figure 2.4: Equilibrium area after dynamics. (left) $A_0 = 4.0; \beta = 5.0; \gamma = 1.0; \lambda = 1.0$. (right) $A_0 = 4.0; \beta = 0.0; \gamma = 1.0; \lambda = 5.0$.

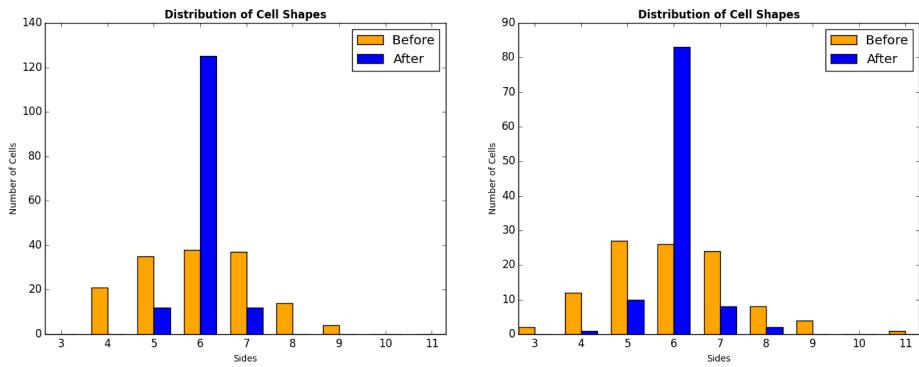


Figure 2.5: Equilibrium distribution of cell shapes after dynamics. (left) $A_0 = 3.0; \beta = 1.0; \gamma = 3.0; \lambda = 3.0$. (right) $A_0 = 4.0; \beta = 5.0; \gamma = 1.0; \lambda = 1.0$.

2.4 The Design of *Epithelium*

The technical details of how a vertex dynamics model can be effectively implemented are not explained in great depth in the literature¹. I will fill this intellectual void in the section that follows by presenting a detailed look at the data structures and algorithms needed for the programming of the Honda-Nagai Model. It is fitting to start with a very high-level overview of the structure of the code.

2.4.1 [Highly Simplified] Pseudocode

Here I will briefly outline how the code works. All of the functions are explained in some detail later in this chapter.

```
mesh_variables <- read_configs()
mesh <- make_mesh()
random_alterations(mesh)
copy(mesh, rotate_mesh)
rotate(rotate_mesh)
print(simulation_info) # So the user can verify all parameters.

for i = 1:num_iters
    if (iter%print_freq == 0)
        print(OFF_file)
    temp_mesh = NagaiHondaForce(mesh)
    temp_rotate_mesh = NagaiHondaForce(rotate_mesh)
    mesh <- mesh + temp_mesh
    rotate_mesh <- rotate_mesh + temp_rotate_mesh
    performT2(mesh)
    performT1(mesh)
    performT2(temp_rotate_mesh)
    performT1(rotate_mesh)
    rotate_back(rotate_mesh)
    compare_mesh(mesh, rotate_mesh)
    print(graphics and error analysis)
```

¹[17] is an exception, but is rather advanced.

2.4.2 Classes

Epithelium has a partially object oriented design. The cell and vertex classes organize the data into meaningful pieces.

- The **Cell** class contains a number of useful functions and data members to make the code easy to read and understand. All cell information could have been stored in arrays, but the OO structure makes the code more readable. All cells know their index, which vertices make them up, they are able to calculate their area and perimeter, can modify their constituent vertices, can tell you whether or not they contain a specific vertex, and can print out a graphical, color coded representation of themselves to an OFF file. For information about further functionality of the cell class, the reader is referred to the `cell.cpp` file in the source code. For information about downloading the source code, see appendix A.

```
public:  
    cell(int index, vector<int> AssociatedVertices, \  
          double target_area = t_area, double gamma = t_gamma)  
{  
    assert(index >= 0);  
    m_AssociatedVertices = AssociatedVertices;  
    m_index = index;  
    m_target_area = target_area;  
    m_target_perimeter = sqrt(pi * m_target_area);  
    m_gamma = gamma;  
}  
  
    cell(){} // Default constructor  
  
    vector<int>GetVertices(){return m_AssociatedVertices};  
    int GetIndex(){return m_index};  
    void SetIndex(int index){m_index = index};  
    void SetTargetArea(double area){m_target_area = area};  
    double GetTargetArea(){return m_target_area};  
    double GetTargetPerimeter(){return m_target_perimeter};  
    double ComputeArea(double * X, double * Y);  
    double ComputePerimeter(double * X, double * Y);
```

```

void PrintCell( ofstream &OffFile );
int ContainsVertex( int index );
void SetGamma( double gamma){m_gamma = gamma;};
double GetGamma(){ return m_gamma;};
void InsertVert( int v1, int v2 );
void EraseVert( int index )
{
    vector<int>::iterator it = find( m_AssociatedVertices , index );
    m_AssociatedVertices .erase( it );
};
void ReplaceVert( int before , int after )
{
    vector<int>::iterator it = find( m_AssociatedVertices , before );
    *it = after ;
};
void SetVertices( vector<int> vertices )
{
    m_AssociatedVertices = vertices ;
};
int GetNumSides(){ return m_AssociatedVertices .size ()};
private:
    vector<int> m_AssociatedVertices; // Stored counterclockwise
    int m_index;
    double m_target_area;
    double m_target_perimeter;
    double m_gamma;
};

```

- The **Vertex** class stores the index of a vertex, and whether or not the vertex will move during the integration. While a vertex object does not store the location of the vertex, it tells the simulator whether or not a vertex is on the border of a mesh and, if it is not, it is allowed to move. It was a design choice that *Epithelium* be able to run three popular types of meshes, those with border, those without, and those that cover a surface. Another benefit of the vertex class is that it allows users to easily extend the code to include forces acting on individual vertices, and to specify other types of

vertices besides interior and exterior. The vertex class could be extended to include a member function such as activeMigration(), or any number of interesting functions.

```
class vertex
{
public:
    vertex( int idx , bool t ) : index(idx) , IsInner(t){};
    vertex(){index = -1; IsInner = 0;};
    int index;
    bool IsInner;
    inline bool operator==(const vertex& rhs)
    {return index == rhs.index;};
};
```

2.5 Executive Decisions About Parameter Implementations

The P_0 and γ parameters have no explicitly stated value in the Honda-Nagai Model, and a number of reasonable choices about their implementation could be made.

Recall that the membrane surface energy term U_S is given by

$$U_S = \beta(P - P_0)^2 \quad (2.1)$$

where P is the cell perimeter, P_0 is a target perimeter, and β is some positive constant. Furthermore, note that the target perimeter is dependent upon the target area. There are several ways to assign a target perimeter $P_0(A_0)$ as a function of the target area. One obvious choice is to assume the cell tends to become a circle in the absence of resistance, and then solve a system using the equations for area and perimeter of a circle, giving $P_0 = 2\sqrt{\pi A_0}$. Another easy choice would be to assume the equilibrium epithelial cell shape is a hexagon, and then compute the target perimeter using the equations for the perimeter and area of a regular hexagon. The correct form for the target perimeter term is not specified in [37], so I take the circle approach, as done in [17].

The second choice that was forced upon us was the specification of the γ term in the cell-cell adhesion potential. Recall that cell-cell adhesion energy U_A is given by

$$U_A = \sum_{j=1}^n \gamma_j d_j \quad (2.2)$$

where γ is some constant depending upon the nature of the two cells interacting on a boundary. The γ term could have been implemented in various ways. I have chosen to assign a “stickiness” to each cell, and then the γ term is calculated as the average stickiness of two interacting cells. This is the approach taken in the molecular dynamics software CHARMM for handling pair interactions of atoms [9].

The γ parameter could also have been implemented as something of the form:

$$\gamma_{ij} = \begin{cases} c_0 & \text{if the cells } i \text{ and } j \text{ are of the same type} \\ c_1 & \text{if the cells are of compatible type} \\ c_2 & \text{if the cells are of incompatible type} \end{cases}$$

And the resulting dynamics might have been quite different. Neither form was explicitly stated in the original Honda-Nagai paper [37], so I had to make a choice about the implementation.

2.6 A Relational Database

There is one other major idea behind the design of *Epithelium*. A popular way to store data since the advent of the relational database is in groups of tables which are connected via *keys*. This type of database is popular for reasons which will become apparent by means of a simple example.

Consider a business which sells a number of products, and wants to keep track of their customers, the customer’s orders, the customer’s addresses, and information about the products ordered. A wasteful way to store this data is to create a large table in which the first column is for customer names. Next to every customer’s name is the customer’s address, and next to the customers address is the customer’s order number. Next to the customer’s order number is an item in that order, and next to each item ordered is the item information. This method of storing data is terribly redundant, because for each order with more than one item you would have the unnecessary storage of the orderid, and of all the customer information. Consider Figure 2.6a for an illustration of the concept. A better idea is to break this data up into several tables which together define a *schema*. By defining an appropriate schema, we can minimize the redundancy of information, and extract specific subsets of data in a cache-efficient way. Consider Figure 2.6b for an illustration of the concept.

Customer	Address	City	State	OrderID	Item	Item Info.
John	1 Main St	Pittsburg	PA	111	Chalk	For writing.
John	1 Main St	Pittsburg	PA	111	Book	For reading.
John	1 Main St	Pittsburg	PA	112	Chalk	For writing.
Mary	2 High St.	Columbus	OH	113	Chalk	For writing.
Mary	2 High St.	Columbus	OH	113	Paper	For drawing.

Customer	Address	City	State
John	1 Main St	Pittsburg	PA
Mary	2 High St.	Columbus	OH

OrderID	Customer
111	John
112	John
113	Mary

Item	Item Info.
Chalk	For writing.
Book	For reading.
Paper	For drawing.

OrderID	Item
111	Chalk
111	Book
112	Chalk
113	Chalk
113	Paper

Figure 2.6: An example of a relational database. Notice how implementation (a) requires more storage space than (b).

The *Epithelium* data structure is a schema made up of the `simulationCells`, `vertexList`, `X`, `Y`, `tempX` and `tempY` tables. The cell and vertex tables are implemented as 1D `std::vectors` of cell and vertex indices, whereas the position tables are implemented as low-level 1D arrays for ease of passing these structures to CUDA C functions in the parallel implementation of the software. The cells can extract coordinate information from the `vertexList` table via the *index* key, and the `coordinateList` can access the position information from the `X` and `Y` arrays via their own *index*. The temporary `X` and `Y` arrays store temporary position information about the vertices before the mesh positions are updated. This choice saves both memory and time because no data is stored redundantly. The coordinate information of a vertex is not stored in every cell that contains it, thus huge data structures need not be passed around when only a small portion of data is required. When updating vector locations, only the position and temporary positions are passed to a function instead of the entire mesh.

2.7 Initial Mesh Design

The `hex_mesh()` function generates an $n \times n$ mesh of cells, where the dimension represents the number of cells touching the boundary in either axial direction. Figure 2.8 shows the

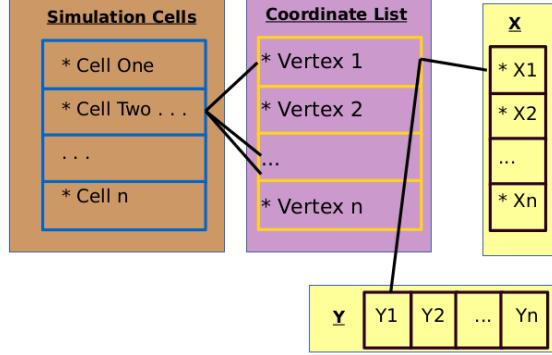


Figure 2.7: Data structure used for storing data in a non-redundant form.

default mesh used by *Epithelium*, with the cell ids and vertex indices labelled. After a sheet of cells is generated by `hex_mesh()`, the tissue is perturbed by performing random T1 swaps on edges of the mesh, and then by perturbing the locations of a select number of vertices.

2.8 Moving the Vertices

While the data structures behind *Epithelium* are important to understand, the most fundamental idea behind a vertex dynamics model is how the vertices in a tissue are moved. *Epithelium* transforms the tissue by looping over all of the vertices, computing the force acting on each vertex, and then computing a displacement with the Forward Euler Method (equation 2.3).

$$r_i^{t+1} = r_i + \Delta t \frac{dr_i}{dt} \quad (2.3)$$

In their original paper, H. Honda and T. Nagai described the use of a Modified Runge Kutta Method[37] to move the vertices, but this method would result in vastly more computations per each time step. A similar approach has been used in [48, 17].

A displacement is calculated and stored in the temporary X and Y arrays. No vertex is permitted to move more than one half of the minimum δ separating vertices (the δ under which a T1 swap will occur) during an integration. By imposing this restriction we are ensuring that we will not miss the event of two vertices coming critically close and a T1 swap occurring. Also, this prevents vertices from passing each other and invalidating the mesh. To ensure that no vertex moves too much, we verify each displacement as we put it in the temporary X and Y arrays. If a displacement is too large, then the entire array of

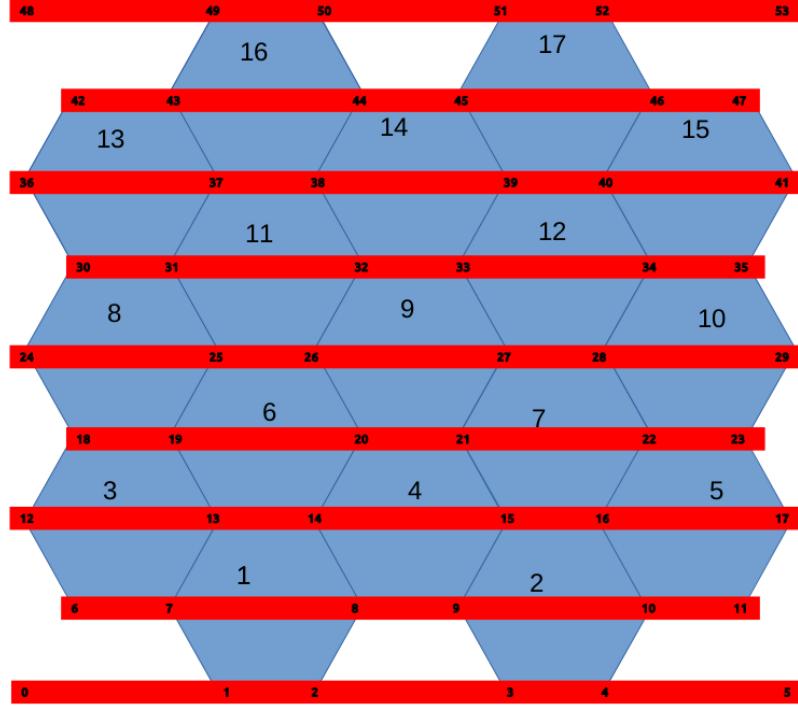


Figure 2.8: A typical mesh generated by *Epithelium*. A 5x5 hexagonal mesh is shown with cell indices written in the cells and vertex indices on red bars.

temporary displacements is erased, the time step is halved, and we begin the integration again. For this reason, we could label the integrator as ‘fault tolerant’.

Another important aspect of the numerical integration is that cell and vertex information must be processed in counterclockwise order. When we are integrating a vertex i , *Epithelium* first searches in the `simulationCells` vector for a cell which contains i , and then finds the next vertex $ip1$ in that cell’s `m_AssociatedVertices`². This step gives us an oriented edge. Some other cell contains that edge if it contains both of the vertices, and, if that cell exists, it must be clockwise from the first (See Figure 2.9). The cells are stored in the reverse order of which they are uncovered by this algorithm before the integration of the equation of motion for i starts. This is a very expensive step ($O(n^3)$) in the computation, and ought to be optimized.

2.9 Error Tolerance of the Algorithm

²`m_AssociatedVertices` is the container for the vertices making up a cell. See section 2.4.2.

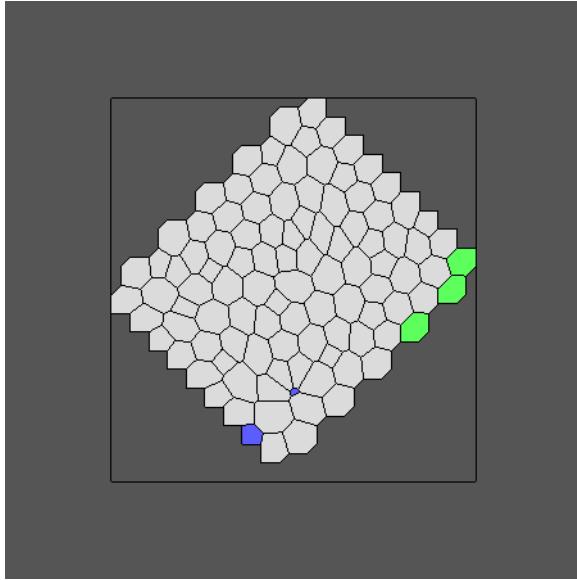


Figure 2.10: A rotated mesh for error analysis.

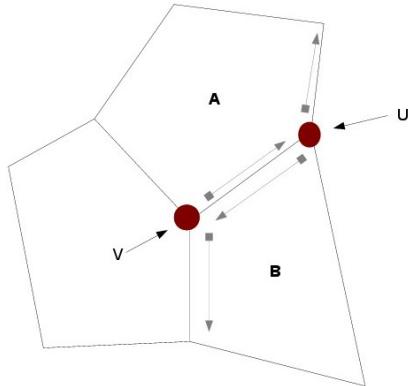
Epithelium outputs an error measurement at the end of a simulation to give the user a sense of the numerical stability of the code. *Epithelium* offers the user the option to run two simulations at the same time, one on the mesh made by `hex_mesh()`, and another on the same mesh which has been rotated 45 degrees (See Figure 2.10). In this way, we hope to expose any dynamics which are dependent upon the rounding of floating point numbers. Then, at the end of a simulation, the rotated mesh is rotated back and corresponding vertices are compared by index using the Euclidean norm. In practice a number near ϵ_{mach} is returned as the maximal Euclidean distance between corresponding vertices.

2.10 Embarassing Parallelism and CUDA

The numerical integrations and the vertex location updates exhibit what Cleve Moler describes as “embarrassing parallelism”. Computing the displacement of vertex a does not depend upon the computation of the displacement of vertex b during a given time step. Similarly, the vertex locations can all be updated in parallel since the update is simply a vector sum operation of the X and Y arrays with the temporary X and Y arrays, respectively. *Epithelium* employs several parallel routines for these types of operations when it is possible.

A popular hardware choice for parallel programming in last decade has been the NVIDIA GPU, and an extension of the C language called CUDA was developed to allow programmers

to send certain portions of C code the GPU for processing. In order to give the reader a better understanding of which portions of an epithelial modeling software can be effectively run on a GPU, a brief introduction to CUDA programming is necessary.



Algorithm for orienting cells counter-clockwise around vertex "V"

Figure 2.9: Getting Cells in Order

The NVIDIA CUDA-capable GPU is a collection of many small, weak processors packed very densely on a graphics card. The chips are so closely packed that each simple microprocessor has only a few registers in which to store memory, but all of the microprocessors have access to a larger global memory stored on the side of the card. One key to effective CUDA programming is to give the microprocessors small bits of data upon which they perform many calculations with minimal need to talk to global memory. Another fundamental trick to efficient GPU programming is to coalesce memory accesses so that neighboring processors grab data from nearby regions in global memory during every global memory transaction.

The CUDA hardware has been optimized such that when neighboring processors on the graphics card attempt to access memory addresses `arr[0]` and `arr[1]`, the transaction is essentially melded into one operation and performed smoothly. On the other hand, when two neighboring processors attempt to get data from `arr[0]` and `arr[100]`, the hardware strains to gather information from different sectors of the global memory and send these large chunks of memory to the local processors [32]. The demand that memory accesses be coalesced has been the single impeding factor in the full CUDA parallelization of *Epithelium*, for while the numerical integrations are *theoretically* parallelizable, data from several arrays (section ??) is used in calculating the force on a vertex.

Nevertheless, CUDA was employed for two obviously parallel tasks, and for one slightly more sophisticated one. The sum of the position arrays and the temporary position arrays was done with a simple vector sum, and the rotation of the mesh for error detection was performed by multiplying the coordinates of each vertex in the mesh by the rotation matrix:

$$\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

CUDA was also employed for the slightly challenging task of calculating the error between the non-rotated and rotated meshes by first subtracting the position arrays of the rotated mesh from the position arrays of the non-rotated mesh, at which point the calculated differences are then stored in dX and dY . Next, the pythagorean theorem is used to calculate the distances between the coordinates, and these lengths are stored in the *dist* array. The maximum distance is then extracted from this array in $\log(N)$ time by comparing neighboring elements and discarding the smaller one until only one element remains, giving the maximum $\| * \|_2$ error in the mesh.

2.11 Computing Topological changes.

Now that we have covered the way that the equation of motion is integrated, we move on to how the topological changes are implemented in *Epithelium*.

2.11.1 The T2 swap.

The `PerformT2s()` function looks at every triangular cell in the mesh and checks the edge lengths. If the cell has a critically small edge, then the cell is deleted from the mesh. The centroid of the cell is calculated, and this becomes the vertex representing the collapsed triangle. Then, one of the three constituent vertices has its position updated to the centroid coordinate and any cell which previously bordered the triangular element has both vertices related to the triangle deleted from `m_AssociatedVertices`. Finally, the centroidal vertex is inserted in the position of the deleted vertices. It is important to note that the `PerformT2s()` function must be called before the `PerformT1s()` function to make sure that all offending triangular elements are deleted before we perform the T1 swaps. The T1 swap requires four vertices to be performed correctly, as explained in the next section.

2.11.2 The T1 Swap

The `PerformT1s()` function loops over all of the cells in the mesh, and checks the edge lengths in each cell. If an edge is critically small (less than δ), then a T1 swap must occur. The first step taken by the `PerformT1s()` function is to find all of the cells and vertices involved in the swap.

Since the critically small edge was detected by looping over all the cells in the mesh

and checking their edges, we can identify the cell that initiated the swap (let's refer to it as $c1$). There is another cell in the mesh which contains the offending edge, and we can call that cell $c2$. Since $c1$ contains at least four vertices, then we can label the four vertices of interest, $im1$, i , $ip1$, $ip2$. i and $ip1$ are the vertices bounding the small edge, $im1$ is the vertex before i , and $ip2$ is the vertex coming after $ip1$ (figure 2.11).

Given these indices for the vertices, we can find the other cell in the mesh which contains $ip1$ and $ip2$, which we call $c3$. Finally, we locate the cell containing $im1$ and i , and call this cell $c4$. Now that all of the cells are located, we proceed to perform the T1 operation during the course of which $c1$ and $c2$ will cease to be neighbors and $c3$ and $c4$ will become neighbors.

The midpoint mp of the edge $(i, ip1)$ is calculated and a perpendicular bisector b is drawn through it. The centroid ($CN1$) of $c1$ is calculated and the vector pointing from m to $CN1$ defines the direction in which a new vertex should be placed. Next, the centroid ($CN2$) of $c2$ is calculated, and the vector pointing from the m to $CN2$ defines the direction in which the other new vertex should go. This could have been implemented in several ways, but in *Epithelium* $ip1$ is deleted from $c2$ and placed along b in the direction indicated by $CN1$ at a distance δ from the midpoint. Vertex i is deleted from $c1$ and is moved a distance δ along b in the direction indicated by $CN2$.

Before the function ends, i is inserted after $ip1$ in $c3$ and $ip1$ is inserted after i in $c4$. Figure 2.11 offers a visual aid to understanding the swap. The entire operation is easy to implement thanks to the counterclockwise storage of vertices.

2.12 Dependencies

Our discussion of *Epithelium* would not be complete without a discussion of software dependencies. In contrast to the large number of packages required to run other epithelial tissue simulators, the non-parallelized version of *Epithelium* requires only Python2.7, a C++11 or C++14 compiler, and BASH. Python and BASH handle some file I/O operations of the program, and are used to make plots and format data for the animations. Indeed, one could run *Epithelium* and generate OFF files of the tissue with only the specified C++ compiler installed, but several error messages would be generated as the program searches for the scripting languages.

In order to visualize the output of Epithelium you must have geomview and ImageMagick

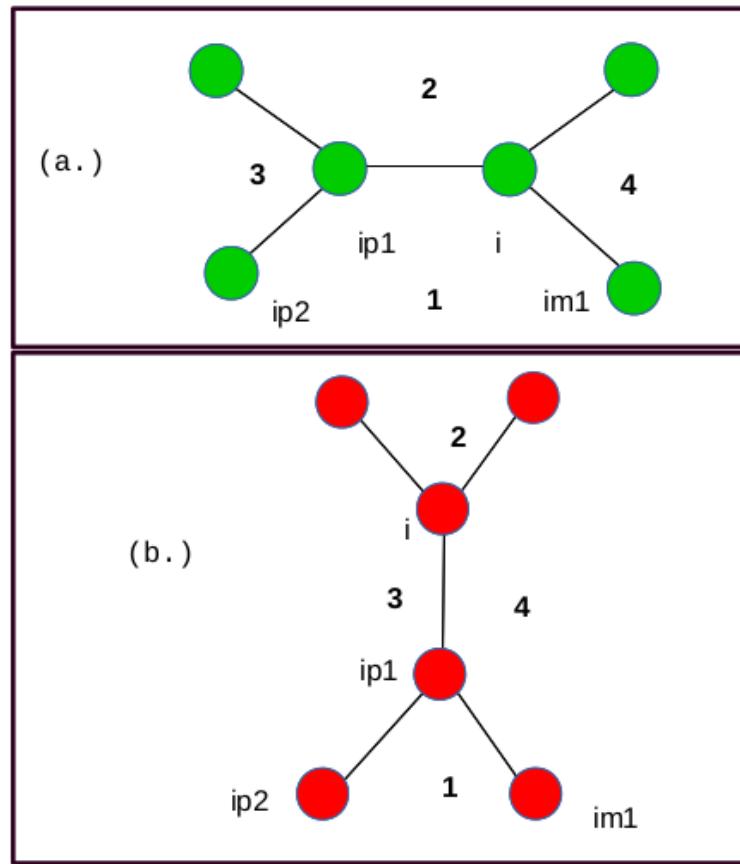


Figure 2.11: The implementation of a T1 swap. The vertices are labeled with indices corresponding to their local indices in cell $c1$. Notice how before the swap (top) $c1$ and $c2$ are neighbors, but after the swap $c3$ and $c4$ have become adjacent.

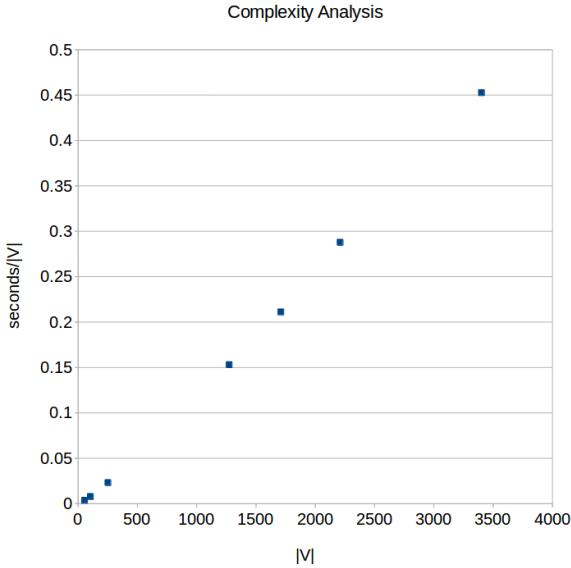


Figure 2.12: The runtime does not scale as the number of vertices.

installed to plot the meshes and to convert the .rgb image files to a .gif animation. All of these programs, languages, and compilers are free and so popular³ that they already installed by default on many Mac and Linux computers.

The parallel version of *Epithelium* has all of the above dependencies, plus the requirement of the nvcc CUDA compiler and a CUDA capable GPU.

2.13 Complexity

Wrapping up our discussion of *Epithelium*, we will look at the time complexity of the algorithm. We will consider the ideal case in which the step size and parameters have been chosen such that the step size will not need to be adapted, and we run the algorithm for 200 time steps. Then, we will divide the runtime by the number of vertices being moved.

For each vertex in the mesh, the displacement is calculated in constant time in the `AreaForce.hpp` and `PerAdhForce.hpp` files, and then each vertex is updated in constant time in the `MoveVerts()` function. The T1 swap will also be linear in the number of vertices, as the `PerformT1s()` function looks at each edge in the mesh twice (hence each vertex 6 times, since a vertex is connected to three edges) and will perform constant-time work on the vertex when an edge is too small. The T2 swap is linear in the number of cells, because

³Except geomview.

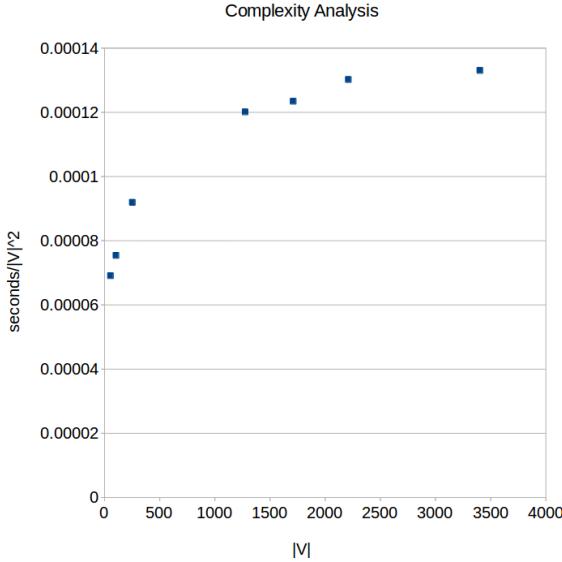


Figure 2.13: The runtime does not scale as $|V|^2$.

T2 swaps are rare, and the first step in the T2 swap is to check if a cell is a triangle. Given these facts, we think that the integration and position updates ought to scale with the number of vertices in the mesh.

To verify the complexity of the algorithm, we use an unperturbed mesh and run the simulation for various numbers of vertices. We make sure that the time step is never halved by choosing small parameter values and aborting whenever Δt is halved. Unfortunately, the runtime does not scale as the number of vertices — V —, likely due to the expensive memory transactions involved in the computations of displacements (Figure ??). A more efficient datastructure is needed, therefore, to parallelize *Epithelium* as well as to make the serial version more memory efficient.

Chapter 3: The Future of *Epithelium*

We have thusfar explored the nature of epithelial tissue, discussed the variety of models that are currently in use, and thoroughly examined the *Epithelium* implementation the Honda-Nagai Model. In this chapter, we will discuss some ideas for future epithelial tissue models.

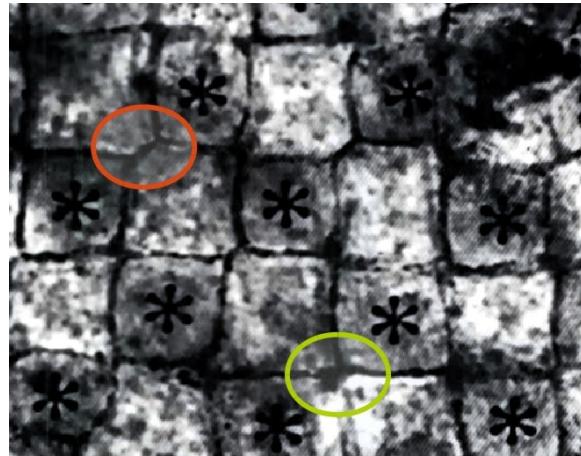


Figure 3.1: Square cells in quail epithelium. Image courtesy of [59].

3.1 Rosette Models

We have discussed the standard topology of epithelial tissue in great detail, but now we will explore a potential model with much different connectivity. Most models of epithelial tissue exclusively support vertices of degree three, but there is at least one tissue for which this topology may not be accurate. Figure 3.1 is taken from a paper by H. Yamanaka and H. Honda [59] in which they model the dynamics of Japanese quail epithelium using exclusively degree three vertices. The fact is, however, that Figure 3.1 lends itself equally to interpretation as a mesh of degree four vertices or degree three vertices. It seems intuitively unlikely that the epithelial tissue of Japanese quails be the only tissue with this appearance, and a reasonable research idea is to find what other tissues have this property, and then subject meshes of this type to Honda-Nagai forces to see if the mesh remains a stable

quadrilateral mesh as in nature. This type of experimentation could help resolve whether or not the Honda-Nagai Model is a good model for all types of tissue, or if it is strictly suited to tissues with predominantly six sided cells in equilibrium. In its current incarnation *Epithelium* can only handle three cells meeting at a vertex, but more energy could be invested to make it handle different connectivities.

3.2 A New Potential, A New Force

In the Honda-Nagai Model, a harmonic form is given for the potential energies due to area and perimeter deformations, the gradient of which gives a hookean form¹ for the force applied to a node (for more details, refer to 1.3). The assumptions underlying a model can render even the most seemingly accurate model physically meaningless if the assumptions are proven to be incorrect, and for this reason we should be skeptical about the harmonic formulation. In the literature the $C(x - x_0)^2$ form for the potential energy is asserted without biological evidence that it is correct, or even reasonable. Furthermore, assuming that cells are indeed elastic, there is no reason to believe that their elasticity fits the linear stress-strain curves prescribed by the hookean model.

There is a large class of materials including rubber and various elastomers that are classified as neo-hookean hyperelastic materials because their stress-strain² curves are only linear in a small region (See Figure 3.2). After this region, the elasticity of the material becomes much higher [7, 34]. Hyperelastic models have been applied to various tissues with impressive results as in [42, 28], but have only been briefly discussed with respect to epithelial tissue, as in [60, 11].

The ideas of hyperelasticity need to be explored in more detail, and the conditions under which the Honda-Nagai elastic model is plausible ought to be better examined.

3.3 Improvements in Imaging

Another recent development in epithelial tissue science is the ability to record high quality videos of epithelial tissue during morphogenesis. One of the most impressive examples to date is the work being done in the Kiehart lab at Duke University where they are recording

¹Hooke's law says that $F = -k\Delta x$.

²The stress-strain curve is the curve showing the amount of deformation(strain) in a material due to the application of varying amounts of force (stress).

the closing of the epithelial envelope of *Drosophila* embryos [53]. Through their recordings they have produced very good fits for certain modeling parameters and have performed an interesting experiment in which they try to keep the envelope from closing, but it closes anyway due to some currently unknown mechanism.

An equally promising development in the field of epithelial tissue imaging is the work being done in the surface reconstruction of *Drosophila* wings. Several softwares exist which can reconstruct a mesh of epithelial cells from an input image, but the most recent software released by L. Bai is able to take sets of images and faithfully reconstruct the apical face of a sheet of wing cells [3], even detecting 3D deformations.

These imaging softwares will provide immense amounts of data about epithelial cell shapes and topology which will guide the development of future models, and perhaps serve as initial conditions and benchmarks for existing models. Unfortunately, the source code for the 3D imaging software is not freely available, and the possibility of developing an open source version ought to be considered. Very little is known about the accuracy of these softwares, as information about them comes exclusively from academic journal articles, and, depending on the quality of these imaging packages, competitive programs may be able to be developed in a short amount of time thanks to advanced libraries like Python's OpenCV [47].

3.4 Further Parallelization

While the preceding ideas are very interesting and would contribute new ideas to the world of epithelial tissue modeling, the project of recreating the Honda-Nagai model was originally undertaken in the hopes of parallelizing the vast majority of the computations. Unfortunately, the trouble of efficiently storing a large tissue rendered the implementation of a parallel integrator seemingly impossible.

The Chaste software is able to perform some computations in parallel, but only on the

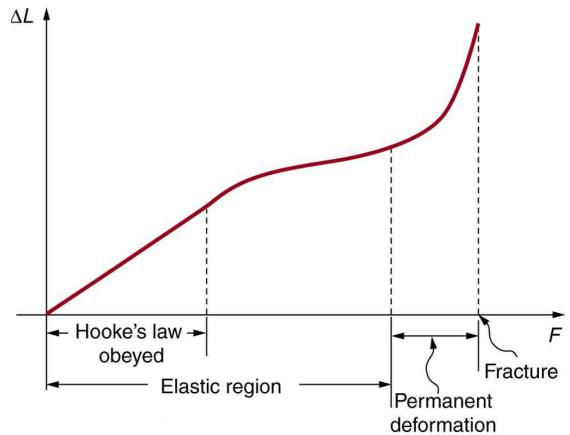


Figure 3.2: The stress-strain curve for rubber. Image courtesy of [6].

scale of a multi-core CPU, and not yet on the scale of a CUDA GPU [10]. It is still possible to make a great contribution to the epithelial tissue community by discovering and sharing an efficient datastructure and set of algorithms for storing and moving a sheet of epithelial cell vertices in parallel.

The first task in fully parallelizing the *Epithelium*'s numerical integrations is storing the data in C arrays instead of classes, which will inevitably make the code's workings much more subtle than they currently are. Then, the next objective will be situating the data in an array in such that processors can effectively grab coalesced data chunks without too many expensive memory transactions.

3.5 Voronoi Tesselations

The Honda-Nagai model ³ uses a periodic Voronoi Tessellation as the initial condition for the tissue [37], yet to the best of our knowledge an open source periodic Voronoi Tessellation program is unavailable. Indeed, the Voronoi Tessellation provides an ideal artificial initial mesh for an epithelial tissue simulation since all vertices have degree three and a careful choice of generating points can give all cells similar areas and perimeters. Several sources offer reliable software which computes a Voronoi Tessellation on the unbounded plane [?, 55], while another freely available code performs the Voronoi Tessellation on 3D surfaces [56], but none of these codes can perform the tessellation in a bounded box, or with periodic boundary conditions.

The C++ Boost Library is a popular, heavily tested library for the C++ language, yet the Boost Voronoi Tessellation only computes the unbounded Voronoi Tessellation of a set of points, despite demand for a bounded Voronoi Tessellation as evidenced by questions [27] on the programming website StackOverflow, and by the needs of epithelial modeling software. The lowest hanging fruit in the field of Voronoi Tessellations is the design of a bounding box algorithm for the Boost Voronoi Tessellation. A good second project would be to modify the Boost Voronoi functions so that they are able to produce periodic meshes. The expansion of this library would be a good service to modelers and to the C++ community in general.

³And other programs. The Voronoi Tessellation is one of the most famous algorithms in all of computational geometry. For more information see [2]

3.6 Visualization Software

A final suggestion for the improvement of *Epithelium*, and data presentation in general, is the development of an easy to use mesh visualization software which will take in data in a format similar to the OFF file format, and output an image of a mesh with highly customizable appearance. The vast majority of epithelial tissue simulations have rather bland graphical output, while others are very sharp and highly customizable. For examples of the current state of tissue visualization, see Figure 1.2.

Bibliography

- [1] Amber Molecular Dynamics. ambermd.org
- [2] D. Austin. Voronoi diagrams and a day at the beach. <http://www.ams.org/samplings/feature-column/fcarc-voronoi>
- [3] L. Bai et. al. 3D surface reconstruction and visualization of the Drosophila wing imaginal disk at cellular resolution. [Awaiting publication]
- [4] K. Bambardekar et. al. Direct Laser Manipulation Reveals the Mechanics of Cell Contacts In Vivo *PNAS* **112** 1416-1421.
- [5] The C++ Boost Library. www.boost.org
- [6] Rubber stress-strain. boundless.com
- [7] M. Boyce. Constitutive models of rubber elasticity: a review. http://biomechanics.stanford.edu/me338/me338_project01.pdf
- [8] Buchmann, A. Alber, M., Zartman, J. *Sizing it up: The mechanical feedback hypothesis of organ growth regulation.* Seminars in Cell and Developmental Biology, 2014.
- [9] CHARMM tutorial. http://www.charmmtutorial.org/index.php/The_Energy_Function
- [10] Chaste Tutorial https://chaste.cs.ox.ac.uk/chaste/tutorials/release_2.1/UserTutorials.html
- [11] V. Conte and JJ Munoz. A 3D finite element model of ventral furrow invagination in the Drosophila melanogaster embryo. *J Mech Behav Biomed Mater.* **2** (188-198).
- [12] Drasdo, D. *Bucking Instabilities of One Layered Growing Tissues.* Physical Review Letters 84.
- [13] Durand, M., Stone, H. Relaxation Time of the Topological T1 process in a Two Dimensional Foam. arXiv.0

- [14] *Epithelial Tissues* http://www.botany.uwc.ac.za/sci_ed/grade10/mammal/epithelial.htm
- [15] Farhadifar, R., Roper, J., Algouy, B., Eaton, S., Julicher, F. The Influence of Cell Mechanics, Cell-Cell Interactions, and Proliferation on Epithelial Packing. *Current Biology* **17** 2095-2104. (2007)
- [16] Fletcher, A. , Osterfield, M., Baker, R., Shvartsman, S. *Vertex Models of Epithelial Morphogenesis*. Biophysical Journal 106, June 2014.
- [17] Fletcher, A., Osborne, J., Maini, P, Gavaghan, D. *Implementing vertex dynamics models of cell populations in biology within a consistent computational framework*. Prog. Biophys. Mol. Biol. 113: 299 - 326.
- [18] R. A. Fort and M. S. Steinberg. The differential adhesion hypothesis: a direct evaluation. *Dev. Biol***278** 255-263. (2005)
- [19] The Giant's Causeway Image. http://images.nationalgeographic.com/wpf/media-live/photos/000/009/cache/giants-causeway_974_990x742.jpg
- [20] Gibson, W., Gibson, M. *Cell Topology, Geometry, and Morphogenesis in Proliferating Epithelia*. Current Topics in Developmental Biology **89** 87-114. (2009)
- [21] T. Gillies and C. Cabernard. Cell Division and Orientation in Animals. *Current Biology* **21** 599-609
- [22] P. Grassia, C. Oguey, R. Saepitomi. Relaxation of the topological T1 process in a two-dimensional foam. *The European Physical Journal E* **35** (2012)
- [23] H. Honda. Description of Cellular Patterns by Dirichlet Domains: The Two-Dimensional Case. *Journal of Theoretical Biology* **72** 523-543. (1978)
- [24] H. Honda, H. Yamanaka, M. Dan-Sohkawa. A Computer Simulation of Geometrical Configurations During Cell Division. *Journal of Theoretical Biology* **106** 423-435. (1984)
- [25] H. Honda. A three-dimensional vertex dynamics cell model of space-filling polyhedra simulating cell behavior in a cell aggregate. *JTB* **226** 439-453 (2004)
- [26] Honda, H. Essence of Shape Formation in Animals. *Forma*, **27** S1-S8. (2012)

- [27] Stackoverflow.org “How to construct a voronoi diagram inside a box?”
- [28] F. Julicher et. al. Ective behavior of the cytoskeleton. *Nonequilibrium physics.***449** (3-28)
- [29] K. Kawasaki, T. Nagai, K. Nakashima. Vertex models for two-dimensional grain growth. *Philisophical Magazine Part B* **60** 399 - 421. (1989)
- [30] LAMMPS Tutorials. <http://lammps.sandia.gov/tutorials.html>
- [31] K. Liu, S. Ernst, V. Lecaudey, O. Ronneberger. Epithelial Rosette Detection in Microscopic Images.
- [32] Luebke, D. and Owens, J. *Intro to Parallel Programming* <https://www.udacity.com/course/cs344> 2013.
- [33] Marshall et. al. What Determines Cell Size? *BMC Biology* **10** (2012)
- [34] A. Muhr. Modeling the Stress-Strain Behavior of Rubber. *Rubber Chemistry and Technology* **78**(391-425.)
- [35] Y. Mao. Differential proliferation rates generate patterns of mechanical tension that orient tissue growth. *The EMBO Journal* **21** 2790-2803. (2013).
- [36] T. Nagai and H. Honda. Wound Healing Mechanism in Epithelial Tissues Cell Adhesion to Basal Lamina. *Proceedings of the 2006 WSEAS Int. Conf. on Cellular & Molecular Biology, Biophysics & Bioengineering***2006** (pp111-116)
- [37] Nagai, T. Honda, H. A dynamic model for the formation of epithelial tissues. *Philisophical Magazine, Pt. B.***81**(2001)
- [38] T. Nagai, K. Kawasaki, K. Nakamura. Vertex dynamics models of two-dimensional cellular patterns. *Journal of Physical Sciences Japan* **57** 2221-2224.
- [39] R. Nagpal, A. Patel, M. Gibson. Epithelial Topology. *BioEssays* **30** 260-266. (2008)
- [40] K. Nakashima, T. Nagai, K. Kawasaki. Scaling Behavior of Two-Dimensional Domain Growth: Computer Simulation of Vertex Models. *Journal of Statistical Physics* **57** 759-787. (1989)

- [41] NAMD Scalable Molecular Dynamics. <http://www.ks.uiuc.edu/Research/namd/>
- [42] A. Natali. Hyperelastic models for the analysis of soft tissue mechanics: definition of the constitutive parameters. *Biomed. Rob. and BioMech* **2006**
- [43] Ohlenbusch, H.M., Aste, T. Dubertret, B., Rivier, N. The Topological Structure of 2D Disordered Cellular Structures. arXiv.
- [44] S.Okuda et. al. Reversible Network Reconnection Model for Simulating Large Deformation in Dynamic Tissue Morphogenesis. *Biomech Model Mechanobiol* **12** 627-644 (2013)
- [45] S. Okuda et. al. Coupling intercellular molecular signalling with multicellular deformation for simulating three-dimensional tissue morphogenesis. *Interface Focus* **5** (2015)
- [46] S. Okuda et. al.
- [47] Open Computer Vision. opencv.org.
- [48] G. Oster and M. Weliky. The mechanical basis of cell rearrangement. *Development* **109** 373-386. (1990)
- [49] <http://www.millerplace.k12.ny.us/webpages/lmiller/photos/636532/Epithelial%20TissueTypes.bmp>
- [50] J. Pease and J. Tirnauer. Mitotic spindle misorientation in cancer - out of alignment and into the fire. *J. of Cell Science* **124** (1007-1016).
- [51] K. Ragkousi and M. Gibson. Cell Division and the Maintenance of Epithelial Order. *Journal of Cell Biology* **207** 181-188
- [52] Restrepo, S., Zartman, J. , and Basler, K. *Coordination of Patterning and Growth by the Morphogen DPP* Current Biology 24, 245- 255
- [53] A. Sokolow et. al. Cell Ingression and Apical Shape Oscillations during Dorsal Closure in *Drosophila*. **102** 969-979.
- [54] R. Thom. Topological Models in Biology. *Topology* **8** 313- 315 (1969)
- [55] Triangle: A two dimensional quality mesh generator. <https://www.cs.cmu.edu/~quake/triangle.html>

- [56] Voro++ - A 3D Voronoi Cell Software Library. <http://math.lbl.gov/voro++/>
- [57] Weaire, D. and Rivier, N. *Soap, Cells, and Statistics*
- [58] F. Xiong et. al. Interplay of Cell Shape and Division Orientation Promotes Robust Morphogenesis of Developing Epithelia. *Cell* **2014** 415-427. (2014)
- [59] H. Yamanaka and H. Honda. A checkerboard pattern manifested by the oviduct epithelium of the Japanese Quail. *Int. J. Dev. Biol.* **34** 377-383.
- [60] J. Zartman. Unit operations of tissue development: epithelial folding. *Annu. Rev. Chem. Biomol. Eng.* **2010**

Appendix A: Getting, Running, and Modifying the Code

A.1 GitHub

When you are working on a large project such as this one, it is a good idea to have some sort of version control system which tracks the changes you have made to your code, and to return to an earlier working version in case something gets terribly broken. Many people who do not know about version control will do just this, except they will save as' every couple of days. Unfortunately, this method is very space inefficient, as each time you 'save as', you save your entire project. Roughly speaking, git saves only the small changes you have made between versions. Git is a popular version control tool, and github is a popular place to store your files online.

The following instructions show you how to create and clone git repositories. The repository for *GrowFlesh* can be found at <https://github.com/J JulianCienfuegos/NAGAIHONDAMODEL>.

Get Your Files on GitHub

- Go to github.com and sign up for an account
- Make a new repo on github
- cd to the directory of your project on your machine.
- git init
- git add .
- git commit -m "some message"
- git remote add origin YOUR URL HERE (This url is given to you from github when you make the repo.)
- git pull origin master
- git push origin master

Access And Modify These Files Somewhere Else

- Simply type `git clone urlofrepositoryhere` into a terminal
- Work on project
- `git push origin master`, when done working.