The Ohio State University

Epithelium

The Lightweight, Customizable, Epithelial Tissue Simulator

Author:
Melvyn Ian Drag
B.A. Mathematics

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 13, 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 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. The code has very few dependencies, and these dependencies are likely already satisfied by the default packages installed on a Linux or Mac computer. In addition, 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 results of past papers, and for quickly designing new computational experiments.

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 recevied 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 endebted 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

Contents

List of Figures

Chapter 1

Epithelial Tissue and Vertex Dynamics

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 [19]; epithelial tissue is our interface with the outside world.

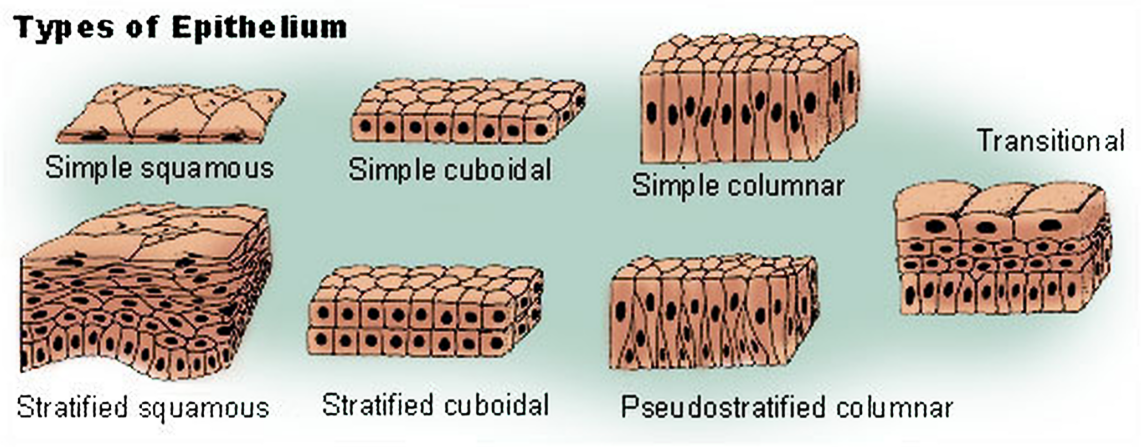

Figure 1.1: **The Types of Epithelial Tissue. Image credit [8].**

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 diffussion 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 abraision.

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 1.2 (a.) and (b.). For an example of a 2D epithelial simulation on a three dimensional surface, see Figure 1.2 (d.). The 3D simulation of stratified tissue is more difficult to implement than a two dimensional model[1]. For an example of one successful 3D model, see Figure 1.2(c.).

Current modeling is producing great results in the field of epithelial tissue morphogenisis, equilibration, and wound healing. The Honda-Nagai model, which we will discuss in great detail below, successfully reproduced the wound healing of cats' corneas [25]. This model has also been able to reproduce all of the essential dynamics of epithelial tissue [26]. Current imaging tools have enabled the recording of epithelial tissue dynamics *in vivo* [**?**] [42], 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 [21] for simulating atomistic materials, and CHARMM [4], Amber [1], and NAMD [30] for the molecular dynamics 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 [11]. In this thesis I will present the basic ideas of **vertex dynamics models**, and then describe the implementation of one of them as a freely available modeling tool for the community.

[1]Even a leading epithelial tissue simulator, Chaste [11], still does not have stable 3D modeling capabilities.

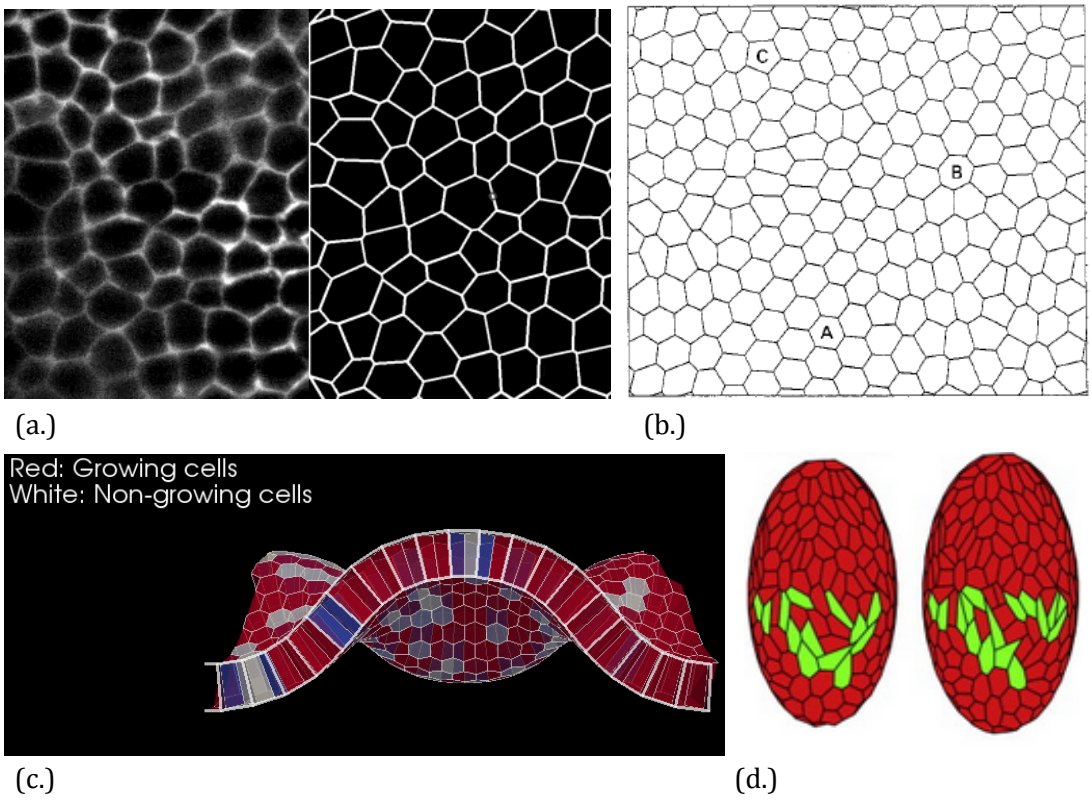

Figure 1.2: **(a.) A comparison of living tissue to a simulation [2]. (b.) A diagram from the original Honda-Nagai paper [26]. (c.) A 2d mesh of cells deforming in three dimensions [32]. (d.) A tissue developing on a surface [10].**

1.1 Modeling Epithelial Tissue

A two dimensional **vertex dynamics model** of epithelial tissue is made up of vertices and edges [17] which bound cells. 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, and this force is applied to all of the vertices in the mesh 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 [9], whereas others have accurately reproduced the dynamics of corneal wound healing [25]. In other research, simulations have faithfully captured the

effects of laser perturbations to epithelial cell junctions [2], and others have quantified parameters which are important in describing the formation of the epithelial envelope in *Drosophila* [39]. 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 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 [34]. 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.

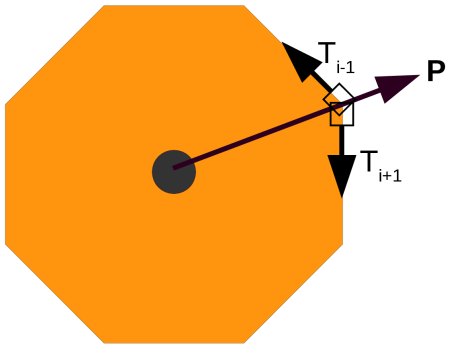

Figure 1.3: **The Weliky-Oster Force.**

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 [27]. In the fantastic review paper *Soap, Cells, and Statistics*, 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 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. Image courtesy of [13].**

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

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

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

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

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

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 stores 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 [26].

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 a force used by another scientist [9][2].

1.2 The Nagai-Honda Model

1.2.1 How the Vertices Move

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

$$\eta \frac{dr_i}{dt} = F_i \tag{1.5}$$

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 [20].

Based upon the notion described in the preceding section, that biological cells move in a way quite similar to crystallites at high temperature[3], the Honda-Nagai model has this equation of motion as its basis. The force on the right hand side of the equation is in turn defined as the gradient of a energy function, since the model presupposes the principle of maximum entropy is the guiding principle behind epithelial cell equiliibria. Then, 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:

$$C(x - x_0)^2 \tag{1.6}$$

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 it will be, and the more forcefully it will want to return to equilibrium. See Figure 1.5.

The last 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

[2]Which suggests that the basic formulation is quite acceptable to physicists.

[3]Often referred to as grain growth

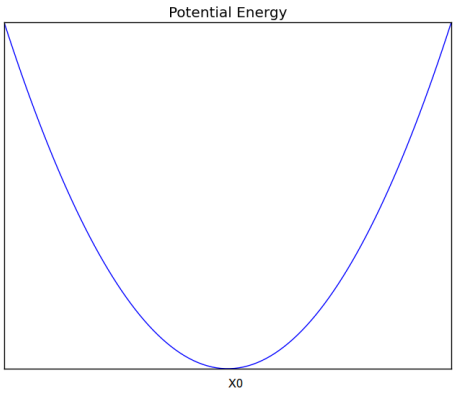

Figure 1.5: Potential energy as a function of distance from equilibrium.

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 membranes of these cells [12].

The precise statement of the potential energies discussed above are the following:

1. The deformation energy term U_D is given by

$$U_D = \lambda(A - A_0)^2 \tag{1.7}$$

   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

$$U_S = \beta(P - P_0)^2 \tag{1.8}$$

   where P is the cell perimiter, P_0 is a target perimeter, and β is some positive constant. 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 wants to become a circle, 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 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 [26], so I take the circle approach, as done in [11].

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

$$U_A = \sum_{j=1}^{n} \gamma_j d_j \tag{1.9}$$

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. Also take note of the fact that the γ term could be 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 [4].

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

$$\gamma_{ij} = \begin{cases} 0 & \text{if the cells } i \text{ and } j \text{ are of the same type} \\ 1 & \text{if the cells are of compatible type} \\ -1 & \text{if the cells are of incompatible type} \end{cases}$$

And the resulting dynamics might be quite different. Neither form was explicitly stated in the original Nagai-Honda paper [26], so I had to make a choice about the implementation.

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

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

As seen in [11], the negative gradient of this potential energy is:

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

where l is the lth 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 [11].

The area of a cell is given by Gauss's Shoelace Formula:

$$A = \frac{1}{2} \sum_{i=1}^{N} \Big(x_i y_{i+1} - x_{i+1} y_i \Big) \tag{1.11}$$

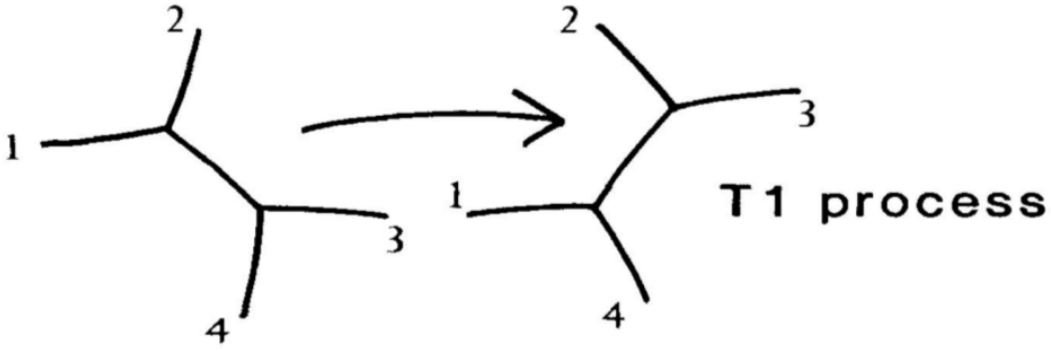

Figure 1.6: **A T1 Swap. Two neighboring cells are no longer neighbors after the swap. Image courtesy of** [41]

where $N + 1 := 1$ Therefore, the gradient is given by:

$$\nabla_i A_l = \frac{1}{2}\Big(y^l_{I+1} - y^l_{I-1},\ \ x^l_{I-1} - x^l_{I+1}\Big) \tag{1.12}$$

where the superscripts 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:

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

Therefore

$$\nabla_i P = \nabla_i d_{i-1} + \nabla_i d_i \tag{1.14}$$

and

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

Substituting the above values into the equation:

$$-\nabla_i U = -\nabla_i (U_D + U_S + U_A) = F_i \tag{1.16}$$

gives the force described in equation **??**.

1.2.2 Topological Changes to the Mesh

There is emprirical evidence that nearly all vertices in a sheet of epithelial tissue have coordination number three (most vertices have three incident edges) [28]. Since the coordination

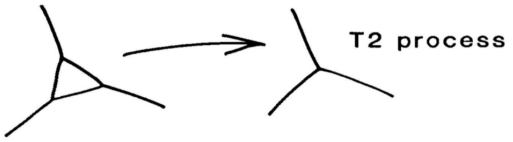

Figure 1.7: **A T2 Swap. Image courtesy of** [41]

number of almost all vertices is three, this 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 the connectivity [41]. As it turns out there are three changes which can occur, called the T1, T2 and T3 swaps, and the original Honda-Nagai Model implements the first two.

The T1 swap, illustrated in Figure **??**, is also called a "neighbor exchanging swap" because two cells that were adjacent cease to be neighbors and two cells that weren't 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. 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 a triangular cell becomes too small and is deleted and replaced by a single vertex. See Figure **??**.

1.2.3 Selection of Parameters

The basic vertex dynamics model requires the programmer 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 [26]. 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 convert these parameters to the simpler ones [33].

Table 1
Table of parameter values used in the simulation shown in Figs. 10 and 11.

Parameter	Description	Value	Dimensions	Reference
d_{min}	Cell rearrangement threshold	0.1	Length	–
η	Drag coefficient	1.0	Time (Length)$^{-1}$	(Nagai and Honda, 2006)
λ	Deformation energy coefficient	55	Force (Length)$^{-3}$	(Nagai and Honda, 2006)
A_0	Mature cell target area	1	(Length)^2	–
β	Membrane surface energy coefficient	0	Force (Length)$^{-1}$	(Nagai and Honda, 2006)
γ_{cell}	Cell–cell adhesion energy coefficient	5	Force	(Nagai and Honda, 2006)
$\gamma_{boundary}$	Cell-boundary adhesion energy coefficient	10	Force	(Nagai and Honda, 2006)
ϕ	Contact inhibition threshold	0.9	Non-dimensional	–
T_{cycle}	Cell cycle duration	$U[10, 14]$	Time	(Meineke et al., 2001)
$T_{mitosis}$	Mitosis duration	1	Time	(Meineke et al., 2001)
Δt	Time step	0.001	Time	–

Figure 1.8: **Parameter values as specified in [11].**

Interestingly, I haved found very few explicit statements of the parameters used in simulation (exceptions are [25, 11, 33], see Figure 1.8). Still, even in these cases, the physical significance of the parameterizations chosen is not stated. Very recently, new imaging techniques has permited the *in vivo* observation of epithelial tissue morphogenesis [4] [39, 42]. This will likely open new doors for the correct parameterization of current models, or for the reformulation of the expressions for forces and potential energies.

In the case of the Honda-Nagai Model, however, there is little difference between equilibrium states attributed to various parameter choices (See Chapter **??**). One of the parameter-independent defining characteristics of the Honda-Nagai model is the strong tendency toward six-sided cells in equilibrium, as shown in Figure 1.9. Still, 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 [14]. For example, drosophila wings, with their highly oriented divisions, have been shown to contain approximately 80% hexagonal cells whereas simulations of of tissues with purely stochastic divisions converge to approximately 47% hexagons [28]. 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 [41].

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

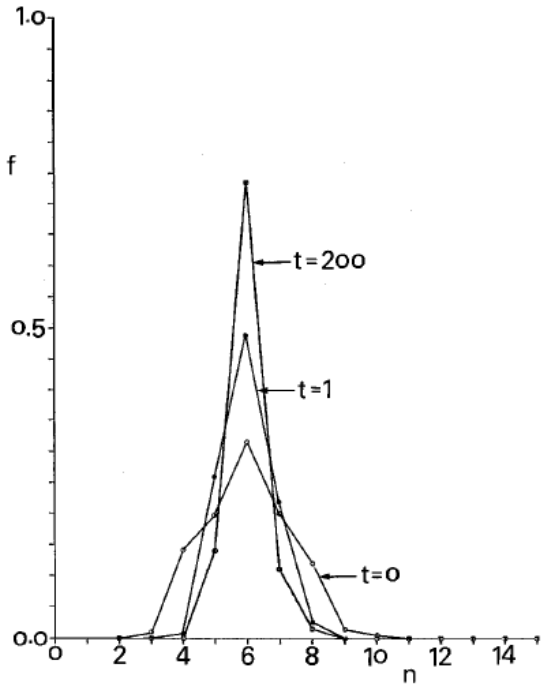

Figure 1.9: **The distribution of cell shapes as a function of time in the original Honda-Nagai Model. Image courtesy of** [26]

1.3 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.

1.3.1 Similar Models of Potential Energy

Interestingly, as mentioned in the section 1.1, the Honda-Nagai form for the energy in a vertex is quite similar to the form developed by Farhadifar [9]. The Farhadifar formulation is:

$$E_i = \sum_{cells} \frac{K}{2}(A - A_0)^2 + \sum_{edge} \gamma_{edge} d_{edge} + \sum_{\alpha} \frac{\beta}{2} P_\alpha^2 \tag{1.17}$$

Remember the formulation of the Honda-Nagai energy:

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

The equations are nearly identical, except that the Farhadifar model asserts that the lowest energy for a cell is the energy in which its perimeter is zero. Nevertheless, the topological results of this model are not wildly different from the Honda-Nagai results [9].

1.3.2 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 [26] that dealt with the equilibration of a fixed number 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 unanimouslly accepted theory) about how the dvision 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 [37] [15]. 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 [28, 36].

1.3.3 The Euler Characeteristic and Its Implications

The majority or 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[28, 14]. In this section I will expand upon some observations made in [41] 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:

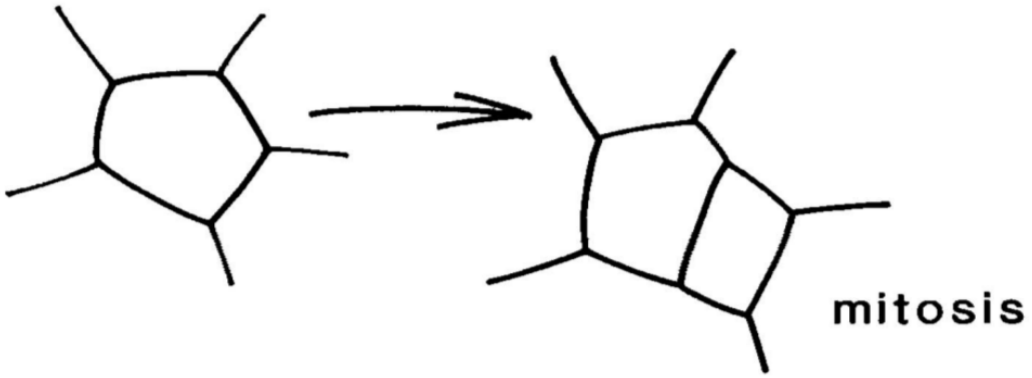

Figure 1.10: **The T3 Swap. Image courtesy of [41].**

$$\chi = V - F + E \tag{1.18}$$

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. Inititally, our intuition tells us that there should be three times as many edges as vertices, which leads us to the incorrect formula:

$$3V = E \tag{1.19}$$

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:

$$3V = 2E \tag{1.20}$$

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 per shape. More clearly, we might guess the following:

$$\sum_{k=3}^{N} kF_k = E \tag{1.21}$$

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

$$\sum_{k=3}^{N} kF_k = 2E \tag{1.22}$$

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

$$V - F + E = \chi \tag{1.23}$$

$$\frac{2E}{3} - F + E = \chi \tag{1.24}$$

$$\frac{5E}{3} - F = \chi \tag{1.25}$$

$$\frac{\sum_{k=3}^{N} kF_k}{6} - F = \chi \tag{1.26}$$

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

Biological cells are very small, and an epithelial tissue is composed of many cells, so we assume that $F \to \infty$ 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.

So we know that the algebraic mean of the number of vertices per face must be 6. 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 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 [22] 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.

1.3.4 Okuda Reconnection

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, can handle simulations of arbitrarily large size, 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 the code 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, as users may need. Figure 2.1 provides a taste of the 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`.

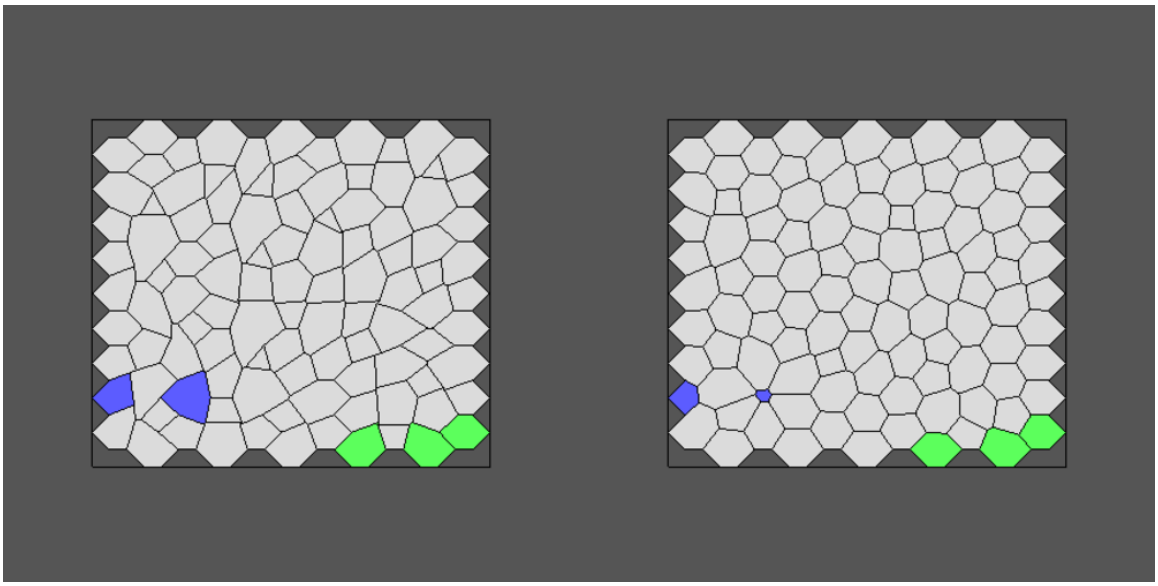

Figure 2.1: Cells before (left) and after (right) the application of force.

2.2.1 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. Most of the quantities are self explanatory. The `dimension` of the mesh refers to the number of cells along a given axis in a square mesh, and the `swap length`, `upper bound`, and `max swaps` parameters are used for performing random transformations to the mesh in the beginning of the simulation. The swap length is the edge length below which a swap is performed. The `max swaps` parameter is 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 upper bound of the range for a random number generator. A random number generator produces a number in the range [1:upper bound], and a random T1 swap is performed on '1'. The `delta` parameter specifies how close two vertices must be to force a T1 swap to occur.

OFF is the file format given to the plotting program **geomview** to plot the mesh. You 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.

```
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 # Make a movie in the end? [1/0]
```

2.2.2 parameters.txt

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 change_mesh.txt

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 γ, A_0, λ, and β modifications will be made to the mesh. Note that P_0 cannot be changed, as P_0 is a function of A_0 (See Chapter 1). The subsequent lines are for specifying the index and new value for each one of these modifications. As can be seen in Figure 2.1, cells can be color coded by parameter value to show 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
```

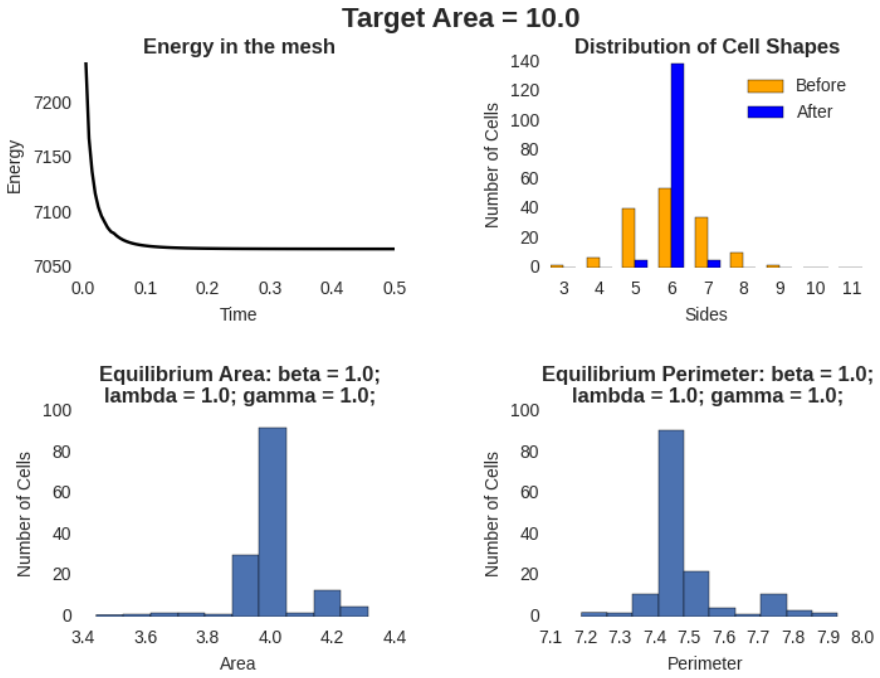

Figure 2.2: **Simulation output**

2.3 Image Gallery

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 - Figure 2.4 show the output from *Epithelium* for several parameterizations, 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, the equilibrium distribution of cell areas, perimeters, and shapes. You may notice that there is very little variation between the plots - this is an essential property of the Honda-Nagai Model [26] that the introduction of cell proliferation and an unbounded mesh could change.

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 literature[1]. I will fill the void in this field in the section that follows and present a detailed look at the data structures and algorithms needed for

[1] [11] is an exception, but is rather advanced

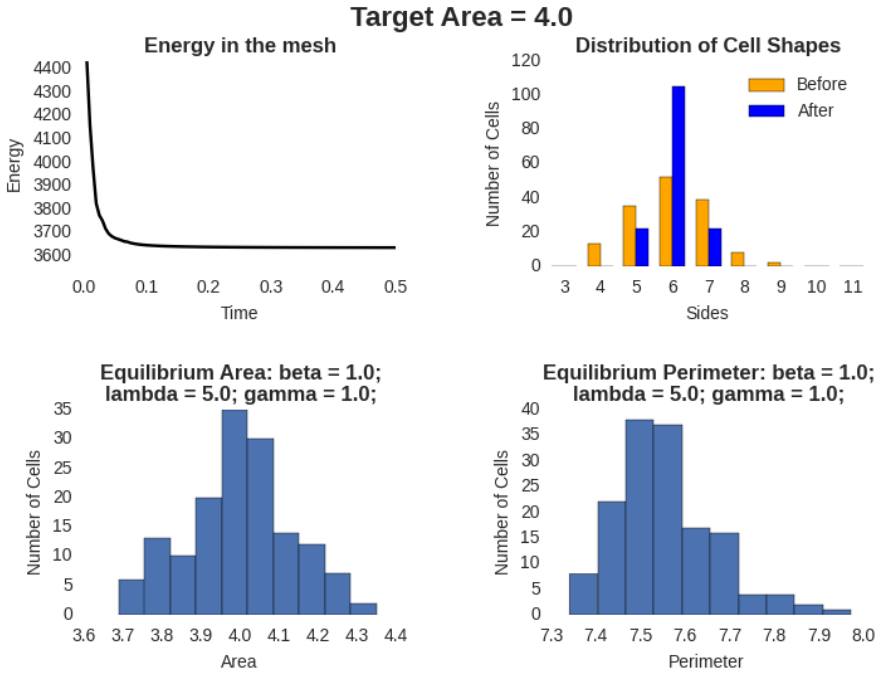

Figure 2.3: **Simulation output**

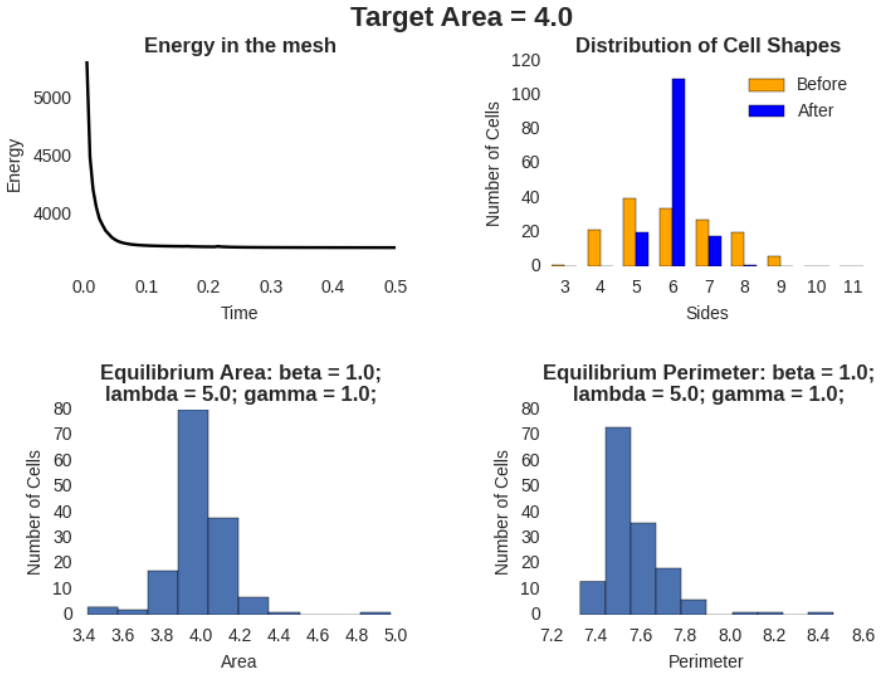

Figure 2.4: **Simulation output**

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)
```

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 calulate their area and perimeter, can modify their constituent vertices, can tell you whether or not ther contain a 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 3.5.

```
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 is 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. This code was developed with eyes to the future.

```
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 A Relational Database

There is one other major idea behind the design of *Epithelium*. A popular way to store data since the 1970's is to store data 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 **??**(a.) 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 **??**(b.) for an illustration of the concept.

The *Epithelium* datastructure 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. 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

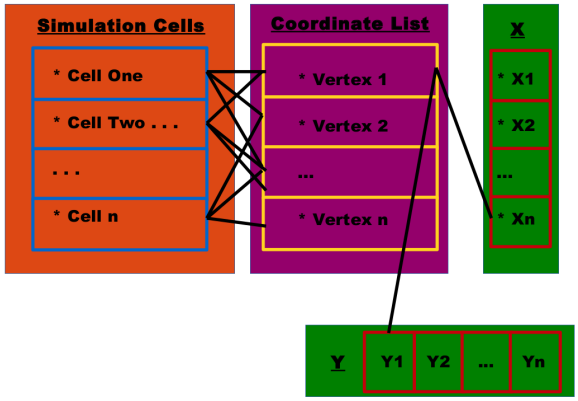

Figure 2.5: **The Relational Database.**

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 (e.g. The coordinate information of a vertex is not stored in every cell that contains it) and huge data structures do not need to be passed around when only a small portion of data is required (e.g. When updating vector locations, only the position and temporary positions are passed to a function instead of the entire mesh).

2.6 Initial Mesh Design

The hex_mesh() function generates an nxn mesh of cells, where the dimension represents the number of cells touching the boundary in either axial direction. Figure 2.6 shows the 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.

2.7 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. In their original paper, H. Honda and T. Nagai described the use of a Modified Runge Kutta Method[26] to move the vertices, but this method would result in vastly more computations

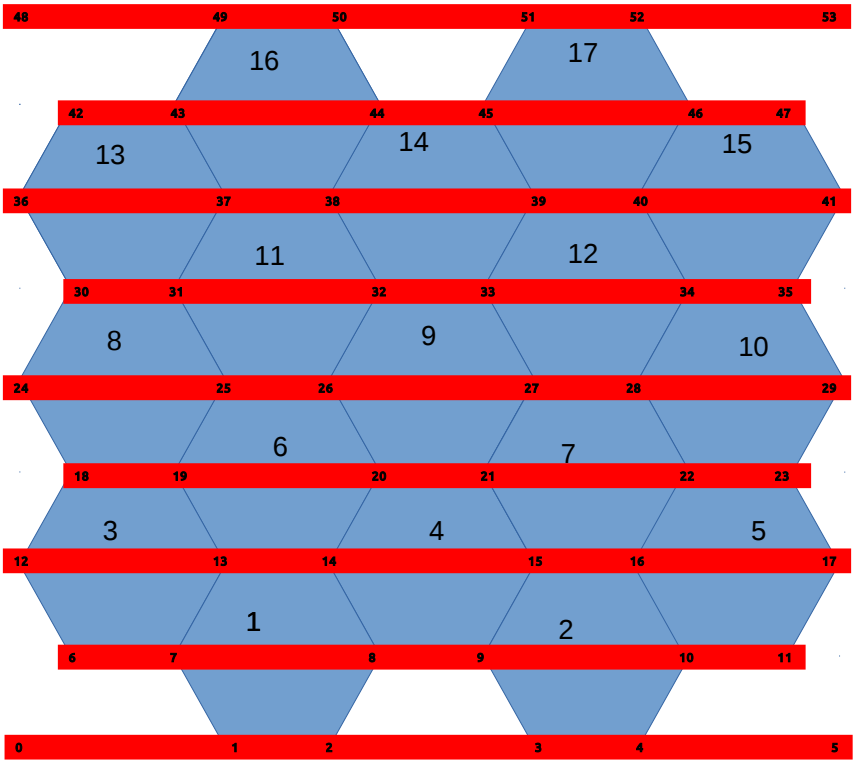

Figure 2.6: **A 5x5 hexagonal mesh. The cell indices are written in the cells and the vertex indices are written on the red bars.**

per each time step. The same decision to use then Euler Method was made by the research group at Oxford that developed CHASTE, as well as by the Weliky-Oster team [34], and so we accept it as a reasonable design decision.

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 delta separating vertices (the δ under which a T1 swap will occur) during an integration. By imposing this restiction we are ensuring that we will not miss the event of two vertices coming critcally 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 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'.

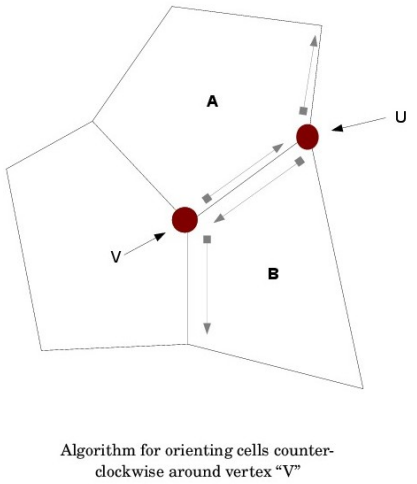

Figure 2.7: Getting Cells in Order

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 the finds the next vertex $ip1$ in that cell's `m_AssociatedVertices`. 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 2.7). 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.

2.8 Error Tolerance of the Algorithm

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* runs two simulations at the same

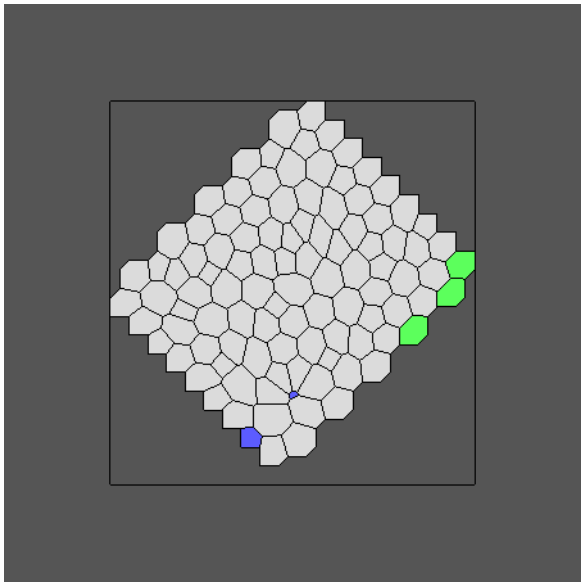

Figure 2.8: **A Rotated Mesh for Error Analysis.**

time, one on the mesh made by hex_mesh(), and another on the same mesh which has been rotated 45 degrees (See Figure 2.8). In this way, we hope to expose any dynamics which are dependent on 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 we see near ϵ_{mach} error for every simulation.

2.9 Embarassing Parallelism and CUDA

The numerical integrations and the vertex location updates exhibit what Cleve Moler describes as "embarassing 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. The GPU is a collection of small processors with very little shared memory, and is well suited so simple computations that

can be performed on memory stored in adjacent locations in C arrays. The demand that memory 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 (See Section **??**) is used in calculating the force on a vertex.

Nevertheless, CUDA was employed for two obvious parallel tasks, and for one slightly 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 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. 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.

2.10 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*.

2.10.1 The implementation of 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`. Next, the centroidal vertex is inserted in the position of the deleted vertices. 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, because these cannot be performed on triangular cells. The T1 swap requires four vertices to be performed correctly, as explained in the next section.

2.10.2 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 is 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 $ip1$.

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. Then 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 of δ from the midpoint. 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 2.9 offers a visual aid to understanding the swap. The entire operation is easy to implement thanks to the counterclockwise storage of vertices.

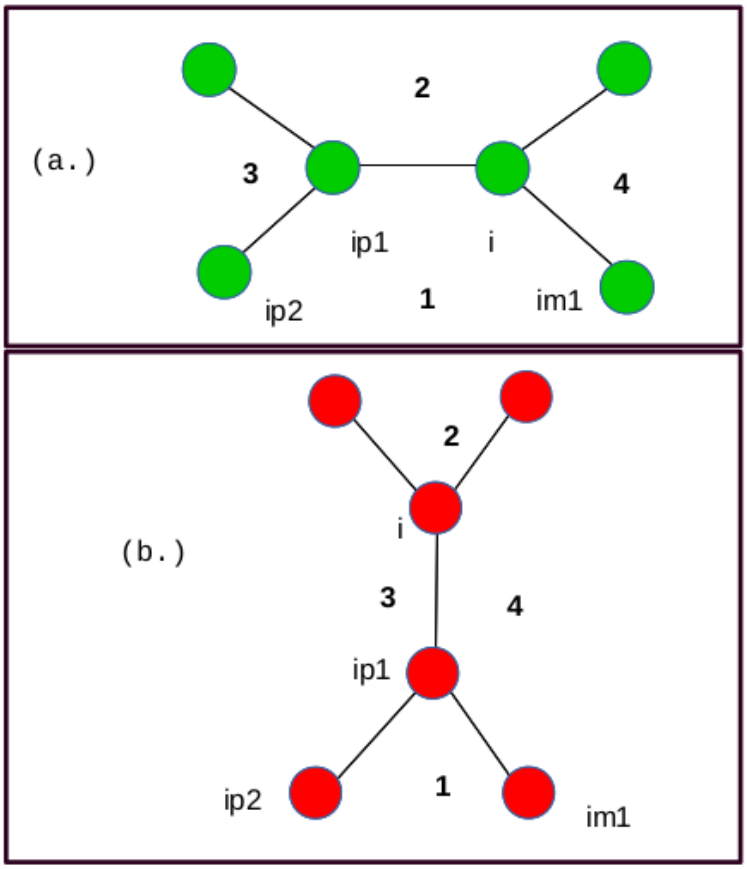

Figure 2.9: **The implementation of a T1 swap.**

Chapter 3

Advances

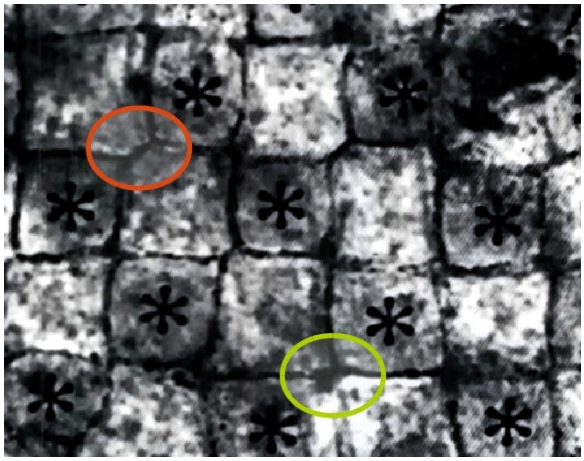

Figure 3.1: Square Cells in Quail Epithelium

An image or Japanese quail epithelium. This tissue is patterned like a checkerboard and has been modeled in the past as a tissue with degree three vertices. It is possible that modelling this tissue as degree four vertices would provide better approximations of the dynamics. In particular, notice that the orange vertices clearly can be modeled with degree three, wherease the green vertex could be one degree four vertex or two degree three.[43]

What happens to the system if we change the vertices to degree 4? How will the equilibria compare? Perhaps a new model force will becom apparent?

3.1 Further Parallelization

Data Redundancy

3.2 A New Potential, A New Force

Change the constant to $\beta(P)$, a function of the perimeter. After differentiating, and using the chain rule, what can be said about the appearance of the new force?

3.3 Images

Imaging input initial condition

3.4 Periodic Voronoi Mesh

C++ doesnt have a bounding box algorithm for the voronoi mesh, or a periodic voronoi algorithm. These would be valuable contributions to the C++ Boost libraries.

3.5 Visualization SOftware

That is standard, and a step above Geomview.

Bibliography

[1] Amber Molecular Dynamics. ambermd.org

[2] K. Bambardekar et. al. Direct Laser Manipulation Reveals the Mechanics of Cell Contacts In Vivo *PNAS* **112** 1416-1421.

[3] Buchmann, A. Alber, M., Zartman, J. *Sizing it up: The mechanical feedback hypothesis of organ growth regulation.* Seminars in Cell and Developmental Biology, 2014.

[4] CHARMM tutorial. http://www.charmmtutorial.org/index.php/The_Energy_Function

[5] *Chaste Tutorial* https://chaste.cs.ox.ac.uk/chaste/tutorials/release_2.1/UserTutorials.html

[6] Drasdo, D. *Bucking Instabilities of One Layered Growing Tissues.* Physical Review Letters 84.

[7] Durand, M., Stone, H. Relaxation Time of the Topological T1 process in a Two Dimensional Foam. arXiv.0

[8] *Epithelial Tissues* http://www.botany.uwc.ac.za/sci_ed/grade10/mammal/epithelial.htm

[9] Farhadifar, R., Roper, J., Algouy, B., Eaton, S., Jullcher, F. The Influence of Cell Mechanics, Cell-Cell Interactions, and Proliferation on Epithelial Packing. *Current Biology* **17** 2095-2104. (2007)

[10] Fletcher, A. , Osterfield, M., Baker, R., Shvartsman, S. *Vertex Models of Epithelial Morphogenesis.* Biophysical Journal 106, June 2014.

[11] Fletcher, A., Osborne, J., Maini, P, Gavaghan, D. *Implementing vertex dynamics models of cell populations in biology within a consitent computational framework.* Prog. Biophys. Mol. Biol. 113: 299 - 326.

[12] R. A. Fort and M. S. Steinberg. The differential adhesion hypothesis: a direct evaluation. *Dev. Biol***278** 255-263. (2005)

[13] The Giant's Causeway Image. `http://images.nationalgeographic.com/wpf/media-live/photos/000/009/cache/giants-causeway_974_990x742.jpg`

[14] Gibson, W., Gibson, M. *Cell Topology, Geometry, and Morphogenesis in Proliferating Epithelia.* Current Topics in Developmental Biology **89** 87-114. (2009)

[15] T. Gillies and C. Cabernard. Cell Division and Orientation in Animals. *Current Biology* **21** 599-609

[16] P. Grassia, C. Oguey, R. Saepitomi. Relaxation of the topological T1 process in a two-dimensional foam. *The European Physical Journal E* **35** (2012)

[17] H. Honda. Description of Cellular Patterns by Dirichlet Domains: The Two-Dimensional Case. *Journal of Theoretical Biology* **72** 523-543. (1978)

[18] H. Honda, H. Yamanaka, M. Dan-Sohkawa. A Computer Simulation of Geometrical Configurations During Cell Division. *Journal of Theoretical Biology* **106** 423-435. (1984)

[19] Honda, H. Essence of Shape Formation in Animals. *Forma*, **27** S1-S8. (2012)

[20] K. Kawasaki, T. Nagai, K. Nakashima. Vertex models for two-dimensional grain growth. *Philisophical Magazine Part B* **60** 399 - 421. (1989)

[21] LAMMPS Tutorials. `http://lammps.sandia.gov/tutorials.html`

[22] K. Liu, S. Ernst, V. Lecaudey, O. Ronneberger. Epithelial Rosette Detection in Microscopic Images.

[23] Luebke, D. and Owens, J. *Intro to Parallel Programming* `https://www.udacity.com/course/cs344` 2013.

[24] Marshall et. al. What Determines Cell Size? *BMC Biology* **10** (2012)

[25] 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)

[26] Nagai, T. Honda, H. A dynamic model for the formation of epithelial tissues. *Philisophical Magazine, Pt. B.***81**(2001)

[27] T. Nagai, K. Kawasaki, K. Nakamura. Vertex dynamics models of two-dimensional cellular patterns. *Journal of Physical Sciences Japan* **57** 2221-2224.

[28] R. Nagpal, A. Patel, M. Gibson. Epithelial Topology. *BioEssays* **30** 260-266. (2008)

[29] 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)

[30] NAMD Scalable Molecular Dynamics. `http://www.ks.uiuc.edu/Research/namd/`

[31] Ohlenbusch, H.M., Aste, T. Dubertret, B., Rivier, N. The Topological Structure of 2D Disordered Cellular Structures. arXiv.

[32] S.Okuda et. al. Reversible Network Reconnection Model for Simulating Large Deformation in Dynamic Tissue Morphogenesis. *Biomech Model Mechanobiol* **12** 627-644 (2013)

[33] S. Okuda et. al. Coupling intercellular molecular signalling with multicellular deformation for simulating three-dimensional tissue morphogensis. *Interface Focus* **5** (2015)

[34] G. Oster and M. Weliky. The mechanical basis of cell rearrangement. *Development* 109 373-386. (1990)

[35] `http://www.millerplace.k12.ny.us/webpages/lmiller/photos/636532/Epithelial\%20TissueTypes.bmp`

[36] J. Pease and J. Tirnauer. Mitotic spindle misorientation in cancer - out of alignment and into the fire. *J. of Cell Science* **124** (1007-1016).

[37] K. Ragkousi and M. Gibson. Cell Division and the Maintenance of Epithelial Order. *Journal of Cell Biology* **207** 181-188

[38] Restrepo, S., Zartman, J. , and Basler, K. *Coordination of Patterning and Growth by the Morphogen DPP* Current Biology 24, 245- 255

[39] A. Sokolow et. al. Cell Ingression and Apical Shape Oscillations during Dorsal Closure in *Drosophila*. **102** 969-979.

[40] R. Thom. Topological Models in Biology. *Topology* **8** 313- 315 (1969)

[41] Weaire, D. and Rivier, N. *Soap, Cells, and Statistics*

[42] F. Xiong et. al. Interplay of Cell Shape and Division Orientation Promotes RObust Morphogenesis of Developing Epithelia. *Cell* 2014 415-427. (2014)

[43] H. Yamanaka and H. Honda. A checkerboard pattern manifested by the oviduct epithelium of the Japanese Quail. *Int. J. Dev. Biol.* **34** 377-383.

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 terrribly 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 you 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/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.