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 11, 2015

Abstract

Epithelial tissue performs many important functions, 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. There are several codes in existence which reproduce certain aspects of epithelial tissue morphogenesis, wound healing, and equilibration, but only one of them are open source projects available to the public. Unfortunately, this code is fairly difficult to install and simulations require a fair amount of patience to design. 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 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; which was an integral part of my development as a public speaker. In general, I couldn't have had an advisor who was more energetic, more encouraging, and dedicated to providing me with all the tools I needed to perform great work. I also have to thank the members of the Sotomayor lab for attending my presentations, and for being such great company. I also have to thank Dr. E. Overman. Dr. Overman also 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 for this paper in your hands. 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

Modeling

Epithelial tissue covers the interior and exterior surfaces of our bodies. Skin, the lining of the esophogas and intestines, the urethra, the lining of the lungs and all of the bronchioles in the lungs 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 [**?**]; epithelial tissue is our interface with the outside world.

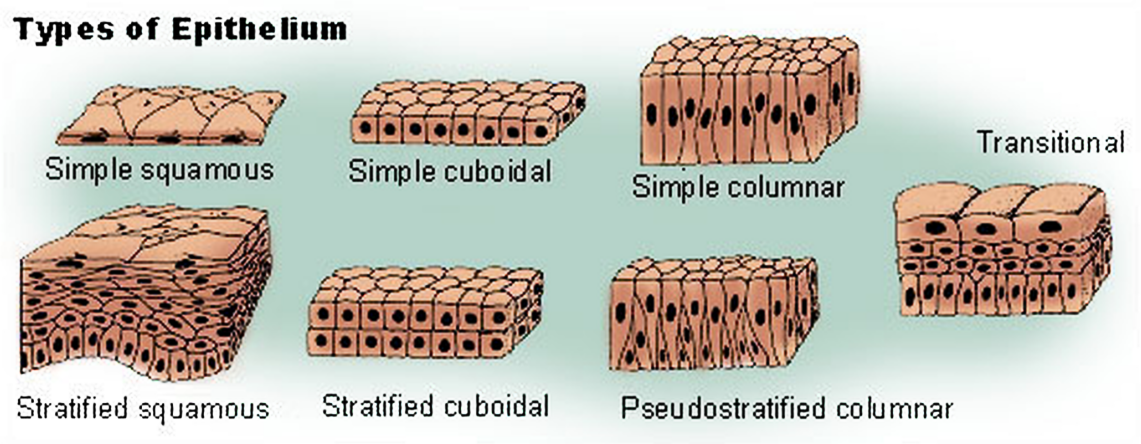


Figure 1.1: The Types of Epithelial Tissue

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 all very flat, much flatter than they are wide. These cells are therefore well suited to allow diffussion across themselves. As such, 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

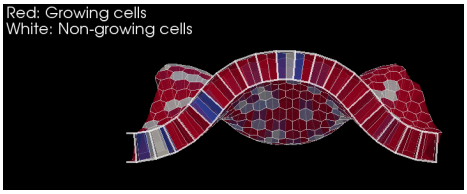
food. Stratified squamous epithelia are several layers thick line the esophogas and mouth and serve to protect against abraision.

What all of these tissues have in common, however, is how amenable they are to computational modeling. The simplest case is that of 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, since the top and bottom of the cells move in tandem and the surface where two cells touch can be approximated by a line. See Figures CITE FIGURES HERE to see examples of current 2D models. Slightly more difficult is the modeling of stratified tissue. In this case, the tissue develops in three dimensions, since underlying cells affect the cells on top of them CITE OKUDA IMAGE FOLLOWING DR. OVERMANS HELP WITH IMAGE FORMATTING. These cannot be modeled in two dimensions, but can be modeled by a solid composed of three dimensional polytopes. The theory behind these models is well developed, but the techincal details of implementing such a model has made it so that very exist, and are often quite limited [1]. For an example of a 3D model, see image INSERT IMAGE NAME HERE.

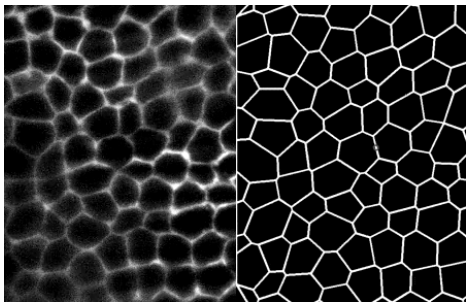
Current modeling is producing great results in the field of epithelial tissue morphogenisis, equilibration, and wound healing. The Honda-Nagai model which we will discuss in great detail in this paper successfully reproduced the wound healing of cats' corneas[22]. This model has also been able to reproduce all of the essential dynamics of epithelial tissue [23]. Current imaging tools have enabled the recording of epithelial tissue dynamics *in vivo* [**?**][37], providing a wealth of experimental data which can serve as either initial conditions for simulations, or as benchmarks to measure the success of computational models. 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 for simulating atomistic materials, and CHARMM, Amber,and NAMD for the molecular dynamics of biomolecules. Unfortunately, there are only a handful of codes in use for the simulation of ET, and only one of them is freely available [10]. In this thesis I will present the basic ideas of **vertex dynamics models**, and then describe the implementation of one of them in a freely available modeling tool for the community.

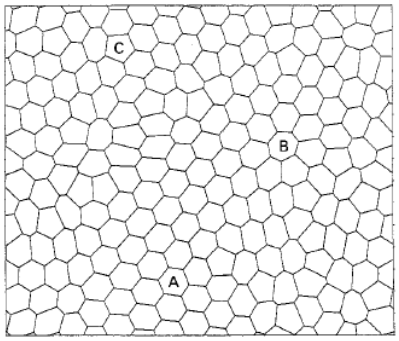
[1] Even a leading epithelial tissue simulator, Chaste, still does not have stable 3D modeling capabilities



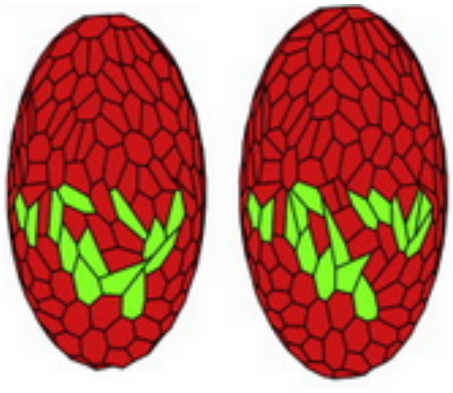
(a) Simple Squamous Tissue Bending in 3D[27]



(b) Comparison of Living Tissue and Simulation[1]



(c) Equilibrium Mesh[23]



(d) Equilibrium Mesh on a 3d Surface [9]

Figure 1.2: Some existing models of epithelial tissue.

1.1 Modeling Epithelial Tissue

The world was so recent that many things still lacked names,
and to mention them one had to point with a finger.
- Gabriel Garcia Marquez

A two dimensional **vertex dynamics model** of epithelial tissue is made up of vertices and edges[15]. A cell is represented as a polygon in 2 dimensions, or a polyhedron in 3 dimension. This model presupposes that the movement of cells in epithelial tissue can be approximated by the movement of the vertices which make up the cell. Some force is hypothesized to be the guiding force behind epithelial cell movement, and this force is applied to all of the vertices in the mesh of cells, transforming the tissue.

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[8], whereas others have accurately reproduced the dynamics of corneal wound healing[22]. In other research, simulations have faithfully captured the effects of laser perturbations to epithelial cell junctions [1], and others have quantified parameters which are important in describing the formation of the epithelial envelope in *Drosophila*[34]. Unfortunately, these results have not come from one standard model of epithelial tissue development, but from a variety of different, often irreconcilable, models. Two models will clearly illustrate the different approaches taken to modeling tissue.

In the model developed by M. Weliky and G. Oster, forces due to osmotic pressure and contractile tension describe how vertices move [29]. 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

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

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.

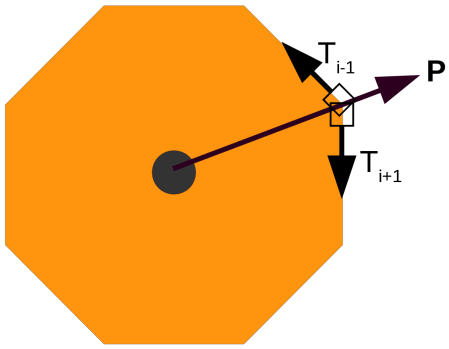


Figure 1.3: The Weliky-Oster Force

The model developed by H. Honda and T. Nagai takes an approach to modeling epithelial tissues rooted in the study of cellular structures. 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

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 free energy function as described above, and then apply the resulting force to the vertices in the epithelial mesh.

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[8] (suggesting that the basic formulation is quite acceptable to physicists). This model has been extensively researched, but can still be better understood (i.e. there is still no rigorous proof of why it achieves the equilibria it does due to certain parameterizations.

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[18].

Based upon the notion described in the preceding section, that biological cells move in a way quite similar to crystallites at high temperature[2], 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 free energy function, since the model presupposes the principle of maximum entropy is the guiding principle behind epithelial cell equiliibria. Then, the free 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

[2]Often referred to as grain growth

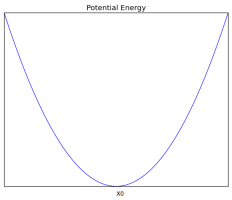


Figure 1.5: Potential Energy as a Function of Distance from Equilibrium.

the equilibrium, the steeper its gradient it will be, and the more forcefully it will want to return to equilibrium.

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 cellular distribution phenomena through the cadherin-subtype-specific 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 proteins in the membranes of these cells [11].

1. The deformation energy term 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 lambda is some positive constant.

2. The membrane surface energy 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. In my model I take the circle approach.

3. The cell-cell adhesion energy 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 dimensins 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 gamma term is calculated as the average stickiness of the two cells. This is the approach taken in the molecular dynamics software CHARMM for handling pair interactions of hetero molecules [3].

γ 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.

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

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

As seen in [10], the negative gradient of this potential energy is:

$$P = \sum_{j=1}^{N} d_j = \sum_{j=1}^{N} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \tag{1.10}$$

Therefore

$$\nabla_i P = \nabla_i d_{i-1} + \nabla_i d_i \tag{1.10}$$

and

$$\nabla_i d_{l,j} = \frac{1}{d_{l,j}} \Big(x_{j+1} - x_j, \;\; y_{j+1} - y_j \Big) \tag{1.10}$$

Substituting the above values into the equation:

$$-\nabla_i U = -\nabla_i (U_D + U_S + U_A) = F_i \tag{1.10}$$

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

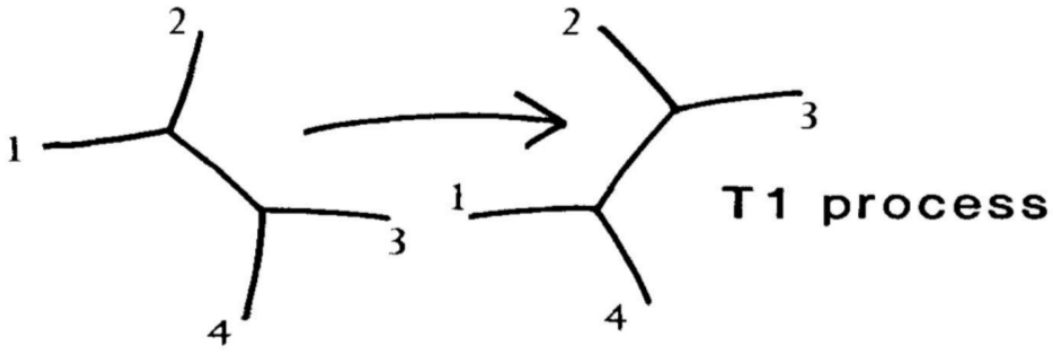


Figure 1.6: A T1 Swap. Two neighboring cells are no longer neighbors after the swap.

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)[24]. Since the coordination 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 [36]. As it turns out, there are three changes which can occur, called the T1, T2 and T3 swaps, and the Honda-Nagai Model implements the T1 and T2 swaps.

The T1 swap and in illustrated in Figure **??**. This is also called a "neighbor exchanging swap" because, as you can see, two cells which 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 force to drive the vertices into each other 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. Our model performs this action discretly 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 **??**,

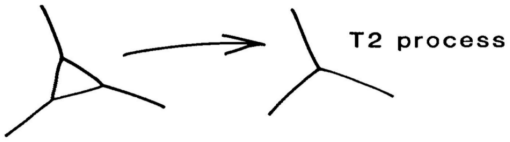


Figure 1.7: A T2 Swap

1.2.3 Selection of Parameters

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 [23]. Typically, one would not choose the values of the parameters α, β and *gamma*, but would instead have some dimensional biological data and go through the necessary conversion steps to convert these parameters to the simpler ones[28].

Interestingly, I haved found very few explicit statements of the parameters used in simulation (exceptions are [22][10][28]). 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 [3][34][37]. This will likely open new doors for the correct parameterization of current models, or for the reformulation of their descriptions of 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 **??**). 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 [12]. 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 [**?**]. 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 [36].

[3]Morphogenesis is the development of shape in an organism.

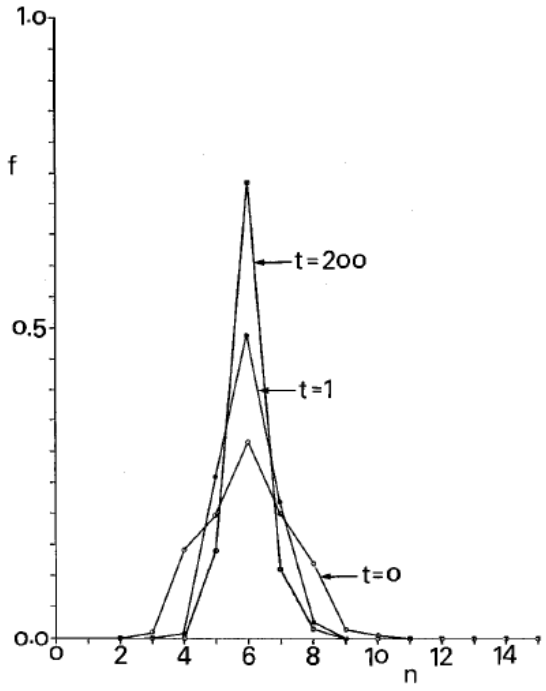


Figure 1.8: The distribution of cell shapes as a function of time [23] in the original Honda-Nagai Model.

1.3 Further Remarks About the HNM, and Epithelial Modeling in General.

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 [8], which gives a feeling of universal acceptance of this formulation. The Farhadifar formulation is:

$$E_i = \sum_{cells} \frac{K}{2}(A - A_0)^2 + \sum_{edge} \gamma_{edge} d_{edge} + \sum_{\alpha} \frac{\beta}{2} P_\alpha^2 \tag{1.10}$$

Remember the formulation of the HNM energy:

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

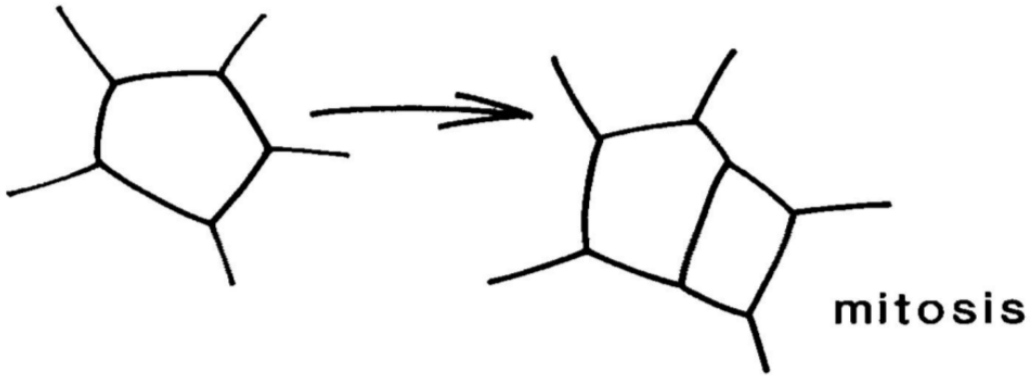


Figure 1.9: The T3 Swap

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 [23] 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 [32][13]. 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 [24]. *Epithelium* is capable of handling the T3 swap, and more discussion of this topic will appear in Chapter 3.

1.3.3 The Euler Characeteristic and Its Implications

In this section I will expand upon some observations made in [36]. **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:

$$\chi = V - F + E \tag{1.10}$$

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. The majority or current vertex dynamics models assume that vertices will have a coordination number of 3, since empirical evidence shows that the vast majority of cells have this propertyEpithelialTopology[12]. While there are certain models which consider *rosettes*, for now we will assume that all vertices connect 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:

$$3V = E \tag{1.10}$$

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

$$3V = 2E \tag{1.10}$$

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 equal to the number of cells with 3 sides plus the number of cells with 4 sides plus the number of cells with 5 sides plus . . . each multiplied by the number of edges per cell. THis gives us:

$$\sum_{k=3}^{N} kF_k = E \tag{1.10}$$

Where N is the highest number of edges encountered by 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 :

$$\sum_{k=3}^{N} kF_k = 2E \tag{1.10}$$

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

$$V - F + E = \chi \tag{1.10}$$

$$\frac{2E}{3} - F + E = \chi \tag{1.10}$$

$$\frac{5E}{3} - F = \chi \tag{1.10}$$

$$\frac{\sum_{k=3}^{N} kF_k}{6} - F = \chi \tag{1.10}$$

$$(\frac{\sum_{k=3}^{N} kF_k}{F} - 6)F = 6\chi \tag{1.10}$$

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 rosettes will not affect this result in the limiting case, and even if our model does include vertices of degree higher than three, they ought not to have a great effect on the average cell shape.

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 a that there exists a distribution which guarantees that any percentage of cells in the tissue have exactly six edges. It is feasible that the tissue could be made up entirely 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.

1.3.4 Rosettes

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. This is because empirical evidence shows that the *majority* of vertices in a tissue will have out degree three. 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 [19] 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.

Chapter 2

Epithelium

"All models are wrong, but some are useful" - G. Pox

2.1 About Epithelium

For my Master's Thesis I have implemented the Nagai-Honda Model for epithelial tissue development. My software is easy to install, takes up less than 50Mb, comes in parallel and non-parallel versions (Version 1.0.0, Version 1.0.1) and has very few dependencies (and they are likely already installed on mac and linux computers).

My code allows users to specify all parameters of interest in a couple of well commented configuration files, can handle simulations of arbitrarily large size, and can generate beautiful animations and plots of epithelial tissue development as well as useful plots of various quantities which are of interest to the researcher.

On top of all of this, 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 very simple formats(which users with scriting experience can transform to fit into the graphical utilities of their choice). In addition, the cell and vertex classes are well documented and can be extended to output new data, as users may need. What follow are a few graphs to give some flavor of what *Epithelium* can do.

Mention how Chaste was a step in the right direction, but the dependencies are too many and the it is difficult to get started. Epithelium is lightwiehgt and the code is easy to read, a better introductory tool.

2.2 Sample Configuration Files

2.2.1 config.txt

In this file, the user an 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 is explained in the Code Design chapter of this paper. 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 is the maximum nuber of random perturbations the code will make to the mesh. The upper bound is an integer which is upper bound of the range for a random number generator. The random swaps occur when a '1' is generated, and we choose random numbers from [1:upper bound].

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. It must be a multiple of ten, so if you want more closely spaced images you can just drop the maximum step size for the numerical integrator. The numerical integrator takes adaptively sized steps, and this will be discussed in more detail later.

```
13 # Dimension of mesh MUST BE ODD!!!!
1 # Plot the energy in the system? [1/0]
1 # Make a movie in the end? [1/0]
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 a bar graph of cell sides? [1/0]
0 # Make a histogram of the areas? [1/0]
0 # Make a histogram of the perimeters? [1/0]
```

2.2.2 parameters.hpp

In this header the user can define some global parameter values for the cells in the mesh. Unfortunately, you must recompile every time a parameter is changed. This could be rewritten such that parameters are read in from a configuration file, but that is just not the way the code was developed.

```
const double beta = 0.1;
const double lambda = 0.1;
const double pi = 3.141592;
const double t_gamma = 1;
const double t_area = 14.0;
```

2.2.3 changemesh.txt

This file is a configuration file, and does not require you to recompile when you change it. You can fine tune the properties of the cells in the mesh with this file.

```
# This file contains number of gamma changes,
# followed by the number of area changes,
# followed by the indices and new values
2 3
16 4.0
17 5.1
3 1.0
4 1.0
10 3.7
```

2.3 Image Gallery

This selection of graphs captures the properties of the mesh with various choices of parameter values. The target area for each cell was set to 4.0, unless otherwise specified. A couple of plots have outliers in the number of sides the cells have, or in the perimeters attained. This is due to the individual specification of cell properties for a handful of cells. In this way, the distribution of cell properties was more finely controlled. The images come in groups of four. There are two parameter studies per page. On the right you will see the

equilibrium area and perimeter reached after applying the force, and the on the right you will see the equilibrium distribution of cell shapes along with a plot of the decreasing energy in the mesh as a function of time.

2.4 Why *Epithelium* is a good piece of software

To close our discussion of *Epithelium* I would like to discuss why it is a great piece of software for *you*. *GrowFlesh* has only a couple of dependencies which are likely already installed on you computer if you are interested in computational science. You will need a compiler supporting the C++11 standard, python2.7, the matplotlib plotting library for python and the numpy python library. The only packages which will likely not be installed on your computer are the geomview geometric visualization software and the imagemagick image conversion program. Geomview and ImageMagick are free, and are such good programs that they have been around for more than 20 years. Of course, all of the animation, and plot information is outputed from this software in file formats that are easy to adapt to the user's choice of visualization software. The energy and histogram files are space separated lists and the mesh information is outputted in the OFF file format, which features a list of coordinates followed by a list of polygons which are defined counterclockwise as a list of vertex indices from the preceding list of vertices. A sample mesh file is shown below to convince you of how simple this file format is, and to suggest that you could write a smal script to change this file to a different format that will be acceptable to your choice of visualization software.

```
{appearance {+edge}
OFF
# This is the OFF formate for a square.
# We have the numVerts, numFaces, and a zero
# Then a list of vertex coordinates
# Then a list of faces described counterclockwise
# by their vertices.
4 1 0
1 0 0
1 1 0
0 1 0
```

```
0 0 0
4 1 2 3 4
```

2.5 The Design of *Epithelium*

In this chapter I will explain the internals of *GrowFlesh*. I will outline the data structure I used for the model, and some of the interesting algorithms I desinged to calculate inter- and intra- cellular forces. It is interesting to note that a number of datastructures and algorithms were scrapped in the beginning of the modeling process because of the complicated relationship between and vertices. The model I have implemented is a vertex dynamics model, and, as such, one would think that the main object of interest should be a vertex. Vertices require information about the cells which contain them in order to be moved. With this as a starting idea, I developed a sophisticated vertex class which contained cells as member variables. Apart from the software bloat coming from many vertices containing massive amounts of cell information, another issue was that cells are made of vertices, and, as such, cells ought to contain vertices as member variables! Unfortunately this bilateral inclusion is not supported by C++. In this chapter I will describe how this issue was circumvented through the use of tables which mimic the behavior of a relational database.

2.6 [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) # if any displacement in temp_mesh,
```

```
                                        # the displacement is recalculated
    temp_rotate_mesh = NagaiHondaForce(rotate_mesh)
    mesh $<$- mesh + temp_mesh
    rotate_mesh $<$- rotate_mesh + temp_rotate_mesh
    performT1(mesh)
    performT2(mesh)
    performT1(rotate_mesh)
    performT2(temp_rotate_mesh)
rotate_back(rotate_mesh)
compare_mesh(mesh, rotate_mesh)
print(graphs and error analysis)
```

2.7 Classes

GrowFlesh uses two classes to organize data. They are the cell and coordinate classes.

2.7.1 The Cell Class

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 (coordinates) 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.

```
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;
};
```

2.7.2 The Coordinate Class

The coordinate class stores the index of a coordinate, and whether or not the vertex will move during the integration. *GrowFlesh* will run with two types of meshes: meshes with border, and meshes without. A mesh with border is a mesh with fixed border elements. A mesh without border is a mesh which was generated with periodic boundary conditions and which maintains the periodicity throughout the calculations done by the program. Border vertices will not move, whereas interior vertices will be able to move; the coordinate class allows us to secify whether or not a vertex is interior or exterior. Another benefit of this class is that is allows use to easily extend the code to include forces acting on individual vertices, and to specify other types of vertices besides interior and exterior. This code was developed with eyes to the future. Another interesting idea to explore in computational biology is active cell migration, ad the cell class allows us to specify which vertices want to move actively.

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

2.8 Initial Mesh Design

The hex_mesh() function generates an nxn mesh of cells, where the dimension represents the number of cells touching the boundary.

2.9 A Relational Database

A popular way to store data since the 1970's is to store data in group 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 store a large table with a column for customer. Next to every customer's name is the customer's address. 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, for each customer you would need $\sum_c \sum_{o_c} |o_c|$ rows in the table, and many rows would have the customer's information repeated. Similarly, item information would be repeated in every row that features this item.

A better idea is to break this data up into several tables which together define a *schema*. In this example, the following tables define a good schema:

1. Customer information. columns: name, address, order number
2. Order numbers: columns: order number, item
3. Items: columns: item, item description

We are now guaranteed that the customer information is not duplicated for every item in an order, and that item information is not duplicated every time an item is in an order.

My data structure is inspired by the relational database model. I have six tables, including the simulationCells, coordinateList, X, Y, tempX and tempY tables. The cell and coordinate tables are implemented as 1d vectors of cell and coordinate indices, whereas the position tables are implemented as 1d arrays for ease of passing these structures to CUDA C functions (will talk more about CUDA C later). The cells can extract coordinate information from the coordinateList 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 memory because the cells, coordinates, and coordinate locations are stored independently of each other, and there is no data redundancy.

2.10 Moving the Vertices

GrowFlesh loops over the vertices, computes the force applied to each vertex and then computes a displacement due to the force using the Euler Method. In their original paper, H. Honda and T. Nagai described the use of a Modified Runge Kutta Method to move the vertices, but this method would result in extra unnecessary computations at each time step. The *Error Tolerance* section of this chapter describes how the Euler Method is numerically stable enough for this application. The same decision to use then Euler Method was made by the research group at Oxford that developed CHASTE, the *other* leading software for implementing the Nagai-Honda Model.

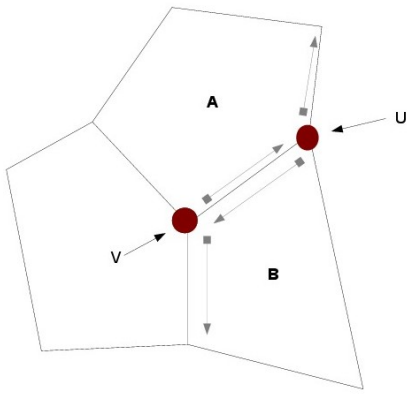
A displacement is calculated and stored in the temporary X and Y arrays. No vertex is permitted to move more than one half of the minimum delta separating vertices (the δ under which a T1 swap will occur) during an integration. By imposing this restiction we are ensuring that we will not miss the event 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 store 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. 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. In Figure 2.4 I illustrate how a vertex can find the cells surrounding it in this order. When we are integrating a vertex i, the vertex first searches in the cell vector for a cell which contains it, and then the vertex finds the next vertex in that cell. Then, we have an edge. Some other cell contains that edge if it contains both of the vertices. That cell must be clockwise from the first. The cells are stored in the reverse order of which they are uncovered by this algorithm before the integration starts. This is a very expensive step ($O(n^3)$) in the computation, and ought to be optimized.

2.11 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.



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

Figure 2.4: Getting Cells in Order

A popular hardware choice for parallel programming in last decade has been the NVIDIA GPU. CUDA is the NVIDIA extension of the C programming language which can run certain parts of C code on the GPU, while still running the serial and i/o operation on the CPU. I wrote some parts of the code in this language, and these parts were very simple to implement. The GPU is a collection of small processors, and the vertices are small, simple objects, so I mapped each vertex in the mesh to one of the hundreds of small processors to get some computational speedup.

Specifically, CUDA was employed to update vertex locations, since the sum of 1d arrays is well suited to a GPU. I also parallelized the rotation of the mesh and the reduction step which found the maximum error between the mesh and the rotated mesh. An interesting possibility is that of parallelizing the force computations. I was unable to do this with the code as it is because the cell data uses C++ vectors, while cuda only likes to take large C arrays as input. There is an interesting library called *thrust* which can strips the vectors down to arrays and does all of the memory allocations for you, but the issue then becomes one of compiling the cells into one big vector with clever structure. This is left as a future project.

CUDA was used to multiply each vertex in the mesh by the rotation matrix:

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

2.12 Computing Topological changes.

2.12.1 The T1 Swap

The PerformT1s() function loops over all of the cells in the mesh, and checks the edge lengths in that cell. If an edge is critically small, then the other three cells involved in the junction are found. One of the two clashing vertices is deleted from the main cell, and the other clashing vertex is deleted from the neighboring cell. The midpoint of the critical edge is calculated, and then a perpendicular line is drawn through it. The vertices are then moved a distance $\delta/2$ into the interior of the cells (δ is the minimum vertex separation, and the vertices now have this minimal spacing). Then, these two vertices are inserted into the cells which previously hadn't been neighbors, making then adjacent after the swap. Figure **??** offers a visual aid to understanding the swap.

There are 8 cases to consider when inserting moving the vertices the distance $\delta/2$. I and Ip1 are the critically close vertices, where Ip1 comes before I in clockwise order in a cell. *mp* is the midpoint of $(I, \bar{I}p1)$. *dx* and *dy* are the displacements which will give a minimal separation after the swap.

1. **Given:**

   I.x < Ip1.x AND I.y == Ip1.y **Then:**

   I $\mapsto$ mp + (0, -dy) AND Ip1 $\mapsto$ mp + (0, dy)

2. **Given:**

   I.x > Ip1.x AND I.y == Ip1.y **Then:**

   I $\mapsto$ mp + (0, dy) AND Ip1 $\mapsto$ mp + (0, -dy)

3. **Given:**

   I.x < Ip1.x AND I.y < Ip1.y **Then:**

   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

4. **Given:**

   I.x > Ip1.x AND I.y > Ip1.y **Then:**

   I $\mapsto$ mp + (-dx, dy) AND Ip1 $\mapsto$ mp + (dx, -dy)

5. **Given:**

   I.x < Ip1.x AND I.y > Ip1.y **Then:**

I $\mapsto$ mp + (-dx, -dy) AND Ip1 $\mapsto$ mp + (dx, dy)

6. **Given:**

   I.x > Ip1.x AND I.y < Ip1.y **Then:**

   I $\mapsto$ mp + (dx, dy) AND Ip1 $\mapsto$ mp + (-dx, -dy)

7. **Given:**

   I.x == Ip1.x AND I.y > Ip1.y **Then:**

   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

8. **Given:**

   I.x == Ip1.x AND I.y < Ip1.y **Then:**

   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

2.12.2 The implementation of the T2 swap.

The PerformT2s() function looks at every cell in the mesh and checks its area. If the area is critically small, then the cell is deleted from the mesh. The centroid of the triangle is calculated, and this become the collapsed triangle. Then, one of the three vertices has its position updated to the centroid coordinate and any cell which contained on of the other two coordinates is assigned the centroid coordinate index as a replacement.

2.13 Error Tolerance of the Algorithm

GrowFlesh has been empirically shown to be numerically stable; the code outputs an error measurement at the end of a simulation to give the user a sense of the stability. There was no clear way to provide an analytical proof of stability but at least we will always have an error measure to strengthen our confidence in the code. *GrowFlesh* runs two simultions at the same time, one on the mesh made by hex_mesh(), and another on the same mesh which has been rotated 45 degrees. 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 machine epsilon error for every simulation.

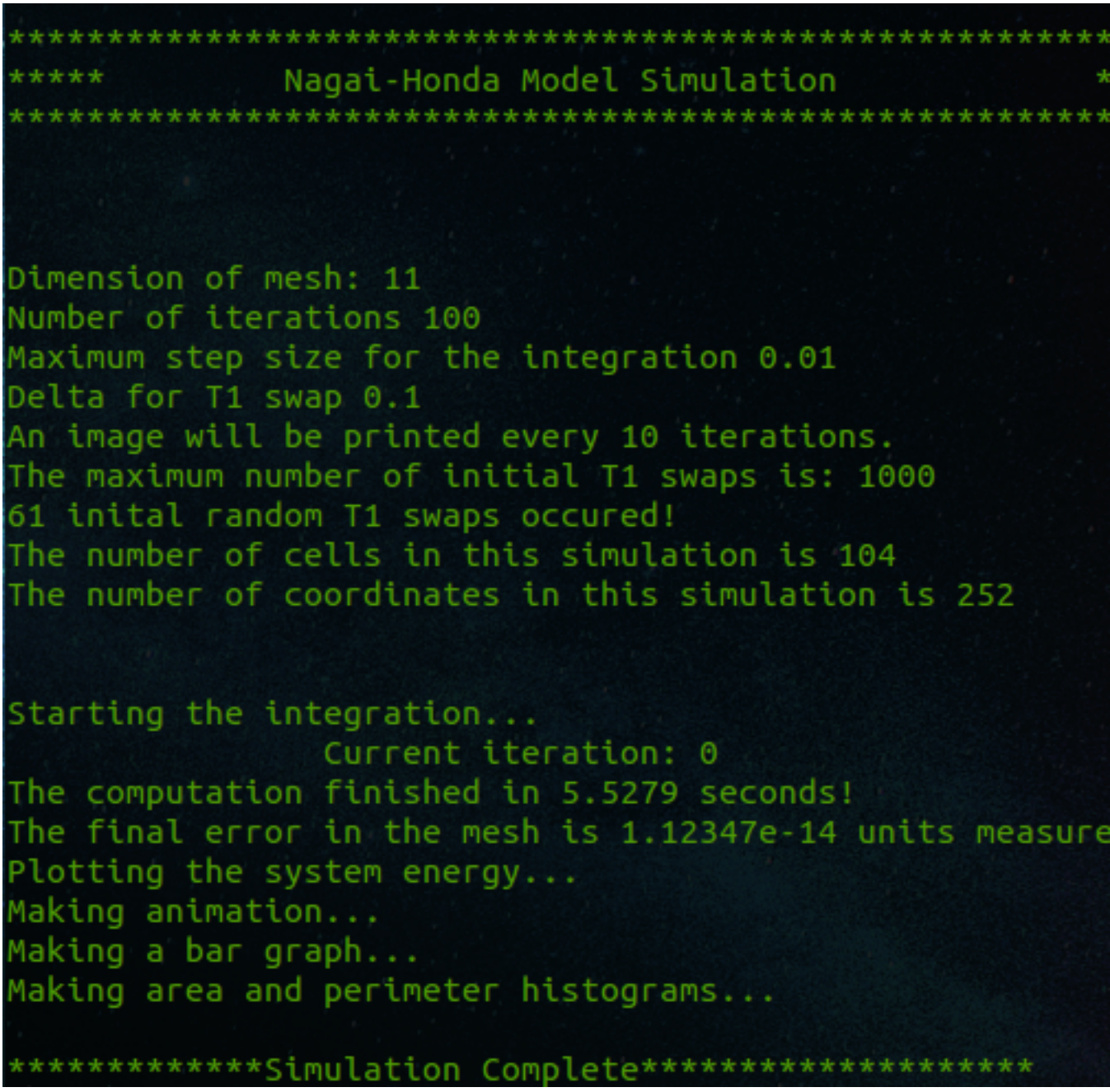


Figure 2.1: A screenshot of the simulator

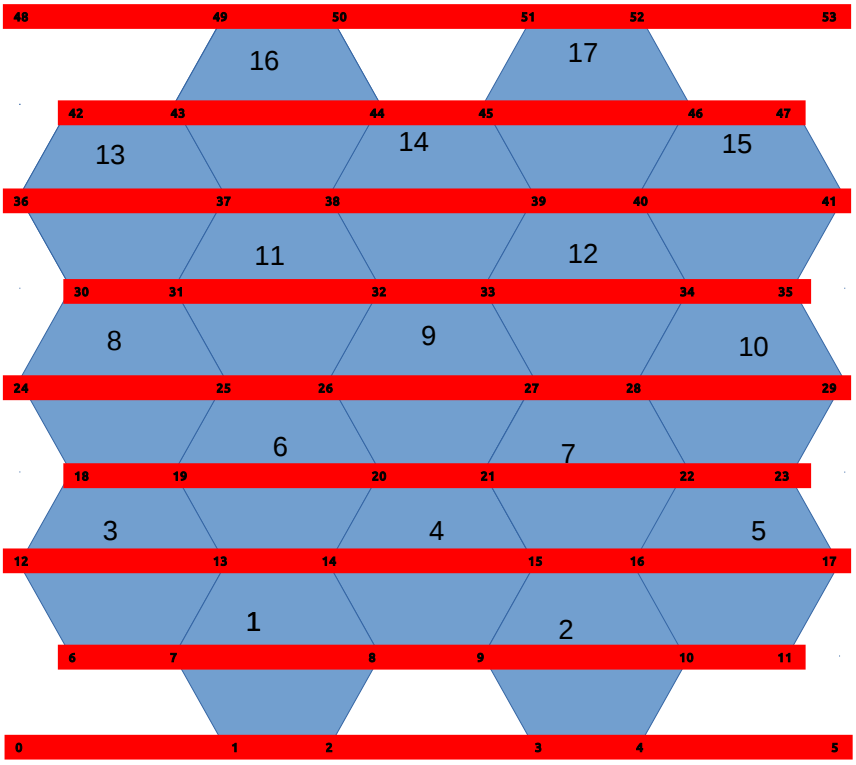


Figure 2.2: A 5x5 hexagonal mesh. The cell indices are written in the cells and the vertex indices are written on the red bars.

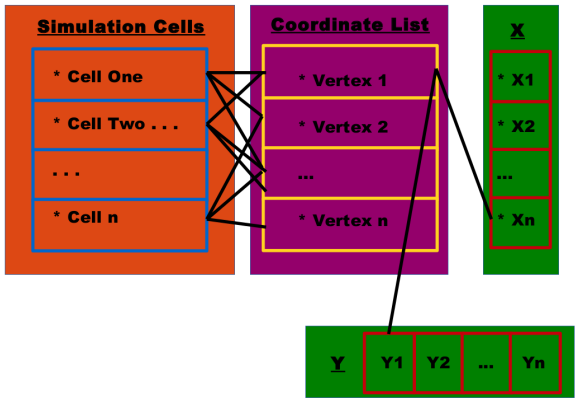


Figure 2.3: The Relational Database

../Images/rotate.png

Figure 2.5: A Rotated Mesh for Error Analysis.

Chapter 3

Advances

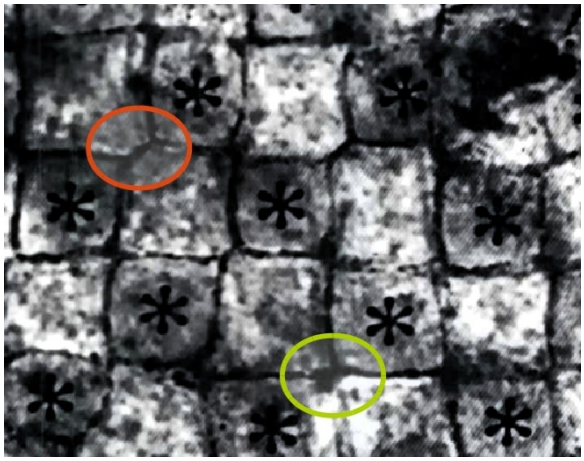


Figure 3.1: Square Cells in Quail Epithelium

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.[38]

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

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

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

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

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

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

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

[8] 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)

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

[10] 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.

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

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

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

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

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

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

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

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

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

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

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

[22] 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)

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

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

[25] 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)

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

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

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

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

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

[31] Potential Energy Graph `http://images.tutorvista.com/content/work-energy-power/potential-energy-variation.gif`

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

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

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

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

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

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

[38] 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.