Computational analysis of three-dimensional epithelial morphogenesis using vertex models

XinXin Du[1,2,*], Miriam Osterfield[3], Stanislav Y. Shvartsman[3,4]

1 Molecular and Cellular Physiology Department, Stanford University, Stanford, CA, USA

2 Bioengineering Department, Stanford University, Stanford, CA, USA

3 Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ, USA

4 Department of Chemical and Biological Engineering, Princeton University, Princeton, NJ, USA

∗ **E-mail: Corresponding xinxin@stanford.edu**

Methods

Dimensionless scaling of energy expression

Recall the energy expression from the main text:

$$E = \sum_{\alpha} \left(a_\alpha \left(A_\alpha - A_\alpha^0 \right)^2 + b_\alpha L_\alpha^2 \right) + \sum_{\langle ij \rangle} \sigma_{ij} l_{ij} + \sum_{\langle \alpha\beta \rangle} c_{\alpha\beta} \left(-\hat{\mathbf{A}}_\alpha \cdot \hat{\mathbf{A}}_\beta + 1 \right) \quad . \tag{1}$$

Let the unit of length λ be the length of the side of a hexagon with area A_0, and let the tension of a bond between two cells σ be the unit of force. Then $A_0 = 3\sqrt{3}/2\ \lambda^2$, and we may write all quantities in Equation 1 as a dimensionless number, indicated notationally by primes multiplying dimensionful units.

$$E = E'\sigma\lambda \quad , \quad a_\alpha = a'_\alpha \frac{\sigma}{\lambda^3} \quad , \quad A_\alpha = A'_\alpha \lambda^2 \quad , \quad b_\alpha = b'_\alpha \frac{\sigma}{\lambda} \tag{2}$$

$$L_\alpha = L'_\alpha \lambda \quad , \quad \sigma_{ij} = \sigma'_{ij}\sigma \quad , \quad l_{ij} = l'_{ij}\lambda \quad , \quad c_{\alpha\beta} = c'_{\alpha\beta}\sigma\lambda \quad . \tag{3}$$

Rewriting equation 1 and substituting the primed quantities, we have:

$$E'\sigma\lambda = \sum_{\alpha} \left(a'_\alpha \frac{\sigma}{\lambda^3} \left(A'_\alpha \lambda^2 - \frac{3\sqrt{3}\lambda^2}{2} \right)^2 + b'_\alpha \frac{\sigma}{\lambda} L_\alpha^{'2} \lambda^2 \right) + \sum_{\langle ij \rangle} \sigma'_{i,j} \sigma l'_{ij} \lambda + \sum_{\langle \alpha\beta \rangle} c'_{\alpha\beta} \sigma\lambda \left(-\hat{\mathbf{A}}_\alpha \cdot \hat{\mathbf{A}}_\beta + 1 \right) \quad . \tag{4}$$

Dividing every term by $\sigma\lambda$, we have:

$$E' = \sum_{\alpha} \left(a'_{\alpha} \left(A'^{2}_{\alpha} - \frac{3\sqrt{3}}{2} \right)^{2} + b'_{\alpha} L'^{2}_{\alpha} \right) + \sum_{\langle ij \rangle} \sigma'_{i,j} l'_{ij} + \sum_{\langle \alpha\beta \rangle} c'_{\alpha\beta} \left(-\hat{\mathbf{A}}_{\alpha} \cdot \hat{\mathbf{A}}_{\beta} + 1 \right) \quad , \tag{5}$$

where conversion between dimensionless numbers and physical quantities are translated according to Equation 3, and one of the values of σ'_{ij} has dimensionless value 1. The scale of time may be set as

$$\tau = \frac{\lambda}{\eta\sigma} \quad , \tag{6}$$

such that the dimensionless mobility $\eta' = \eta\tau\sigma/\lambda = 1$. In the main text, primes are dropped from dimensionless variables for notational simplicity.

Verification of the independence of cell area vector from triangulation

Consider a single triangle resulting from a triangulation of a cell bounded by $\{\mathbf{x}_i\}$ and triangulated with $\mathbf{x}_c$. According to Stoke's Theorem, we have

$$\int_{\partial S} \mathbf{F} \cdot d\boldsymbol{\ell} = \int_{S} \nabla \times \mathbf{F} \cdot d\mathbf{S} \tag{7}$$

where S is a surface and ∂S is the boundary of S, and $\mathbf{F}$ is a tensor field. If there exists an $\mathbf{F}$ such that $\nabla \times \mathbf{F} = I$, the identity matrix, then the RHS of Equation 7 will be equal to the area vector of the triangle: $\int_S d\mathbf{S} = \mathbf{A}$. Meanwhile, the LHS depends only on the path bounding the triangle. If the area vector of a triangle depends only on the path of the triangle boundary, then, as long as the cell forms a closed loop, as we sum the area vectors of triangles forming a cell, all contributions to line integrals that pass through the point of triangulation cancel. We only need to exhibit a second rank tensor $\mathbf{F}$ satisfying $\nabla \times \mathbf{F} = I$ in 3d, where I is the identity matrix. Consider

$$\mathbf{F} = \begin{pmatrix} 0 & -\frac{z}{2} & \frac{y}{2} \\ \frac{z}{2} & 0 & -\frac{x}{2} \\ -\frac{y}{2} & \frac{x}{2} & 0 \end{pmatrix} \quad . \tag{8}$$

The mkth component of $\nabla \times \mathbf{F}$ in 3d is given by $\epsilon_{ijk}\frac{\partial F_{mj}}{\partial x^i}$, where ϵ_{ijk} is the Levi-Civita symbol, and summation over repeated indices is understood. By inspection, $\mathbf{F}$ satisfies $\nabla \times \mathbf{F} = I$. So we have shown that the area vector of a cell defined by a loop is independent of triangulation.

Resolving repeated formations of T1 junctions using noise

For T1 junctions that form repeatedly, we implement a step in which the tension of the bond that results from the resolution of the junction is either multiplied or divided by a factor p after every additional successive instance that the junction forms, where multiplication or division is chosen at random in each instance. The factor p depends on the number of successive times the junction has already formed, for example, p may be an increasing function of this number, such that the code effectively "tries harder" to resolve more stubborn T1 junctions. The tension of the resulting bond is restored after a time t_{relax}, where t_{relax} should be chosen to be shorter than the time scale of large-scale tissue changes.

The effects of this particular resolution of T1 junctions is shown in Figure S1 and Movies S7 and S8. We simulate a T1 junction that forms repeatedly due to the fact that it is temporarily stable during the time evolution, but whose configuration after all vertices reach equilibrium consists of two 3-vertices with a *very short* vertical bond. Numerical implementations with and without our treatment for stubborn T1 junctions are analyzed. With the implementation of noise, it takes ≈ 11000 time steps to integrate the equations of motion over a simulation time range that covers the resolution of the junction, whereas without our implementation, it takes ≈ 26000 time steps to integrate the equations of motion over the same time range.

These details of implementation for repeated T1 junctions are somewhat arbitrary, and simulation results do not depend significantly on them. Below, we indicate a choice of implementation that might be used to efficiently integrate the equations of motion past repeated T1 junctions. Due to the noise introduced from this implementation, different final equilibrium configurations can result for each simulation for the same set of parameter values. However, if the system is robust, the various final equilibrium configurations resulting from implementation of noise should be very similar; an example of this was analyzed in [1].

In the example given in Figure S1, for the resolution of the repeated T1 junction, we have set the factor p to be a decreasing step function of the number of times n that the same T1 junction has already formed, with $p = 0.8$ for $n_{\text{cut}} < n \leq 2n_{\text{cut}}$, $p = 0.7$ for $2n_{\text{cut}} < n \leq 3n_{\text{cut}}$, $p = 0.6$ for $3n_{\text{cut}} < n \leq 4n_{\text{cut}}$,

and $p = 0.5$ for $4n_{\text{cut}} < n$, where n_{cut} is an threshold integer intuitively indicating "patience level" for "stubborn" T1 junctions such that larger n_{cut} indicates willingness to wait longer before increasing p. For other values of n, we chose $p = 1$. For the example of budding, we put $n_{\text{cut}} = 6$. Additionally, we recognize instances where two bonds close to each other are repeatedly intercalating in an alternating pattern (an ABABAB... pattern) or a triplet pattern (an ABCABCABD... pattern); these usually signify a 6-vertex-like or 8-vertex-like structure, respectively. In both these cases, in our examples from the main text, we put $p = 0.5$ for patterns that repeat more than 9 times.

After the formation of a T1 junction, we disallow the same T1 junction from forming for a certain number of time integration steps. Additionally, if the tension of a bond is temporarily changed by the factor p, the tension is restored after a time t_{relax}, where t_{relax} is small compared to times required for large cell movements, and t_{relax} is larger for smaller values of p. Specifically, in most of our examples, we have set $t_{\text{relax}} = 0$ for $p = 1$, $t_{\text{relax}} = 1$ for $p = 0.8$, $t_{\text{relax}} = 2$ for $p = 0.7$, $t_{\text{relax}} = 3$ for $p = 0.6$, and $t_{\text{relax}} = 4$ for $p = 0.5$.

Details of numerical implementation

The equation of motion

$$\frac{d\mathbf{x}}{dt} = -\eta \frac{\partial E}{\partial \mathbf{x}} \tag{9}$$

is propagated for each vertex at position $\mathbf{x}$ using a second-third order, time-adaptive, Bogacki-Shampine method [2]. The updating scheme at each time step is briefly as follows:

1. Let the configuration be given by $\{\mathbf{x}\}$ at time t. Evaluate for each vertex, based on all the vertex positions, the right hand side of Equation 9 or equivalently, the forces in Equation 6 of the main text. Use these forces to evaluate all derivatives of $\{\mathbf{x}\}$ used for the implementation of the Bogacki-Shampine method.

2. Calculate the size of the time step dt to be taken specified by the Bogacki-Shampine method as a function of the numerical error tolerance which is set by the user. The size of the first time step is set explicitly by the user.

3. Use dt to evaluate $\{d\mathbf{x}\}$ and put

$$\mathbf{x}(t+dt) = \mathbf{x}(t) + d\mathbf{x} \tag{10}$$

for each vertex.

4. Update all explicitly time-dependent parameters to their values at $t+dt$. For example, parameters such as bond tension may be prescribed in the simulation to increase as a function of time.

5. Perform all necessary discrete rearrangements such as intercalation.

6. Update the values for line tension in bonds participating in an intercalation event: a bond that was part of a "cable" of tension may no longer be after an intercalation, and visa versa; additionally, bonds having a position-dependent line tension would have moved during the intercalation rearrangement.

7. Set $t \to t+dt$.

All simulations are coded in object-oriented C++. The main objects in the algorithm are the Vertices, Cells, and Bonds. Cells store data pertaining to patterning at the cell level, i.e. cell type, and parameters a_α, b_α, and A^0 corresponding to that cell. Cells additionally store data about their areas and perimeters as well as the identities of the Vertices participating in that cell. Bonds store data pertaining to patterning at the bond level, i.e. σ_{ij}, as well as their length, the identities of Vertices participating in that Bond, and the Cells separated by that Bond. Vertices store data about their positions, whether they belong to the boundary, and the identities of adjacent Bonds and Cells. At each time step, Vertices are iterated over, and force on each Vertex is calculated using the parametric and geometric information stored in the Vertex object. A Simulation object implements time propagation and stores data pertaining to explicitly time-dependent parameters.

References

1. Osterfield M, Du X, Schüpbach T, Wieschaus E, Shvartsman SY (2013) Three-dimensional epithelial morphogenesis in the developing *Drosophila* egg. Developmental Cell 24: 400–410.

2. Bogacki P, Shampine LF (1989) A 3(2) pair of runge-kutta formulas. Applied Mathematics Letters 2: 321–325.