Implementation of the two-dimensional vertex model

Rastko Sknepnek, University of Dundee, United Kingdom

21st August 2024

These are lecture notes for the lecture on 'Vertex model for tissue mechanics' delivered at the Summer School 'Non-equilibrium Processes in Physics and Biology' at the Erwin Schrödinger International Institute for Mathematics and Physics at the University of Vienna in August 2024. The lecture notes are accompanied by a code available on GitHub: github.com/sknepneklab/VMTutorial.

1 Introduction

Morphogenesis, i.e. a biological process by which organisms develop, wound healing, cancer progression, etc. all involve coordinate movement of collectives of cells. In vitro experiments on epithelial cell monolayers grown on a substrate have been one of the prime systems for studying collective cell behaviours, and physics of active matter in general. Such biological and in vitro processes involve generation of mechanical forces by cells and their transmission to the neighbouring cells and the surrounding. In order to model and understand processes that drive collective cell behaviours, one resorts to a range of theoretical models. A large class of such models are cell-level models, which model the tissue with a cell-level resolution. The level of details depends on the model and the problem of interest, but the general idea is to approximate properties of individual cells and interactions between them in term so several parameters, which enables studies of large collectives of cells.

In this lecture, we will focus on modelling mechanical properties of confluent epithelial sheets. Epithelial cells form epithelia, one of the four primary types of tissues. Epithelial tissues form the covering or lining of all internal and external body surfaces, including skin, the lining of the digestive tract, and the lining of blood vessels. They act as a barriers, provide protection, and are involved in absorption, secretion, and sensation. Due to epithelia acting as barriers, their cells are tightly connected to each other forming confluent (i.e. no gaps) structures. Depending on the location in the body and their function, epithelia can be of several types (e.g. simple, stratified, glandular, etc.).

We focus on simple epithelia formed of a single layer of cells (Fig. 1a) . We also assume that the epithelium is nearly flat and the we are only interested in processes that occur at the apical (i.e. top) side. In this case, the epithelium can be approximated as a two dimensional sheet to tightly packed cells. Seen from above (Fig. 1b) the sheet looks like a group of polygons packed tightly together (Fig. 1c).

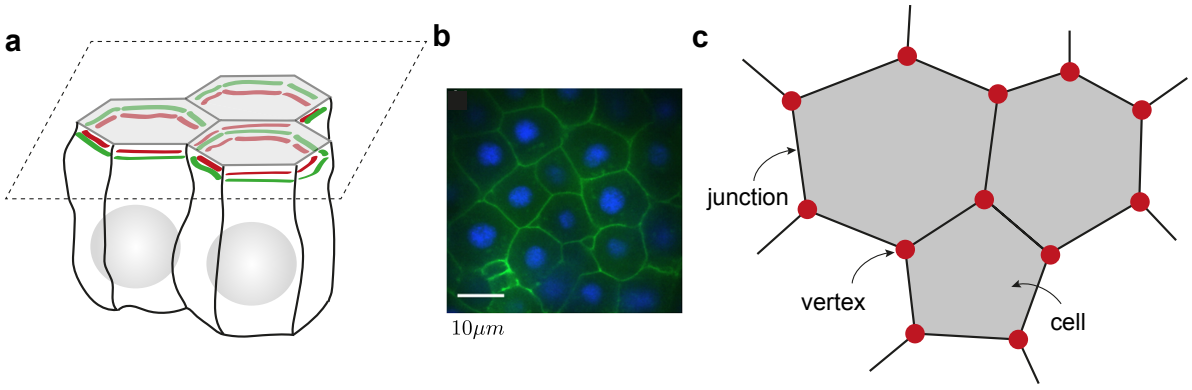

Figure 1: The idea behind the apical two-dimensional vertex model. **a** Artistic representation of the columnar epithelium (inspired by [2]). **b** A top (apical) view of zebrafish ectoderm cells [3]. **c** Apical side of the tissue is represented as a polygonal tiling of the plane. Cells are represented as polygons; two cells share a junction; three or more junctions meet at a vertex (red dots).

It is, therefore, reasonable to model such planar epithelium as a polygonal tiling of the plane, giving rise to the two-dimensional apical *vertex model*. The name originates from the observation that two polygons share an edge (i.e. a junction). Three of more junctions meet at a vertex. Moving vertices changes shape of polygons connected to it, so the vertex is the degree of freedom. All one needs to do is assign mechanical energy to each configuration and use it to derive equations of motion for each vertex. The goal of this lecture is to explain how to do this.

The vertex model has a very long history. For a recent review see, e.g. [1].

2 Vertex model

2.1 Energy

The mechanical energy of a two-dimensional confluent tissues can written as

$$E = \sum_C \frac{\kappa_C}{2} \left(A_C - A_0^C\right)^2 + \sum_C \frac{\Gamma_C}{2} P_C^2 - \sum_{\langle ij \rangle} \Lambda_{ij} l_{ij}, \tag{1}$$

where the sum is over all cells, κ_C is the cell stiffness, Γ_C is the perimeter stiffness, Λ_{ij} is the line tension, which can be both positive or negative, A_C is area of cell C, P_C is its perimeter, and l_{ij} is length of the junction connecting vertices i and j. A_0^C is the preferred apical cell area of cell C, and $\langle ij \rangle$ denotes sum over all junctions. In general, all parameters can be cell-dependent. If, however, for simplicity we assume that all parameters have the same value for all cells and junctions, i.e. $A_0^C \equiv A_0$, $\kappa_C \equiv \kappa$, $\Gamma_c \equiv \Gamma$, and $\Lambda_{ij} \equiv \Lambda$, we can simplify the last term

$$\sum_{\langle ij \rangle} \Lambda_{ij} l_{ij} = \Lambda \sum_{\langle ij \rangle} l_{ij} = \frac{\Lambda}{2} \sum_C P_C,$$

When deriving the last expression, we assumed that all junctions are shared between two cells (i.e. there are no boundaries). We can then combine (we assume $\Lambda > 0$)

$$\sum_C \frac{\Gamma}{2} P_C^2 - \frac{\Lambda}{2} \sum_C P_C = \frac{\Gamma}{2} \sum_C \left(P_C^2 - \frac{\Lambda}{\Gamma} P_C + \underbrace{\left(\frac{\Lambda}{2\Gamma}\right)^2 - \left(\frac{\Lambda}{2\Gamma}\right)^2}_{=0} \right)$$
$$= \frac{\Gamma}{2} \sum_C \left(P_C - \frac{\Lambda}{2\Gamma} \right)^2 - \sum_C \frac{\Lambda^2}{8\Gamma}.$$

The last term is a constant $-\frac{\Lambda^2}{8\Gamma} N_C$, where N_C is the number of cells. It, however, does not depend on the configuration of the tissue and, therefore, does not contribute to dynamics. So, we simply ignore it. If we introduce the preferred perimeter $P_0 = \frac{\Lambda}{2\Gamma}$, we can write the expression for energy in a symmetric form

$$E = \frac{\kappa}{2} \sum_C \left(A_C - A_0 \right)^2 + \frac{\Gamma}{2} \sum_C \left(P_C - P_0 \right)^2 . \tag{2}$$

We will use both forms of energy [Eqs. (1) and (2)] in these lecture notes.

2.2 Equation of motion

The equation of motion of vertex i is

$$\gamma \frac{d\mathbf{r}_i}{dt} = -\frac{\partial E}{\partial \mathbf{r}_i} + \boldsymbol{\xi}_i, \tag{3}$$

where γ is the friction constant. It is worth noting that the assumption in this model is that the dissipation is with the environment, i.e. the substrate cell reside on and/or with the surrounding fluid. The functional form $\gamma \mathbf{v}$ suggests Stokes like dissipation with the fluid (dry friction with the substrate is usually independent of the velocity magnitude). This means that, roughly speaking, $\gamma \sim \eta R$, where η is the dynamic viscosity of the fluid and R is the typical cell size. An important point is that this model completely ignores 'internal' dissipation, i.e. energy loss due to viscosity of the cells themselves. Such dissipative effects would involve terms proportional to the velocity gradients inside the sheet.

$\boldsymbol{\xi}_i$ is the zero-mean (i.e. $\langle \xi_i^\alpha \rangle$, where $\alpha \in \{x, y\}$), delta-correlated noise (i.e. $\left\langle \xi_i^\alpha (t)\, \xi_j^\beta (t') \right\rangle = 4D\delta_{ij}\delta_{\alpha\beta}\delta (t - t')$, where D is the diffusion constant). In practice, noise term is often neglected.

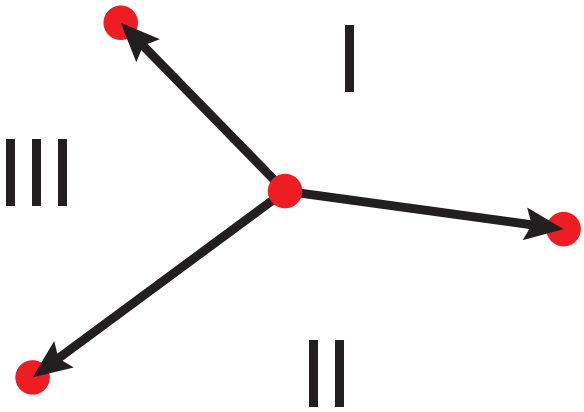

Figure 2: Labelling of cells used to compute area and perimeter forces. Arrow indicate that one 'look' away from the vertex.

2.3 Expressions for forces

For simplicity, we will use the expression in Eq. (2). Force on the vertex i is then given as

$$\begin{aligned}\mathbf{F}_i &= -\frac{\partial E}{\partial \mathbf{r}_i} \\ &= -\sum_C \left[\kappa\left(A_C - A_0\right)\frac{\partial A_C}{\partial \mathbf{r}_i} + \Gamma\left(P_C - P_0\right)\frac{\partial P_C}{\partial \mathbf{r}_i}\right]\delta_{C,\mathbf{r}_i} \\ &= \sum_C \left[p_C\frac{\partial A_C}{\partial \mathbf{r}_i} + t_C\frac{\partial P_C}{\partial \mathbf{r}_i}\right]\delta_{C,\mathbf{r}_i}, \end{aligned} \tag{4}$$

where the symbol $\delta_{C,\mathbf{r}_i}$ means that only cells that contain vertex i (typically three of them) have to be taken into account, and we introduced

$$p_C = -\kappa\left(A_C - A_0\right) \tag{5}$$

$$t_C = -\Gamma\left(P_C - P_0\right). \tag{6}$$

We now proceed to compute the force in terms of vertex coordinates. We start with the area term, by noting that

$$A_C = \frac{1}{2}\sum_j \left(\mathbf{r}_j \times \mathbf{r}_{j+1}\right)\cdot \mathbf{N},$$

where the sum is over the loop of all vertices in the cell C, $\mathbf{N} \equiv \mathbf{e}_z$ and the sign of the area is guaranteed to be positive as long as vertices are ordered in the counterclockwise direction. Finally, $n+1 \equiv 1$, where n is the number of vertices in C. Using Levi-Chivita symbol, we write

$$A_C = \frac{1}{2}\sum_j \varepsilon_{\alpha\beta\gamma} N_\alpha x^j_\beta x^{j+1}_\gamma.$$

This allows us to readily compute,

$$\begin{aligned}\frac{\partial A_C}{\partial x^i_\delta} &= \frac{1}{2}\sum_j \varepsilon_{\alpha\beta\gamma} N_\alpha \left(\delta_{\beta\delta}\delta_{i,j} x^{j+1}_\gamma + \delta_{\gamma\delta}\delta_{i,j+1} x^j_\beta\right) \\ &= \frac{1}{2}\left[\varepsilon_{\alpha\delta\gamma} N_\alpha x^{i+1}_\gamma + \varepsilon_{\alpha\beta\delta} N_\alpha x^{i-1}_\beta\right].\end{aligned}$$

If we use symmetry properties of the Levi-Chivita symbol and assume summation over pairs of repeated Greek indices, we have,

$$\begin{aligned}\frac{\partial A_C}{\partial x^i_\delta} &= \frac{1}{2}\left[-\varepsilon_{\delta\alpha\gamma} N_\alpha x^{i+1}_\gamma + \varepsilon_{\delta\alpha\beta} N_\alpha x^{i-1}_\beta\right] \\ &= -\frac{1}{2}\left[\mathbf{N}\times(\mathbf{r}_{i+1}-\mathbf{r}_{i-1})\right]_\delta,\end{aligned}$$

or

$$\begin{aligned}\frac{\partial A_C}{\partial \mathbf{r}_i} &= -\frac{1}{2}\mathbf{e}_z\times(\mathbf{r}_{i+1}-\mathbf{r}_{i-1}) \\ &= -\frac{1}{2}\mathbf{e}_z\times((\mathbf{r}_{i+1}-\mathbf{r}_i)-(\mathbf{r}_{i-1}-\mathbf{r}_i)) \\ &= -\frac{1}{2}\mathbf{e}_z\times(\mathbf{r}_{i+1,i}-\mathbf{r}_{i-1,i}) \\ &= \frac{1}{2}\mathbf{e}_z\times(\mathbf{r}_{i-1,i}-\mathbf{r}_{i+1,i}),\end{aligned}$$

i.e. magnitude of the gradient of the area is equal to the area of the triangle spanned by vertices, $\mathbf{r}_{i-1}$, $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$. If we use notation used in Fig. 2, we have

$$\begin{aligned}\mathbf{F}^{\text{area}}_i &= \frac{1}{2}p_I\mathbf{e}_z\times(\mathbf{l}_3-\mathbf{l}_1)+\frac{1}{2}p_{II}\mathbf{e}_z\times(\mathbf{l}_1-\mathbf{l}_2)+\frac{1}{2}p_{III}\mathbf{e}_z\times(\mathbf{l}_2-\mathbf{l}_3) \\ &= \frac{1}{2}(p_{II}-p_I)\,\mathbf{e}_z\times\mathbf{l}_1+\frac{1}{2}(p_{III}-p_{II})\,\mathbf{e}_z\times\mathbf{l}_2+\frac{1}{2}(p_I-p_{III})\,\mathbf{e}_z\times\mathbf{l}_3,\end{aligned}$$

or

$$\mathbf{F}^{\text{area}}_i = \sum_e \frac{1}{2}(p_{e+1}-p_e)\,\mathbf{e}_z\times\mathbf{l}_e, \tag{7}$$

where the $e-$sum loops over the edges in the clockwise direction, p_e (p_{e+1}) corresponds to the cell to the left (right) of edge e when looking form i towards j.

We proceed to compute gradient of the perimeter. Perimeter of cell C is

$$P_C = \sum_j |\mathbf{r}_{j+1}-\mathbf{r}_j|,$$

where the sum is over all vertices and, as before, $n+1\equiv 1$. It is convenient to rewrite P_C it terms

of dot products, i.e.,

$$\begin{aligned} P_C &= \sum_j \left[(\mathbf{r}_{j+1} - \mathbf{r}_j) \cdot (\mathbf{r}_{j+1} - \mathbf{r}_j) \right]^{1/2} \\ &= \sum_j \left[\left(x_\alpha^{j+1} - x_\alpha^j \right) \left(x_\alpha^{j+1} - x_\alpha^j \right) \right]^{1/2}. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \frac{\partial P_C}{\partial x_\beta^i} &= \sum_j \frac{1}{|\mathbf{r}_{j+1} - \mathbf{r}_j|} \left(x_\alpha^{j+1} - x_\alpha^j \right) \left(\delta_{\alpha\beta}\delta_{i,j+1} - \delta_{\alpha\beta}\delta_{i,j} \right) \\ &= \frac{x_\alpha^i - x_\alpha^{i-1}}{|\mathbf{r}_i - \mathbf{r}_{i-1}|} - \frac{x_\alpha^{i+1} - x_\alpha^i}{|\mathbf{r}_{i+1} - \mathbf{r}_i|}, \end{aligned}$$

or

$$\frac{\partial P_C}{\partial \mathbf{r}_i} = \hat{\mathbf{r}}_{i,i-1} - \hat{\mathbf{r}}_{i+1,i} = -\left(\hat{\mathbf{r}}_{i-1,i} + \hat{\mathbf{r}}_{i+1,i} \right).$$

Using the notation of Fig. 2, we have

$$\begin{aligned} \mathbf{F}_i^{\text{perim}} &= -t_I \left(\hat{\mathbf{l}}_3 + \hat{\mathbf{l}}_1 \right) - t_{II} \left(\hat{\mathbf{l}}_1 + \hat{\mathbf{l}}_2 \right) - t_{III} \left(\hat{\mathbf{l}}_2 + \hat{\mathbf{l}}_3 \right) \\ &= -\left(t_I + t_{II} \right) \hat{\mathbf{l}}_1 - \left(t_{II} + t_{III} \right) \hat{\mathbf{l}}_2 - \left(t_{III} + t_I \right) \hat{\mathbf{l}}_3, \end{aligned}$$

or

$$\mathbf{F}_i^{\text{perim}} = -\sum_e \left(t_e + t_{e+1} \right) \hat{\mathbf{l}}_e, \tag{8}$$

where, as above, the $e-$sum loops over the vertex in the clockwise direction, t_e (t_{e+1}) corresponds to the cell to the left (right) when looking along the edge e away from vertex i. We can combine Eqs. (7) and (8) to obtain

$$\mathbf{F}_i = \sum_e \left[\frac{1}{2} \left(p_{e+1} - p_e \right) \left(\mathbf{e}_z \times \mathbf{l}_e \right) - \left(t_{e+1} + t_e \right) \hat{\mathbf{l}}_e \right],$$

where $\hat{\mathbf{l}}_e = \mathbf{l}_e / l_e$ and $l_e = |\mathbf{l}_e|$. We conclude that the force *on* vertex v_i *from* the edge that connects it to vertex v_j is

$$\mathbf{F}_i^{(ij)} = \frac{1}{2} \left(p_2 - p_1 \right) \mathbf{e}_z \times \mathbf{l}_{ij} - \left(t_2 + t_1 \right) \hat{\mathbf{l}}_{ij}, \tag{9}$$

where index 1 corresponds to the face 1 (2) is to the left (right) of the edge (ij) we looking along the edge away from vertex i. Eq. (9) can be implemented directly in a code.

2.4 Making the model dimensionless

Although we will mainly work with the vertex model in its full form, in the literature one often encounters it being written in the dimensionless form. We start with

$$\begin{aligned} E &= \frac{\kappa}{2}\sum_C (A_C - A_0)^2 + \frac{\Gamma}{2}\sum_C (P_C - P_0)^2 \\ &= \frac{\kappa A_0^2}{2}\sum_C \left(\frac{A_C}{A_0} - 1\right)^2 + \frac{\Gamma A_0}{2}\sum_C \left(\frac{P_C}{\sqrt{A_0}} - \frac{P_0}{\sqrt{A_0}}\right)^2, \end{aligned}$$

or

$$e = \frac{1}{2}\sum_C (a_C - 1)^2 + \frac{\tilde{\Gamma}}{2}\sum_C (p_C - p_0)^2, \tag{10}$$

where

$$\begin{aligned} e &= \frac{E}{\kappa A_0^2} \\ \tilde{\Gamma} &= \frac{\Gamma}{\kappa A_0} \\ a_C &= \frac{A_C}{A_0} \\ p_C &= \frac{P_C}{\sqrt{A_0}} \\ p_0 &= \frac{P_0}{\sqrt{A_0}}. \end{aligned} \tag{11}$$

Parameter p_0 is called the shape index and it plays a key role in determining mechanical properties of the vertex model. For $p_0 < p_0^c \approx 3.81$ (for disordered tilings) the tissue is in the solid phase. For $p_0 > p_0^c$, the tissue is fluid. What this effectively means is that the energy barrier for cell neighbour exchanges vanishes as p_0 crosses p_0^c [4].

Finally, from the equation of motion to show that the unit of time is

$$t^* = \frac{\gamma}{\kappa A_0}. \tag{12}$$

Therefore, in its simplest form, the vertex model is controlled by two parameters, p_0 and $\tilde{\Gamma}$. We note that this is not the only way to make the model dimensionless since one can, e.g. use a different combination of parameters for construct the unit of energy.

3 Implementation

The key part of any vertex model implementation is the representation of the two-dimensional tissue. Given that we are working with vertices, junctions (edges), and cells (faces), we need a model for a two-dimensional mesh. There are many way one can represents a mesh, each of them with its

advantages and disadvantages. Here, we will adhere to the principle where we are aware of the performance issues but not at the expense of clarity and maintainability of the code.

3.1 Half-edge representation

In our implementation, we use the *half-edge* data structure [5] to represent the mesh, i.e. the confluent cell monolayer. Briefly, the mesh consists of vertices, edges (junctions), and faces (cells). An edge connecting vertices i and j is 'split' longitudinally into two *directed* half-edges, he_1 pointing from vertex i towards vertex j and he_2 pointing from vertex j towards vertex i (see Fig. 3). The data structure is organised as follows (this is just a skeleton and we use * to indicate pointers)

```
Vertex
{
  HalfEdge* he;  // outgoing half-edge
}

Edge
{
  HalfEdge* he;  // one of the two half-edges
}

Face
{
  HalfEdge* he;  // one of half-edges (it can be any edge)
}

HalfEdge
{
  Vertex*    v;    // vertex this half-edge points from
  Edge*      e;    // edge it is part of
  HalfEdge*  pair; // half-edge along the same edge pointing in the opposite direction
  HalfEdge*  next; // next half-edge in the same face
  HalfEdge*  prev; // previous half-edge in the same face
}
```

It is customary to adopt a convention in which half-edges within a given face (cell) are ordered in the *counterclockwise* direction. Keeping track of the direction is very important for the model to work properly. Given this setup, it is very simple to traverse the mesh. For example, if we would like to loop over a face f all we need to do is

```
he = f->he;
first = he;
```

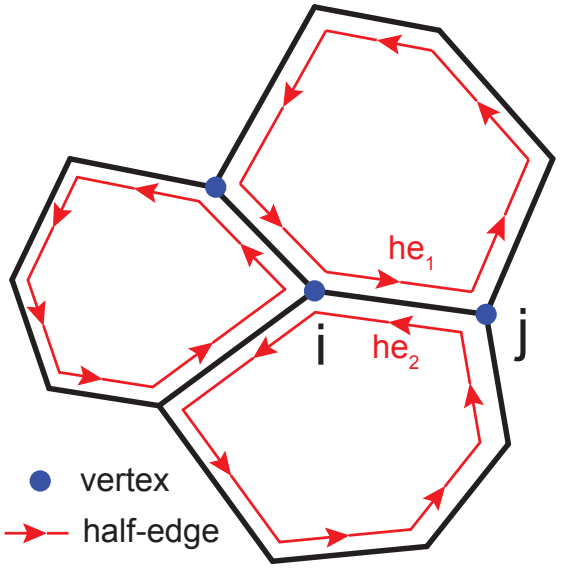

Figure 3: Half-edge data structure. Each edge is split into two directed half-edges. Half-edges in each face are ordered counterclockwise. Edge connecting vertices i and j is split into two directional half-edges, he_1 pointing from i towards j, and he_2 pointing from j towards i.

```
do
{
    // do something
    he = he->next;
} while (he != first);
```

Or, if we would like to loop over all neighbours of a given vertex v we would need code like this

```
he = v->he;
first = he;
do
{
    // do something with half-edge he
    he = he->pair->next;
} while (he != first);
```

It is, however, important to note that in the last case neighbours of a vertex are traversed in the *clockwise* direction.

In the actual implementation, we will use C++ standard library containers (*list*, in this case) and iterators instead of raw pointers. This makes memory management simpler at a minimal performance overhead.

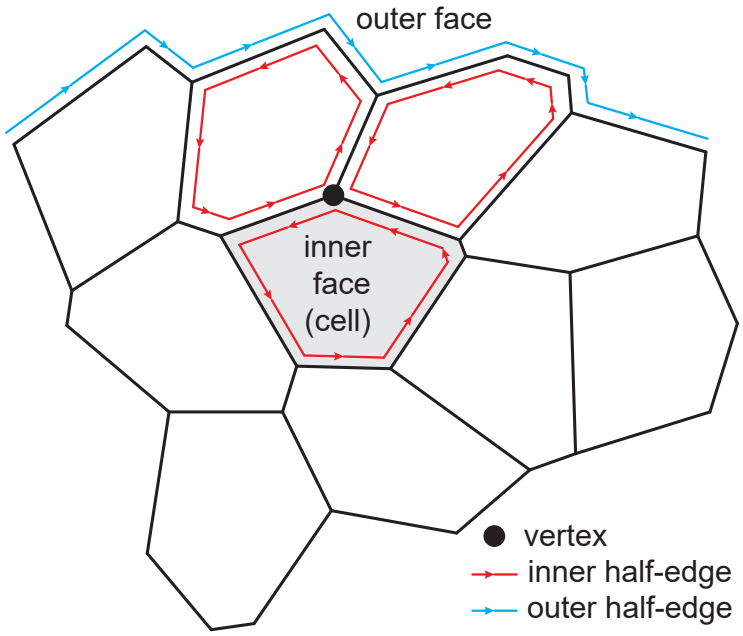

Figure 4: An example of a tissue with a boundary. The outer face contains half-edges that are ordered clockwise (blue lines). Therefore, its orientation is negative.

3.2 Handling boundaries

In general, we want to be able to simulate both systems subject to periodic boundary conditions as well as systems with open boundaries. In the later case, one needs to be able to distinguish boundary edges and vertices. There are several ways to do this. Here we adopt the approach where we introduce so-called *outer* face, i.e. a fictitious face that encircles the entire tissue and has clockwise orientation. An example of the part of the outer face is shown in Fig. 4.

A junction is labelled as being boundary if one of its half-edges belongs to the outer face. A vertex is a boundary vertex if at least one of the edges attached to it is boundary.

For simplicity, we do not consider tissues with holes, although the generalisation to such cases would be straightforward.

3.3 Handling topological changes

As the tissue evolves according to the equations of motion [Eq. (3)] cells may change neighbours. This is, however, not automatic since one needs to update the mesh connectivity. The half-edge data structure makes such changes relatively simple to implement. In Fig. 5 we show all possible connectivity changes.

For simplicity, here we focus only on the implementation of the T1 transition, i.e. cell neighbour exchanges.

Example of the T1 transition in a polygonal mesh is showed in the Fig. 6. Let us discuss in details how to implement a T1 move in a mesh represented using the half-edge data structure. First

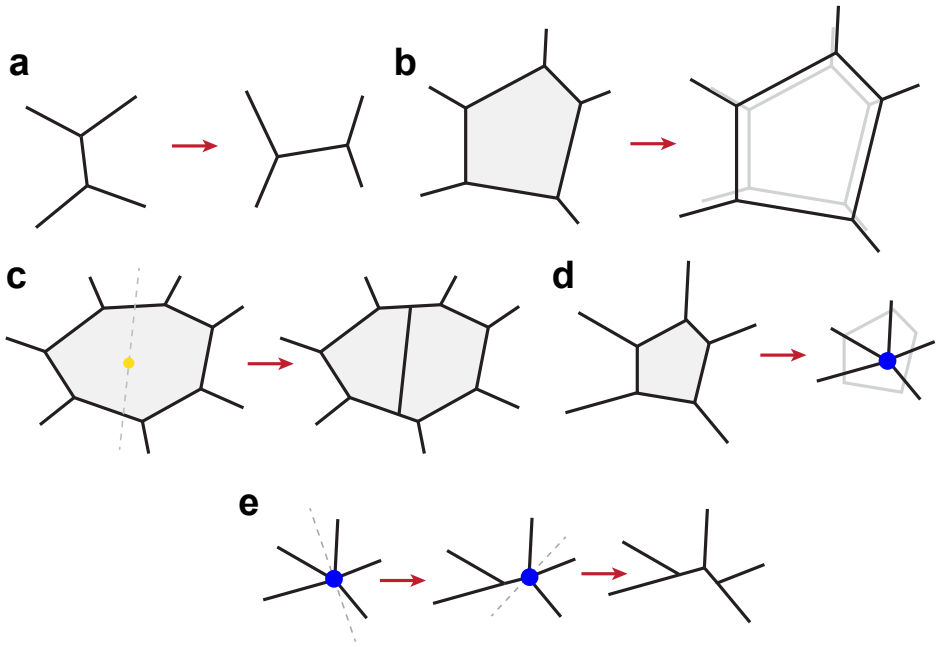

Figure 5: All possible connectivity changes. **a** Cell neighbour exchange via T1 transition, i.e. so-called cell intercalation. **b** Cell growth. **c** Cell division. **d** Cell ingression/extrusion. **e** Rosette split.

we need to identify an edge that could undergo the T1 transition. This is can be done by setting a length-based criterion, such as a minimum bond length, l_{min} below which the edge is flipped. The algorithm that determines if the edge is to flip is:

1. select half-edge *he* and it pair *hep*
2. get vertex $v_1 = he \rightarrow from\,()$
3. get vertex $v_2 = he \rightarrow to\,()$
4. get coordinates $\mathbf{r}_1 = v_1.r$ and $\mathbf{r}_2 = v_2.r$
5. if $\|\mathbf{r}_1 - \mathbf{r}_2\| < l_{min}$ mark *he* for flip.

Now that we have selected *he* for the T1 transition, we first proceed and rotate it by $\frac{\pi}{2}$ in the positive direction around its centre.

1. Get coordinates $\mathbf{r}_1$ and $\mathbf{r}_2$ (steps 2-4 in the algorithm above).
2. Find edge vector $\mathbf{l} = \frac{1}{2}(\mathbf{r}_2 - \mathbf{r}_2)$
3. Find rotated vector $\mathbf{l}^r = -l_y\mathbf{e}_x + l_x\mathbf{e}_y$
4. Find edge centre $\mathbf{r}_c = \frac{1}{2}(\mathbf{r}_2 + \mathbf{r}_2)$

5. Update positions as $v_1.r = \mathbf{r}_c - \frac{1}{2} l_{new} \mathbf{l}^r$ and $v_2.r = \mathbf{r}_c + \frac{1}{2} l_{new} \mathbf{l}^r$, where l_{new} is the length of the new edge; we must have $l_{new} > l_{min}$.

Note that at this stage we have not yet made any changes to connectivity. We therefore proceed to update connectivities:

1. Update half-edges
   (a) $v1 \rightarrow he() = he$
   (b) $v2 \rightarrow he() = hep$
2. Get neighbouring half-edges:
   (a) $he1 = he \rightarrow prev()$
   (b) $he2 = he \rightarrow next()$
   (c) $he3 = hep \rightarrow prev()$
   (d) $he4 = hep \rightarrow next()$
3. Update half-edge connectivities
   (a) $he1 \rightarrow next() = he2$
   (b) $he2 \rightarrow prev() = he1$
   (c) $he3 \rightarrow next() = he4$
   (d) $he4 \rightarrow prev() = he3$
   (e) $he \rightarrow next() = he1 \rightarrow pair()$
   (f) $he \rightarrow prev() = he4 \rightarrow pair()$
   (g) $hep \rightarrow next() = he3 \rightarrow pair()$
   (h) $hep \rightarrow prev() = he2 \rightarrow pair()$
   (i) $he1 \rightarrow pair() \rightarrow prev() = he$
   (j) $he2 \rightarrow pair() \rightarrow next() = hep$
   (k) $he3 \rightarrow pair() \rightarrow prev() = hep$
   (l) $he4 \rightarrow pair() \rightarrow next() = he$
4. Update vertices
   (a) $he1 \rightarrow to() = v2$
   (b) $he1 \rightarrow pair() \rightarrow from() = v2$
   (c) $he3 \rightarrow to() = v1$
   (d) $he3 \rightarrow pair() \rightarrow from() = v1$

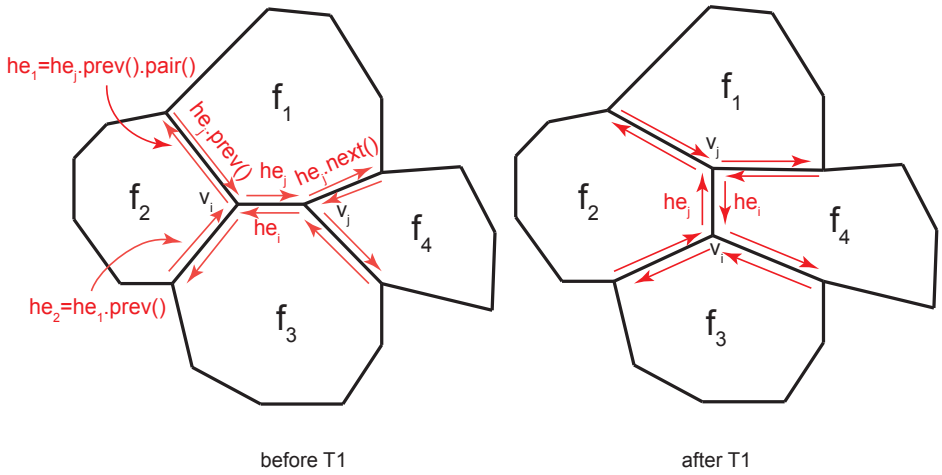

Figure 6: Example of the T1 transition. Edge e is flipped, such that faces f_2 and f_4 become neighbours while faces f_1 and f_3 are no longer connected. To save space, in the figure we used the 'dot' ($he.prev\,()$) instead of the 'arrow' notation $he \to prev\,()$.

5. Update faces

   (a) $he \to face() \to he() = he2$

   (b) $hep \to face() \to he() = he4$

   (c) $he \to face() = he1 \to pair() \to face()$

   (d) $hep \to face() = he2 \to pair() \to face()$

This concludes the move. The advantage of this approach is that it boils down to simple relabelling. No new data is created or destroyed, which means that this can be implemented very efficiently.

Finally, we note that if we use a distance-based criterion to determine when a T1 transition is going to occur we are at a risk of getting into a loop where an edge flips between many repeated back and forth transitions. In order to avoid this, we can simply blacklist an edge for a while and prevent T1 attempts for a certain number of simulation time steps.

One can use similar approaches to implement other connectivity changes of the mesh.

4 Acknowledgements

I would like to thank Daniel Barton, K. V. S. Chaithanya, Silke Henkes, Andrej Košmlj, Daniel Matoz-Fernandez, Euan MacKay, Jan Rozman, and Sijie Tong on many discussions about the vertex model.

References

[1] Silvanus Alt, Poulami Ganguly, and Guillaume Salbreux. Vertex models: from cell mechanics to tissue morphogenesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1720):20150520, 2017.

[2] Reza Farhadifar, Jens-Christian Röper, Benoit Aigouy, Suzanne Eaton, and Frank Jülicher. The influence of cell mechanics, cell-cell interactions, and proliferation on epithelial packing. *Current biology*, 17(24):2095–2104, 2007.

[3] M Lisa Manning, Ramsey A Foty, Malcolm S Steinberg, and Eva-Maria Schoetz. Coaction of intercellular adhesion and cortical tension specifies tissue surface tension. *Proceedings of the National Academy of Sciences*, 107(28):12517–12522, 2010.

[4] Dapeng Bi, JH Lopez, Jennifer M Schwarz, and M Lisa Manning. A density-independent rigidity transition in biological tissues. *Nature Physics*, 11(12):1074–1079, 2015.

[5] David E. Muller and Franco P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.