TensionMap: Image-Based Inference of Intracellular Tensions

Louis de Benoist
Wolfson College, Cambridge
ld620@cam.ac.uk

Abstract

Measuring properties of cells in vivo is a challenging problem in cellular mechanics, paving the way for vision-based approaches. We introduce TensionMap[1], a Python library for end-to-end estimation of tension in epithelial cell images. TensionMap features a novel pipeline for cell topology deduction, as well as an implementation of the VMSI algorithm for tension inference, which we then evaluate on synthetically generated data.

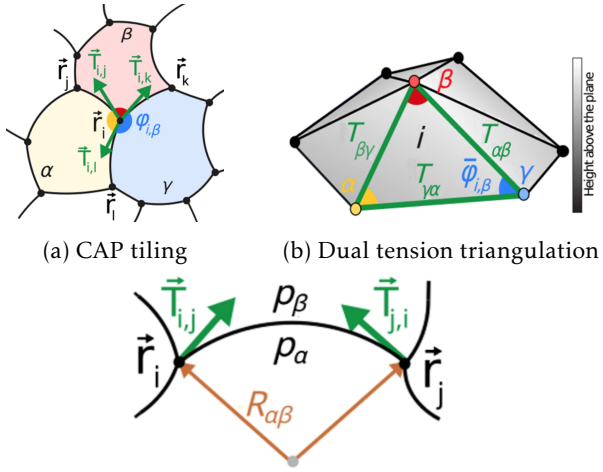
1 Introduction

Cell shape and development are governed by cell mechanics, which describes how a set of intracellular and extracellular forces determine animal and plant cell structure [5]. Cell and tissue mechanics are thus of great importance to developmental biology and, in particular, the study of embryonic morphogenesis and organ formation [8][5].

Epithelial tissue, in particular, is made up of two-dimensional monolayers with junctions between adjacent cells and adhesion to the substrate extracellular matrix [8]. The behavior of these monolayers is guided by the tissue-wide network formed by cytoskeletal cortices coupled by intercellular adherens junctions.

An epithelial cell's shape at any given instant in time is determined by two factors: the intracellular osmotic pressure, which opposes any decrease in the cell's volume caused by adjacent cells, and the balance of local actomyosin cytoskeletal contractility [8][5][9].

One of the main challenges with measuring mechanical properties of cells, such as tension, is that they are usually destructive (for example, UV laser ablation) [4][1][8].



(a) CAP tiling (b) Dual tension triangulation

(c) Young Laplace law

Figure 1: Cap tiling, its associated dual tension triangulation, and the elements which determine cell shape according to the Young-Laplace law [8]. As we can see, the tension triangulation has a z-component.

Given the wide availability of live imaging data, this has paved the way for computer vision based alternatives. Different methods have been developed, notably some based on polygonal tiling parametrizations of the two-dimensional cell geometry [3]. Variational Approach to Stress Inference (VMSI), the core of our inference pipeline and the current state of the art, developed by Noll et al., explores the mathematical duality between the cell array geometry and the non-planar triangulation formed by the equilibrium values of interfacial tensions [8]. They then exploit this duality to obtain the tensions from a segmented cell graph [8].

In this paper, we introduce TensionMap, the first Python library to provide an end-to-end estimation of tensions from images of epithelial cells. Our contributions are threefold. Firstly, we introduce a new,

[1] https://github.com/Louis2B2G/TensionMap

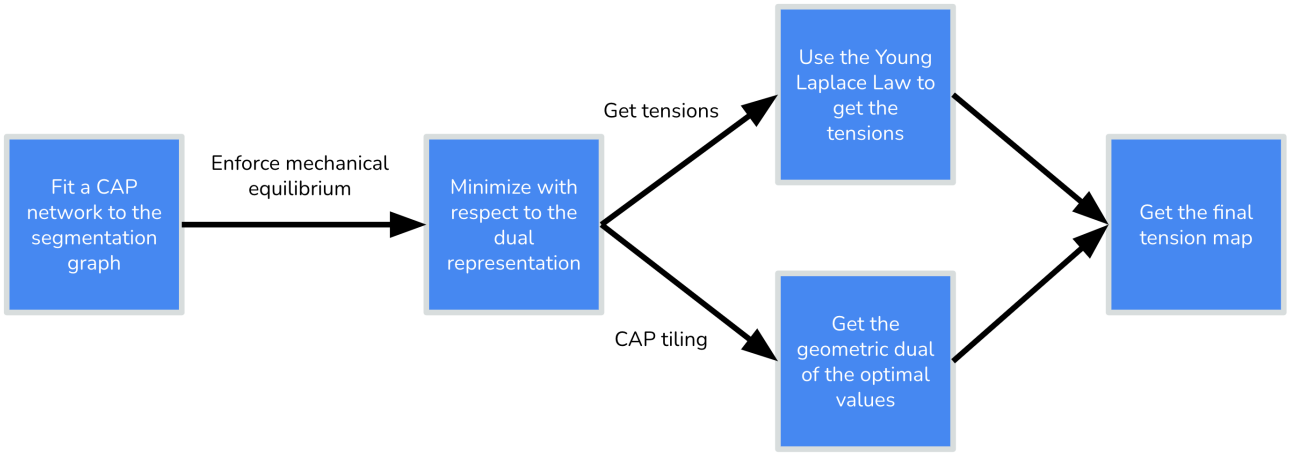


Figure 2: VMSI Overview - the inference process relies on the minimization of the pseudoenergy with respect to the dual representation, followed by back-converting to the geometric representation, applying Young's law to get the tensions, and putting it all together.

efficient pipeline for segmenting the cells and deducing the associated cell graph. Secondly, we provide an implementation of the VMSI tension inference algorithm. Finally, we provide a way to generate synthetic data and, using synthetic examples, show that our VMSI implementation does indeed accurately estimate tensions.

Most importantly, TensionMap is well documented and intuitive, making it easy to use for both biologists and physicists who wish to quickly estimate tensions in images of epithelial tissue.

2 Background

In this section, we outline the physics behind the mechanical equilibrium of two-dimensional cell arrays, as well as the VMSI algorithm, which enables the inference of tensions from cell graphs. For more detailed derivations, refer to the original VMSI paper [8].

2.1 CAP Geometry and Tension Duality

Like in the VMSI paper, we assume that epithelial cells can be represented by a planar polygonal tiling parametrized by vertices $\mathbf{r_i}$, points at the intersection of multiple cells.

In particular, as can be seen in Figure 1.a, we suppose that a circular arc polygonal (CAP) tiling can be used to represent the edges in mechanical equilibrium.

More specifically, for any two adjacent cells α and β with vertices $\mathbf{r_i}$ and $\mathbf{r_j}$, the edge can be viewed as the arc of a circle with center $\rho_{\alpha\beta}$ and radius $R_{\alpha\beta}$ between the two endpoints, as we can see in Figure 1.c. The curvature of this edge is determined by the difference in pressures between the two cells, $p_\alpha - p_\beta$ and the interfacial tension $T_{\alpha\beta} = |\mathbf{T}_{i,j}| = |\mathbf{T}_{j,i}|$, a relationship given by the Young-Laplace law [2][8],

$$R_{\alpha\beta} = \frac{T_{\alpha\beta}}{p_\alpha - p_\beta} \tag{1}$$

At any vertex $\mathbf{r_i}$, mechanical equilibrium implies that the tension vectors (as can be seen in Figure 1.a) must add up to zero,

$$\mathbf{T}_{i,j} + \mathbf{T}_{i,k} + \mathbf{T}_{i,l} = 0 \tag{2}$$

which gives rise to a tension triangle — doing so for every vertex yields a triangulated surface, the dual tension triangulation which can be seen in Figure 1.b. As we will see later, this duality between the CAP tiling and the tension triangulation is key. A direct relationship between the two representations can be obtained by applying the sine law to each triangle, i.e.

$$\frac{T_{\alpha\beta}}{T_{\alpha\gamma}} = \frac{\sin(\varphi_{i,\gamma})}{\sin(\varphi_{i,\beta})} \tag{3}$$

where α, β, and γ are the adjacent cells which define triangle i in Figure 1.b. Then, for any cell α, we can take the product over its vertices $\mathcal{V}_\alpha$ and obtain the following constraint on the angles in the CAP tiling,

$$\prod_{i\in\mathcal{V}_\alpha} \frac{T_{\alpha\beta}}{T_{\alpha\gamma}} = \prod_{i\in\mathcal{V}_\alpha} \frac{\sin(\varphi_{i,\gamma})}{\sin(\varphi_{i,\beta})} = 1 \tag{4}$$

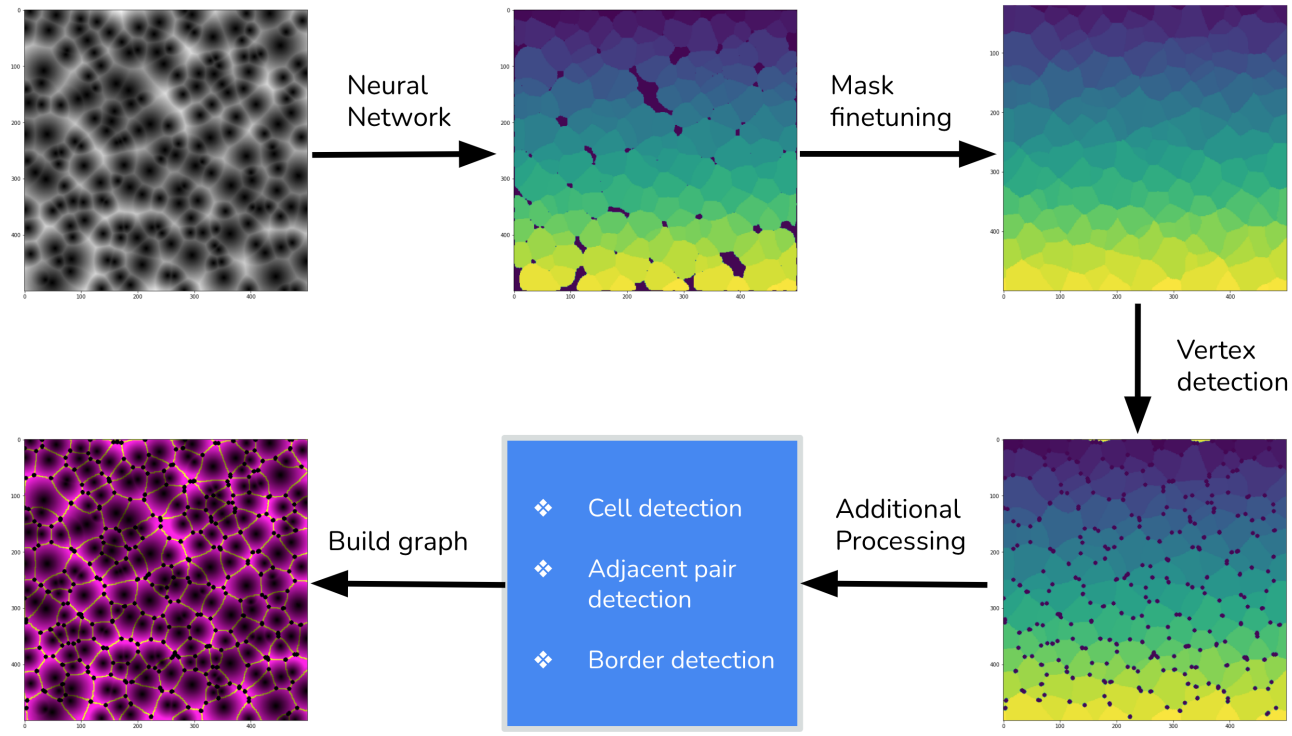


Figure 3: Overview of the cell segmentation and graph deduction pipeline

For some cell α, we obtain the relationship in Equation 1 for every edge. At vertex $\mathbf{r_i}$, Equation 1 implies that

$$\frac{T_{\alpha\beta}}{R_{\alpha\beta}} + \frac{T_{\beta\gamma}}{R_{\beta\gamma}} + \frac{T_{\gamma\alpha}}{R_{\gamma\alpha}} = 0 \tag{5}$$

Combining this with the constraint in Equation 4, we can deduce the following geometric constraint,

$$\frac{\sin(\varphi_{i,\gamma})}{R_{\alpha\beta}} + \frac{\sin(\varphi_{i,\alpha})}{R_{\beta\gamma}} + \frac{\sin(\varphi_{i,\beta})}{R_{\gamma\alpha}} = 0 \tag{6}$$

2.2 VMSI

Now that we have constraints and a correspondence between the dual representations, we can introduce the tension inference mechanism, VMSI. One of the key assumptions is that cells are in mechanical equilibrium; however, that is rarely the case, which is why we consider fitting an equilibrium tiling via least squares. The main conceptual flow of VMSI can be seen in Figure 2.

Given a segmented cell graph (i.e. an image where we have identified the cells and the edges between them), we wish to fit an equilibrium CAP tiling, i.e. a set of arcs with centers $\boldsymbol{\rho}_{\alpha\beta}$ and radii $R_{\alpha\beta}$ that each correspond to an edge. This can be done by minimizing the distance between each pixel in each edge and the corresponding pixel in the fitted arc. The pseudoenergy function to be minimized with respect to the centers and radii is thus

$$\boldsymbol{E}(\boldsymbol{\rho}, R) = \frac{1}{n_e} \sum_{(\alpha,\beta)} \sum_{k}^{N_{\alpha\beta}} (\|\boldsymbol{r}_{\alpha\beta}(k) - \boldsymbol{\rho}_{\alpha\beta}\| - R_{\alpha\beta})^2 \tag{7}$$

where n_e is the number of edges, $\mathbf{r}_{\alpha\beta}(k)$ is the position of the k^{th} pixel in the edge between cells α and β, and $N_{\alpha\beta}$ is the number of pixels in that edge. We abused notation for $\boldsymbol{\rho}$ (likewise for R) so that $\boldsymbol{\rho}_{\alpha\beta}$ denotes the two-dimensional position of the center.

This pseudoenergy, however, does not constrain the CAP tiling to be in mechanical equilibirum. To do so, we need to transpose Equation 7 in terms of physical values — i.e. the dual representation — with which we can impose such constraints. In the VMSI paper, it is shown that the change of variables,

$$\boldsymbol{\rho}_{\alpha\beta} = \frac{p_\beta \boldsymbol{q}_\beta - p_\alpha \boldsymbol{q}_\alpha}{p_\beta - p_\alpha} \tag{8}$$

$$R_{\alpha\beta} = \sqrt{\frac{p_\alpha p_\beta \|\boldsymbol{q}_\alpha - \boldsymbol{q}_\beta\|^2}{(p_\alpha - p_\beta)^2} - \frac{p_\alpha z_\alpha^2 - p_\beta z_\beta^2}{p_\alpha - p_\beta}} \tag{9}$$

forces the resultant CAP network to be in mechanical equilibrium [8]. The minimization of Equation 7

is then in terms of $(\boldsymbol{q}_\alpha, z_\alpha, p_\alpha)$, the dual representation. After the minimization, in order to recover the tensions, we can simply apply the Young-Laplace law, i.e. Equation 1.

3 Segmentation and Cell Graph Deduction

In order to apply the VMSI algorithm and deduce the tensions, we first need a segmentation graph of the cells in the image. In particular, adjacent cells need to be identified, along with the borders separating them. In TensionMap, we introduce a new pipeline for automating this process.

As can be seen in Figure 3, the pipeline consists of (1) using a neural network to obtain rough segmentation masks and fine-tuning them to get a valid and accurate segmentation, (2) finding the vertices, and (3) finding the adjacent cell pairs and detecting the borders between them.

Algorithm 1 Finding the closest mask to a point

Input: masked image I with some holes, some point $x \in I$ such that $I(x) = 0$.

steps_before_rotating $\leftarrow 0$
$\delta \leftarrow -1$
current_point $\leftarrow x$

while True **do**
  for $j \in \{0, 1\}$ **do**
    for $k \in \{0, \cdots, \text{steps\_before\_rotating} - 1\}$ **do**
      if $j = 0$ **then** ▷ step in the y direction
        current_point $\leftarrow$ current_point + $\binom{0}{\delta}$
      end if
      if $j = 1$ **then** ▷ step in the x direction
        current_point $\leftarrow$ current_point + $\binom{\delta}{0}$
      end if
      if current_point is not out of bounds **then**
        if I(current_point) $\neq 0$ **then**
          return current_point
        end if
      end if
    end for
  end for
  $\delta \leftarrow -\delta$
  steps_before_rotating $\leftarrow$ steps_before_rotating + 1
end while

3.1 Image Segmentation

The first step consists of segmenting the image, i.e. having a segmentation mask for each cell.

Getting the Initial Mask

Given the variety of possible cell resolutions, shapes, and lighting conditions, hand-crafted methods — for example, those based on the Watershed algorithm [7] — tend to fail. As a result, we instead use Cellpose, a convolutional neural network that was pretrained on over 70 000 cells [10]. One of the advantages of this method is that it has been extensively trained and used and, as a result, is quite robust. However, as we can see in Figure 3, it typically fails to fully segment the image, leaving out non-negligible spots that do not belong to any mask.

Mask Finetuning

Consequently, we introduce an additional step which takes the output of the neural network and fills in the holes by assigning every point in the image to the nearest mask.

To do this efficiently, for every point that does not belong to a mask, we iteratively build a spiral around it until a mask is reached. After this, every point in the spiral is also assigned to this mask. Algorithm 1 details the part of the algorithm that assigns a mask to a point whose value is 0 in the masked image.

In addition, we order the cells according to their geometric center's distance from the origin — this allows for identifiability since otherwise the cell order is random (as the masks have no inherent ordering).

3.2 Vertex Deduction

Once we have segmentation masks for each cell, we need to identify vertices, points where multiple cells come together. This can be done by convolving the masked image with $\mathbf{K}$, a filter working on the 3×3 neighborhood of each pixel, defined as

$$\mathbf{K}(x) = \mathbf{Card}\left(\{\mathcal{N}(x)\}\right) \tag{10}$$

where $\mathcal{N}(x)$ returns all of the neighbors of x in a 3×3 grid. Since each mask is a different color, we can simply count the number of different values that can be found in the neighborhood of x. Deducing the vertices after the convolution becomes trivial: if x is a vertex, then the value of $\mathbf{K}(x)$ will be greater than or equal to 3. A byproduct of this step is obtaining an outline of the graph (edges have a value of 2, vertices a value of 3 or above, and the rest is 1).

3.3 Building the Graph

With the masks and the vertices, we can build a cell graph in which the nodes are the cells and the edges are the segmented borders between them.

Finding Adjacent Cells

Using the vertices, we can easily find adjacent cells. To do so, we simply loop through the vertices and, for each vertex, store every combination of unique neighbors (once again, in the 3×3 neighborhood) in a set.

Algorithm 2 Finding the border between two cells

Input: masked image I, cell colors α and β.

// zero out everything except the two cells
$I(x) \leftarrow 0 \quad \forall x$ such that $I(x) \notin \{\alpha, \beta\}$

// to find the border, count the values in each point's neighborhood (removing 1 if the value is 0)
kernel: $x \mapsto$ set(x.flatten()).length - x.flatten().count(0)

// Create a bounding box over which we convolve
non_zero $\leftarrow \{x \mid I(x) \neq 0\}$
min_x (resp. max_x) $\leftarrow$ max (resp. min) x value in non_zero
min_y (resp. max_y) $\leftarrow$ max (resp. min) y value in non_zero
border $\leftarrow I$ restricted to the box defined by [min_x, max_x] and [min_y, max_y]
border $\leftarrow$ convolution(border, kernel)
border_points $\leftarrow \{x \in \text{border} \mid \text{border}(x) = 2\}$

// shift back up outside of the box
for point $\in$ point + border_points **do**
    point $\leftarrow \binom{\min\_x}{\min\_y}$
    return border_points

Detecting Borders

The last step in the graph deduction pipeline is to associate an edge to every pair of adjacent cells. For each pair, the process is as follows. We first zero out every pixel in the masked image except for those belonging to the two cells. Then, we create a variant of the filter $\mathbf{K}$, which we denote $\mathbf{K}'$ and define as

$$\mathbf{K}'(x) = \mathbf{Card}\left(\{N(x)\} - \{0\}\right) \tag{11}$$

Convolving $\mathbf{K}$ with the image containing the two masks enables us to only select points in whose neighborhoods a color change happens that excludes those involving 0 (i.e. the other borders of the masks in question). Those points, which form the edge between the two cells, will be the only ones to have a value of 2.

For performance reasons, we first find the smallest bounding box around the two selected masks, apply the convolution to the reduced image in order to obtain the edges, and then rescale the values back up to get the final result. The algorithm is more precisely detailed in Algorithm 2.

4 Tension Inference

In this section, we outline the main process implemented in TensionMap in order to infer the tensions using the VMSI method.

4.1 Pseudoenergy Minimization

The minimization of Equation 7 after the change of variables is quite straightforward — we simply collect all of the input variables into a single vector θ which we then extract inside of the function to be minimized.

4.1.1 Initialization

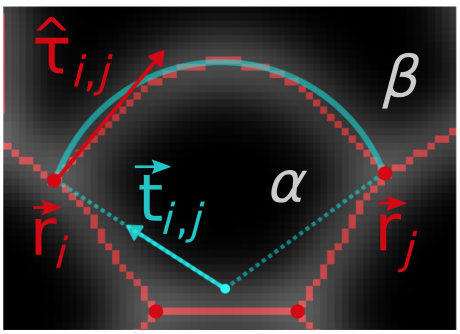


Figure 4: Values used in the minimization to initialize the values of $\boldsymbol{q}_\alpha$ and p_α [8]

However, the minimization is very localized and is thus quite sensitive to the initialization. In the VMSI paper, they recommend performing an additional minimization to add additional structure to the initial points [8].

The idea is to enforce orthogonality between the vectors pointing from the center of the circular arc $\boldsymbol{t}_{i,j}$ to the vertices and the vectors tangent to the edge at the same points $\boldsymbol{\tau}_{i,j}$, as can be seen in Figure 4.

In terms of the minimization parameters, $\boldsymbol{t}_{i,j}$ can be written as

$$\boldsymbol{t}_{i,j} = (p_\alpha - p_\beta) r_i - (p_\alpha \boldsymbol{q}_\alpha - p_\beta \boldsymbol{q}_\beta) \tag{12}$$

using the equations to switch between the dual representations introduced in the Background section. The minimization to initialize $\boldsymbol{q}_\alpha$ and p_α is thus,

$$\boldsymbol{E}_{\text{initial}}(\boldsymbol{q}, p) = \frac{1}{n_e} \sum_{(\alpha,\beta)} \langle \boldsymbol{t}_{i,j}, \boldsymbol{\tau}_{i,j} \rangle^2 + \langle \boldsymbol{t}_{j,i}, \boldsymbol{\tau}_{j,i} \rangle^2 \tag{13}$$

which simply tries to make the center of the corresponding CAP tiling as orthogonal as possible to both of the vertices in any given cell edge. In practice, in

order to find the tangent vectors, $\boldsymbol{\tau}_{i,j}$ and $\boldsymbol{\tau}_{j,i}$, we used the normalized vector passing between each vertex and the mean of the 5 closest other points on the edge. The initialization that we used to minimize $\boldsymbol{E}_{\text{initial}}$ is setting $\boldsymbol{q}_{\alpha_0}$ to be the cell's barrycenter and uniformly sampling p_{α_0}.

However, this minimization by itself is not very useful — indeed, we wish to avoid the trivial solution: $\boldsymbol{q}_\alpha = \boldsymbol{q}_\beta$ and $p_\alpha = p_\beta$. To avoid this, we add the following constraint,

$$\frac{\frac{1}{2n_e}\sum_{(\alpha,\beta)}\|\boldsymbol{t}_{i,j}\|}{\frac{1}{n_e}\sum_{(\alpha,\beta)}|p_\alpha - p_\beta|} = \sum_{(\alpha,\beta)}\frac{\frac{1}{2}\|\boldsymbol{t}_{i,j}\|}{|p_\alpha - p_\beta|} = R_{\text{avg}} \tag{14}$$

which imposes that the ratio between the average norm of $\boldsymbol{t}_{i,j}$ and the average pressure difference is the same as the average radius of curvature in the image R_{avg}.

R_{avg} only needs to be computed once. To obtain it, in our implementation we chose to minimize the following function via the least squares method for each edge $\mathcal{E}$,

$$\boldsymbol{\rho}_{\mathcal{E}} = \text{argmin}_{y\in\mathbb{R}^2}\textbf{std}\left(\{\|x-y\|\}^2_{x\in\mathcal{E}}\right) \tag{15}$$

where we simply minimize the sample variance of the distances from the centroid that we are trying to fit. Essentially, we suppose that the points on the edge were sampled from a circle — we then maximize the likelihood of the data. The corresponding radius is the mean distance from the fitted centroid. Then, R_{avg} can be found by averaging the radii corresponding to the fitted $\boldsymbol{\rho}_{\mathcal{E}}$ over every edge $\mathcal{E}$.

To initialize z_α, we can then simply use Equations 8 and 9 alongside the fitted $\boldsymbol{q}_\alpha$ and p_α.

4.2 CAP Tiling

Once the minimization is complete, given the optimal values of $(\boldsymbol{q}_\alpha, z_\alpha, p_\alpha)$, we can use Equations 8 and 9 to obtain corresponding centers and radii.

Figuring out which part of the circle to plot involves (1) sampling some predetermined number of points from the circle (2) identifying the points x_1 and x_2 on the circle which minimize the distance to the two vertices, and (3) avoiding edge cases by finding the arc made up of x_1 and x_2 of minimal length.

Once the arc has been defined, its color can then be set according to the tension (which can be found using the Young-Laplace law in Equation 1), as we can see in Figure 6.

5 Experiments and Results

In this section, we demonstrate how TensionMap can be used and evaluate it on both synthetic and real data.

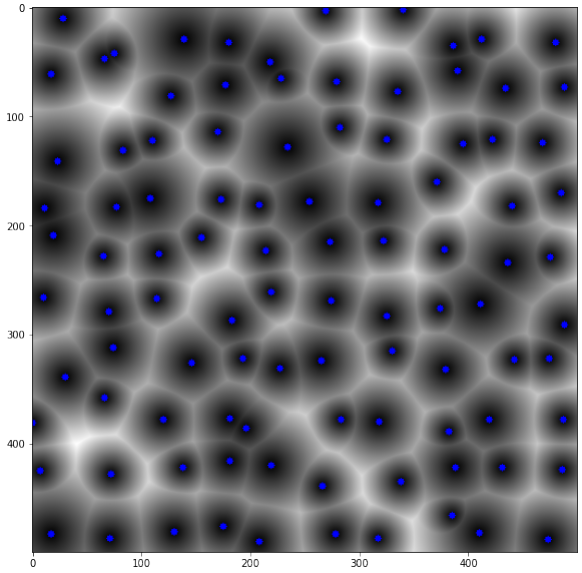


Figure 5: Synthetic cells induced by random generating points q_α (colored in blue) with pressures uniformly sampled from a user-defined interval.

5.1 Generating Synthetic Data

Our TensionMap library also has a built-in synthetic data generator. This is to test the performance of the VMSI inference algorithm. Indeed, the idea is to generate images using known pressure values and positions of generating points. Then, after segmentation, we take the tensions from the inferred CAP tiling (using the true q_α, z_α, and p_α values used to generate the image) as the ground truth. We can then compare these tensions with those obtained using the predicted q_α, z_α, and p_α via the VMSI.

In short, we generate points in the dual representation space and attempt to recover them using VMSI.

5.1.1 Generalized Voronoi Construction

Generating the synthetic images can be done via a generalized Voronoi construction. First, we randomly sample some generating points $(\boldsymbol{q}_\alpha, z_\alpha, p_\alpha)$ according to some user-defined distributions (typically uniform).

Then, the grayscale value of each pixel will be the distance to (non-Euclidean) the nearest generating point. The distance not only has to take into account q_α, but it also needs to incorporate z_α and p_α.

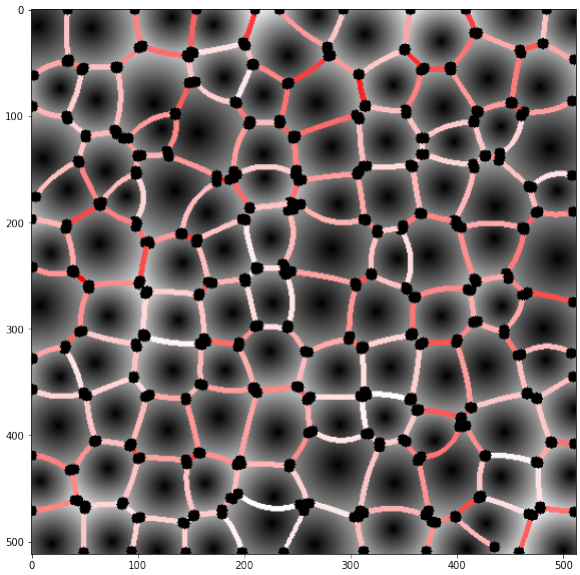


Figure 6: Example of a tension map output on synthetic data. The more red, the more tension is present on the edge.

The distance between some pixel $\boldsymbol{r}_i \in \mathbb{R}^2$ and a generating point α is scaled by p_α as follows

$$d_\alpha(\boldsymbol{r}_i) = \sqrt{p_\alpha(\|r_i - q_\alpha\|^2 + \|z_\alpha\|^2)} \tag{16}$$

Then, the pixel intensity $I(\boldsymbol{r}_i)$ is given by

$$I(\boldsymbol{r}_i) = \min_\alpha d_\alpha(\boldsymbol{r}_i) \tag{17}$$

The resulting image will thus have an edge $\mathcal{E}_{\alpha\beta}$ between cells α and β that is given by

$$\mathcal{E}_{\alpha\beta} = \left\{\boldsymbol{r}_i \mid \sqrt{p_\alpha} d_\alpha(\boldsymbol{r}_i) = \sqrt{p_\beta} d_\beta(\boldsymbol{r}_i)\right\} \tag{18}$$

As we can see, if every $p_\alpha = 1$ and every $z_\alpha = 0$, then the image reduces to a standard Voronoi diagram.

5.2 Results

In order to test the validity of the method, we used synthetically generated data and compared the predicted tensions with the true values, as discussed in the previous section. As we can see in Figure 7, there is a high correlation between the two two sets of tensions, confirming that our implementation of VMSI is indeed correct.

In addition to using synthetic data, we also evaluated our pipeline on real data obtained from embryonic development [6]. To illustrate the robustness of our method, we purposely picked a blurry, low resolution image with multiple cells with odd shapes.

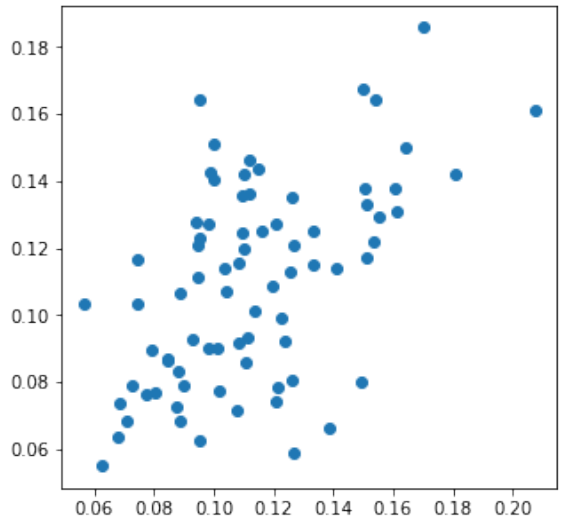
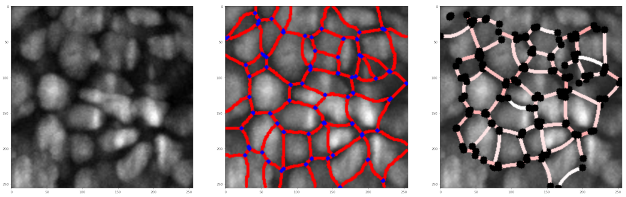


Figure 7: Correlation plot between the predicted tensions and the actual tensions. Pearson correlation coefficient of 0.54



(a) Original (b) Segmentation (c) Tension map

Figure 8: Segmentation and tension map output next of a real image of epithelial cells (embryonic development).

As we can see in Figure 8, our segmentation tool provides a clean segmentation of the image, while the VMSI tension inference provides a reasonable inference of the underlying tensions — however, as previously mentioned, the only real insights as to the validity of the method can be taken from synthetically generated tests (since we do not know the true tensions for the real images). Upon closer inspection, we can see some mistakes in the segmentation and, thus, in the tension map output (for example, some larger cells are mistaken as two smaller ones). This is most likely due to the low resolution and image quality of the input, as mentioned before.

6 Conclusion

We have introduced TensionMap, the first end-to-end Python library for estimating tension from images of

epithelial tissue via the VMSI inference algorithm. In addition to this novel implementation, we have contributed a new, robust cell graph deduction pipeline to use in conjunction with VMSI as well as a tool to quickly and efficiently generate synthetic data.

Appendix

A Basic Usage

In this section, we briefly show the few lines of code that are necessary to automatize the whole process.

A.1 Generating Synthetic Data

Once the library is imported, one can generate and visualize synthetic data as follows,

```
config = {
          'max_x' : 256,
          'max_y' : 256,
          'min_p' : 0.001,
          'max_p' : 0.004,
          'num_points' : 50
         }

generating_points = Points(**config)

dtr = DistanceTransform(generating_points)
dtr.compute_transform()
dtr.visualize_transform()
```

A.2 Cell Graph Segmentation

In order to obtain the segmented output, compute everything necessary to use the VMSI inference, and view the segmentation, one can do

```
seg = Segmenter()
seg.segment(image, diameter=None)
seg.visualize('outlines', overlay=True)
```

If one has a prior estimate for the average diameter of the cells, one can input it as an argument (which will speed up the segmentation) — otherwise, it will be automatically deduced.

A.3 VMSI and CAP Tiling

In order to infer the tension maps from the cell graph segmentation, the code is as follows

```
model = VMSI(cell_pairs = seg.pairs(),
             edges = seg.edges(),
             num_cells = len(seg.cells[0]),
             cells = seg.cells[0],
             barrycenters = seg.barrycenters[0],
             edge_cells = seg.get_edge_cells(),
             height=256, width=256)

q, z, p = model.fit()
CAP_tiling = model.CAP(image, q, z, p)
```

References

[1] I. Bonnet et al. “Mechanical state, material properties and continuous description of an epithelial tissue”. In: *J R Soc Interface* 9.75 (2012), pp. 2614–2623.

[2] G. W. Brodland et al. “CellFIT: a cellular force-inference toolkit using curvilinear cell boundaries”. In: *PLoS One* 9.6 (2014), e99116.

[3] H. Honda, H. Yamanaka, and M. Dan-Sohkawa. “A computer simulation of geometrical configurations during cell division”. In: *J Theor Biol* 106.3 (1984), pp. 423–435.

[4] M. S. Hutson et al. “Forces for morphogenesis investigated with laser microsurgery and quantitative modeling”. In: *Science* 300.5616 (2003), pp. 145–149.

[5] Thomas Lecuit and Pierre-François Lenne. “Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis”. In: *Nature Reviews Molecular Cell Biology* 8.8 (2007), pp. 633–644. DOI: 10.1038/nrm2222. URL: https://doi.org/10.1038/nrm2222.

[6] Katie McDole et al. “In Toto Imaging and Reconstruction of Post-Implantation Mouse Development at the Single-Cell Level”. In: *Cell* 175.3 (2018), 859–876.e33. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2018.09.031. URL: https://www.sciencedirect.com/science/article/pii/S0092867418312431.

[7] Fernand Meyer. *The watershed concept and its use in segmentation : a brief history*. 2012. arXiv: 1202.0216 [cs.CV].

[8] Nicholas Noll, Sebastian J. Streichan, and Boris I. Shraiman. “Variational Method for Image-Based Inference of Internal Stress in Epithelial Tissues”. In: *Phys. Rev. X* 10 (1 2020), p. 011072. DOI: 10.1103/PhysRevX.10.011072. URL: https://link.aps.org/doi/10.1103/PhysRevX.10.011072.

[9] Martin P Stewart et al. “Hydrostatic pressure and the actomyosin cortex drive mitotic cell rounding”. In: *Nature* 469.7329 (2011), 226—230. ISSN: 0028-0836. DOI: 10.1038/nature09642. URL: https://doi.org/10.1038/nature09642.

[10] Carsen Stringer, Michalis Michaelos, and Marius Pachitariu. “Cellpose: a generalist algorithm for cellular segmentation”. In: *bioRxiv* (2020). DOI: 10.1101/2020.02.02.931238. eprint: https://www.biorxiv.org/content/early/2020/02/03/2020.02.02.931238.full.pdf. URL: https://www.biorxiv.org/content/early/2020/02/03/2020.02.02.931238.