Assignment 3 – Cellular Structure and Dynamics

CS 279, BIOE 279, BMDS 245, BIOPHYS 279, and CME 279

Due: Thursday, Nov 12, 2026 at 1:00 PM

In the previous assignments, we've talked about the structure of biomolecules at an atomic resolution. The goal of this assignment is to discuss structure and organization at the cellular level, and to gain an appreciation for the techniques we can use to learn about these structures.

Acknowledgements: Many thanks to Prof. Julie Theriot and Prof. Zach Pincus for access to the data sets used in this assignment. For more on their methods and software, check out:
Pincus Z & Theriot JA. Comparison of quantitative methods for cell-shape analysis. J Microscopy 227 pp. 140–156 (2007).

1 Preliminaries

- **Computing options:** We have found that software installation for Assignment 3 can be challenging for students. As a result, we **strongly recommend** using the Stanford LTS computers, which have all the necessary software pre-installed. See the Assignment 3 Software Guide for platform-specific setup instructions. Our suggestions for computing options in order of preference are:

    1. Using in-person LTS machines: We strongly recommend using these machines for this assignment. Here are links to the full list of LTS locations and the library hours for using the in-person LTS machines. The LTS machines in residences are available 24/7, as are the machines in the Lathrop study room. Once logged into the LTS machine, download the assignment files from the course website. **Make sure to save your files to Google Drive or an external drive. Files downloaded onto the LTS machines are deleted upon logging out.**
    2. Self-installed software: See the Assignment 3 Software Guide for details on software installation and setup. **Note:** Installing packages directly on Windows can be challenging, and we may be unable to provide support with this process on Windows. Mac, WSL on Windows, and Linux users have a better chance of success with self-installation, but it has proven difficult for many students in the past.
    3. Remote LTS access: Connect at https://stanford.apporto.com/. The number of LTS machines available for remote access is **very** limited, so you may or may not be able to access a machine at any given time, particularly shortly before the due date. **You should use in-person LTS machines if you live, work, or take classes on campus. Only fully remote students should use remote LTS machines.**

- **LTS users**: Please make sure to use the command "python3.9" instead of "python" for every block of code where "python" is currently written.
- We'll ask you to submit screenshots of a few functions throughout the assignment. Please only submit the functions we request. We do not want any other code submitted.

2 Cellular Structure

What do we mean when we refer to cellular structure and organization? Unlike the atoms within a biomolecule, the molecules within a cell do not typically have 3D coordinates that are conserved within all individual cells of a given type. We know that the overall shape of a given cell (and the organization of bio-machinery within that cell) is strongly tied to its functionality and chemical environment.

Humans can observe the variations in cellular structure that occur across cell types, between mutants, and after changes in chemical environment. These observations make for a strong qualitative analysis, but is it possible to develop a quantitative method for analyzing the structure of cells that captures this human intuition? Modern image processing techniques and basic statistical procedures allow us to extract meaningful features of different cell types.

2.1 Microscopy Image Analysis

We'll begin by displaying some of the microscopy images we will be analyzing. We'll be working with three sets of images in this assignment. The first image corresponds to a mitochondrial membrane structure for rod and cone mitochondria. The second set of images is of Caulobacter, a bacterium which is often used as a model organism to study the cell cycle and differentiation. The third set of images is of fish epidermal keratocytes, fast-moving skin cells from fish scales that crawl across surfaces and are a classic model for studying how cells move and control their shape.

Many of these are grayscale images, where each pixel value represents the observed intensity at the corresponding (x, y) point. There are many ways to look at images, but we provide you with a simple script in your `imgs` folder to display a given file.

To see the images for this assignment, `cd` into the assignment starter code folder, then `cd` into the `imgs` folder in terminal. Then call the provided show.py file to look at the images in mitochondria, caulobacter, and keratocytes folders. Run it on every image in all three folders, including the `.tif` keratocyte images (for example, `python show.py keratocytes/A.tif`).

Usage:

```
python show.py <path to image file>
```

Note: You may have to replace `python` with `python3.9` in order to point to the correct python version.

Now, let's work on doing some image analysis. We'll start by trying to denoise the images using some of the techniques discussed in class (Image Analysis Slides). In the next few problems and exercises, you will construct various filters. To run these filters on images, use the `filter.py` script from your `imgs` folder.

Usage:

```
python filter.py <path to image file> <filter name>
```

For Exercises 1–4 and Questions 1–3, use the mitochondrial image (`mitochondria.png`) in the `mitochondria` folder. The commands in `filter.py` will output the original image and the filtered image, which may appear on top of each other. Both will have a color map on the right, which maps the colors in the image to a range of values.

Exercise 1: *Implement the* `mean` *low-pass filter in* `filter.py`*. The problem documentation has more information on what to implement.*

Question 1: *Play around with the window size: try a* 3×3 *and* 6×6 *array. Test your* `mean` *low-pass filter on the mitochondrial image.*

(a) Insert your code for the `mean` *filter (a screenshot works well).*

(b) Include a screenshot of the resulting 3×3 *and* 6×6 *mean-filtered images (please include the colorbar in your screenshot).*

(c) In 1–2 sentences, why is it good practice to make the entries in the `mean` *filter sum to 1?*

Exercise 2: *On the mitochondrial image, try out* `median` *and* `gaussian` *filters with different sizes and standard deviations respectively. Try to find a size and standard deviation respectively that create an improved image (as determined by visual inspection). You do not need to submit anything for this exercise, but understanding filter parameters will help for future questions.*

Convolutional filters designed to remove noise are generally referred to as "low-pass" filters, because they smooth out the original signal, leaving the low frequencies while reducing the high frequencies. What if, instead of denoising the images, we wanted to extract an outline of the cell's structures? In this case, we want a "high-pass" filter, which will emphasize abrupt changes in intensity rather than gradual changes.

Exercise 3: *Implement the* `hp` *and* `gaussianHP` *filters.*
The `hp` *filter will convolve the original image with a matrix such as:*

$$\begin{bmatrix} -c & -c & -c \\ -c & +8c & -c \\ -c & -c & -c \end{bmatrix}$$

The `gaussianHP` *filter exploits the fact that the original image minus a Gaussian low-pass filter gives a reasonable high-pass filter.*

Question 2: *Test your* `hp` *and* `gaussianHP` *filters on the mitochondrial image (the same image as Question 1).*

(a) Include your code for the `hp` *filter.*

(b) Try different values of c. What happens to the image as you change the value of c? Include a screenshot of the output of your `hp` *filter, and include the final value of c that you choose.*

(c) Include your code for the `gaussianHP` *filter.*

(d) Include a screenshot of the output of your `gaussianHP` *filter.*

(e) Note that in the simple `hp` *filter, the entries in the matrix sum to 0. Why is this a good idea?*

Now, take a look at the Caulobacter images in `imgs/caulobacter`. These images were collected using phase-contrast microscopy, which gives us very good contrast between "Cell" and "Not Cell", but poor contrast within the cell. This property allows us to extract the outlines of the cells with a simple threshold filter.

Exercise 4: *Implement the* `threshold` *filter. You should select a threshold value where the contrast is clearest and then set all values above this threshold to 0 and all below this threshold to 1. This will invert the image so that the cell outline appears with a pixel value 0 (a darker color) and the background appears with a pixel value 1 (a lighter color). Note that the color representation may depend on your image color map.*

Question 3: *Test your implementation of the* threshold *filter on the mitochondrial image.*

(a) Include your code for the `threshold` *filter.*

(b) Include a representative screenshot of the output of your `threshold` *filter.*

(c) In 1–2 sentences, explain how you derived your threshold value and note what your final threshold value was.

2.2 Principal Component Analysis of Caulobacter Cells

In this section, we'll use binary masked images to analyze cell shapes with Caulobacter images. In the `masks` folder, there are sequences of binary Caulobacter images that have already been processed so that any part of the image that is a cell is completely white (= 1) and all other parts of the image are completely black (= 0).

Let's start by making some qualitative observations.

Question 4: *Describe the shapes of the cells that you see. In particular, what attributes of an individual Caulobacter cell's shape would allow you to distinguish it from the others?*

Now that we have some sense for what the cells in our data set look like, we'll use the software Celltool to help us make these qualitative observations quantitative. Celltool has tools to represent

cell shapes as outlines. The first thing we need to do is extract this outline or "contour".

Question 5:
Go into the `masks` *folder.*

```
cd masks
```

Extract the contours from `caulobacter` *as follows. If copy/pasting this command, make sure that the underscore is there and that the spacing is right so that the command line arguments are parsed correctly!*

```
celltool extract_contours \
--resample-points=100 --smoothing-factor=0.001 \
--destination=cauloContours caulobacter/*.tif
```

Then, run the following command to plot the contours into an `svg` *file.*

```
celltool plot_contours \
--output-file="cauloContours.svg" cauloContours/*.contour
```

Include a screenshot of `cauloContours.svg` *in your writeup. (It's easiest to view .svg files by opening them from your folder using Chrome or another web browser.)*

You should now have a file in the folder called `cauloContours.svg`. Take a look at this file. You should see the outlines of a number of cells laid across the image. It will also be helpful to look at one of the `.contour` files in the `cauloContours` folder. These files can be viewed in your native text editor software (e.g. TextEdit for Mac).

In particular, note that these outlines are saved as an array of (x,y) points. In order to be able to compare and contrast these vectors of points, we must align the vectors such that the ith point in each vector corresponds to an analogous point in each cell.

Exercise 5:
Align the extracted contours using the following command.

```
celltool align_contours \
--allow-reflection \
--destination=cauloAligned cauloContours/*.contour
```

Then plot these aligned contours.

```
celltool plot_contours --color-by=points \
--output-file="cauloAligned.svg" cauloAligned/*.contour
```

***Note**: You do not need to submit* `cauloAligned.svg`.

Open `cauloAligned.svg`. The cells should now be aligned, where the indices of points along one cell's outline generally correspond to analogous points on other cells' outlines. Note that a cell's contour is colored by the point ordering of its vertices. Now that we have a consistent representation of the cells, we can begin analyzing the cells quantitatively.

Before talking about our specific data, we should first discuss the main technique that Celltool will use to build a model for the cells: Principal Component Analysis (PCA). Recall from the

Formally, the first principal component is defined for a data set X centered about the origin as

follows.

$$c_1 = \underset{\|c\|=1}{\text{argmax}} \sum_{x \in X} (x \cdot c)^2$$

Recall that a dot product is greatest in absolute value when x and c point along the same axis. Thus, because we assume that X has mean 0 in every coordinate, maximizing the sum of squared dot products results in choosing a direction, c, which maximizes the overall variance after projection.

We'll see that maximizing the variance preserved from the data set allows us to reduce our representation of the data while minimizing the error that is introduced. To get a sense of what this definition means in practice, consider the following example.

Consider the spread of points in Figure 1. You can view the data more carefully on your own screen by running: `python pca/plotData.py`. Typically, we would describe the coordinates of the points as (x, y) pairs with the bottom left corner at (0,0) moving upward towards the upper right corner at (6, 5).

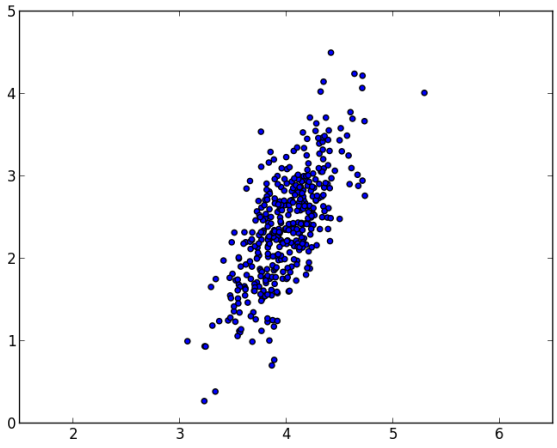

Figure 1: 2D data points

Question 6: *Suppose, to simplify our representation, we wanted to only store one value for each coordinate while maintaining our ability to distinguish between points as much as possible.*

(a) Clearly, one solution would be to save just the x-coordinate or just the y-coordinate. These solutions project the data onto the lines $y = 0$ and $x = 0$ of slope 0 and ∞, respectively. cd into your `pca` *folder. Use the* `project.py` *script in the* `pca` *folder to see what the data looks like under these projections.* ***Note****: These commands will produce two plots, which may appear stacked on top of each other.*

```
cd <assignment folder path>/pca
python project.py 0
python project.py inf
```

Look at the output of the command in your terminal window. What is the empirical variance after projection in each case?

(b) The choice to project onto $x = 0$ and $y = 0$ was arbitrary. By picking the projection line more carefully, we should be able to preserve more of the variance in the data points. Play around with different slope values. What slope, rounded to the nearest tenth, maximizes the variance preserved? What is the empirical variance of the data under this projection?

(c) The slope you found in the last part is precisely the first principal component — the line that maximizes the spread of points after projection. The second principal component is defined similarly after removing the variation in data points according to the first principal component. In particular, the next principal component must be orthogonal (perpendicular) to the first, regardless of the number of dimensions. (You should convince yourself of this fact.) For intuition: projecting onto the first principal component captures all of the spread along that line. Once that variation is removed, all that is left of each point is its offset perpendicular to the first component, so the only direction with any variance remaining is the one orthogonal to it. What are the slope and variance preserved of the second principal component?
Hint*: What's the relationship between perpendicular lines and their slopes?*

(d) Submit a plot of the data projected onto the first principal component and onto the second principal component.

Note that because the data we worked with in the previous question was only 2-dimensional, there are only 2 principal components. In general, if we are dealing with n-dimensional vectors, there will be n principal components. In the case of our cells, where we use 200 values (100 x,y pairs) to represent each cell, there will be 200 principal components, which are each 200-dimensional vectors, which capture all of the variance in the set of cell images. That said, the hope is that the majority of the variance is captured by the first few principal components. If this is the case, then we will have a succinct way of representing and discriminating between individual cells.

Now, we will jump back to using Celltool to investigate the data set of cells.

Exercise 6: *Go back to the* `masks` *folder and run the following commands, which should extract the first two principal components of cells' shape from the contours you generated before.*

```
celltool shape_model --variance-explained=0.95 \
--output-prefix="cauloModel" cauloAligned/*.contour
```

Then plot the model.

```
celltool plot_model "cauloModel.contour"
```

Question 7: *Include a screenshot of* `cauloModel.svg` *in your writeup. Qualitatively, what do the first two principal components of the Caulobacter data look like? What differences are they capturing? How well do these components correspond to the qualitative descriptions you originally described (question 4)?*

We can think about these principal components in a few ways. Firstly, this analysis shows that $> 95\%$ of the variance in Caulobacter shape corresponds to only two axes in our original 200-dimensional space. This means that, to a good first approximation, we could represent each cell as 2 numbers, rather than 200 because we only lose 5% of the variance in the population. Additionally, the correspondence between the principal components and our qualitative observations gives a rigorous statistical justification for using these observations as discriminative features. Finally, by using the principal components, we can easily generate new images of cells, which we've never seen before, but which are statistically similar to cells that we have observed.

2.3 Characterization of Keratocytes

Now we'd like to use Celltool to characterize cells based on their shape. For this section, let's use the following hypothetical situation as context. We've collected images of keratocytes grown in two different media: one in normal media and the other in media diluted to 25% the concentration. Unfortunately, we were not careful in the lab and we mixed up which images correspond to which condition. We also happened to take images of keratocytes grown in another chemical condition that causes a similar osmotic stress on the cells as growing in 25% media. Thus, these cells should look more similar to one another than to the normal cells. We'd like a quantitative method to identify which images correspond to which condition.

Exercise 7: *Inside the* `masks/keratocytes` *folder the prefix of each image corresponds to the growing conditions, unknowns A and B and known chemical environment X. Build a shape model from your* ***masks*** *folder for the whole data set that will allow you to observe differences in the cell types. Navigate to the* `masks` *folder.*

```
cd <assignment folder path>/masks/
```

Then, run the following commands to generate the model.

```
celltool extract_contours --resample-points=100 \
--smoothing-factor=0.001 \
--destination=kContours keratocytes/*/*.tif
```

```
celltool align_contours --allow-reflection \
--destination=kAligned kContours/*.contour
```

```
celltool shape_model --variance-explained=0.90 \
--output-prefix="kModel" kAligned/*.contour
```

```
celltool plot_model "kModel.contour"
```

Take a look at `kModel.svg`.
Note*: You can ignore the* `Warning: Contour alignment did not converge after 10 iterations` *warning after the* `align_contours` *command if it appears.*

Now that we have a model that characterizes the top principal components, we can plot the images as points on the axes of these components. (We'll plot along the first two principal components for ease of visualization.)

Exercise 8: *Run the following commands to measure the average area of each cell type and degree of each principal component present for each cell type.*

```
celltool measure_contours --output-file="A.csv" \
--area --shape-modes kModel.contour 1 2 - kAligned/A*.contour
```

```
celltool measure_contours --output-file="B.csv" --area \
--shape-modes kModel.contour 1 2 - kAligned/B*.contour
```

```
celltool measure_contours --output-file="X.csv" --area \
--shape-modes kModel.contour 1 2 - kAligned/X*.contour
```

Then plot the distribution of cell types along the axes of the principal components as follows.

```
celltool plot_distribution --x-column=3 --y-column=4 \
--output-file="dist.svg" A.csv B.csv X.csv kAligned/*.contour
```

Finally, plot the distribution of areas of the cells as follows.

```
celltool plot_distribution --x-column=Area \
--output-file="areas.svg" A.csv B.csv X.csv
```

Question 8: *Take a look at your outputs from Exercise 8. We'll use these files to answer Question 9.* ***Note****: If you are using an LTS machine, you may have to open* `areas.svg` *in Chrome instead of Inkscape to display the legend colors correctly.*

- *Include a screenshot of* `dist.svg` *in your writeup.*
- *Include a screenshot of* `areas.svg` *in your writeup.*

Question 9: *Which set of cells was grown in normal media and which was grown in dilute media? How did you come to this conclusion? You should refer to the plots generated in the previous exercise.*

Now that we can effectively analyze and model cellular shape, we can start to talk about the intracellular organization and dynamics.

3 Diffusion Simulations

Once we have a model for the structure of a given type of cell, we frequently want to model the dynamics within the cell. Next, we will look at how biological molecules diffuse throughout cells.

Diffusion is the process by which particles spread out and mix together with other particles. Fundamentally, diffusion is governed by the random motion of individual particles. However, by combining many particles' random motions, we can derive deterministic laws that govern the aggregate movement of a substance through solution.

Let's start by thinking about motion in one dimension.

Question 10: *Assume there is a single particle on a number line, starting at position 0, and at each time step it moves left or right by 1 unit with equal probability.*

(a) Give the probability distribution over positions at $t = 4$ and $t = 5$. In other words, list the probability of the particle being at each possible position, at $t = 4$ and $t = 5$ respectively. A two-column table (one column of position, one column of probability) has historically worked well.
Extra Credit: Give a general expression for the probability of the particle being at position $x = x'$ at time $t = t'$.

(b) What is the expected position x at time $t = t'$? What is the expected squared displacement after t' time steps?

Hint*: The* ***Diffusion and Cellular-Level Simulation*** *slides could be helpful here.*

Now, let's consider what happens when we add more particles.

Question 11: *Under the same model as the previous question, assume we have 1000 particles, all starting at position 0.*

(a) At the first time step, what is the expected number of particles that pass from $x = 0$ to $x = 1$?

(b) Let $n(x, t)$ be the number of particles at position x at time t. After 1 time step, what is the expected number of particles that pass from $x = x_0$ to $x = x_0 + 1$? What is the expected number of particles that pass from $x = x_0 + 1$ to $x = x_0$? And combining these two numbers, what is the net movement of particles from $x = x_0$ to $x = x_0 + 1$?
Your answers should be expressed in terms of $n(x, t)$.

This result underpins "Fick's Law," which states that the flux (the net movement of particles across a unit area per unit time) is proportional to the concentration gradient.

$$J = -D\frac{\partial C}{\partial x}$$

where J is the flux, C is the concentration, and D is a diffusion coefficient and is given in units of (distance)^2 per time.

While it is important to understand that particles move from high concentrations to low concentrations in space — and that this fact can be derived from a simple random motion model — what we'd really like is a way to express the change in concentrations over time. If we assume that particles are neither lost nor created, then the change in concentration over time must equal the negative of the change in flux over distance.

$$\frac{\partial C}{\partial t} = -\frac{\partial J}{\partial x}$$

Using this fact and substituting our first expression for J, we can derive the diffusion equation, which states that the change in concentration over time is proportional to the change in concentration *gradient* over space.

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2}$$

Thus, at a given point in space, the change in local concentration over time is proportional to the degree to which the concentration gradient changes spatially.

Question 12: *Assume that concentration changes linearly across a system, such that the concentration gradient is constant: $\frac{\partial C}{\partial x} = k$ for some constant k. How does the concentration throughout the system change over time? Explain this macroscopic phenomenon in terms of the motions of individual particles of the system.*

Now, we will put our theoretical understanding to the test and implement two types of simulations of particles diffusing through a cell. We will make the simplifying assumption that cells are rectangles and that if a particle moves beyond a boundary, say over the rightmost boundary, it will appear on the opposite, leftmost, boundary. We will have you complete the code necessary to update the state of the cell from one time step to the next.

In the `StochasticDiffuser`, you will fill in the number of particles at each block (cell) for the next state (`newBlocks`) by iterating through the current cell state and randomly assigning a new direction for each particle to move.

In the `LaplacianDiffuser`, you will update the "concentration" of particles deterministically based on the discrete second derivative of the concentrations (also called the Laplacian).

Before you get started, carefully read over the code in `diffuser.py`. You should understand what `self.blocks` represents and where the initial simulation parameters are set, as well as how you would use the functions `self.nextX()` and `self.nextY()`.

Exercise 9: *Implement* `update()` *in* `StochasticDiffuser` *in* `diffuser.py`.

Exercise 10: *Implement* `update()` *in* `LaplacianDiffuser` *in* `diffuser.py`.

Hint: *To derive the Laplacian, we will approximate the appropriate derivatives using finite differences (i.e.* $\frac{df}{dx} \approx \frac{f(x+h)-f(x)}{h}$*). Specifically, you will approximate the second derivative of concentration with respect to x (*$\frac{d^2C}{dx^2}$*) or y (*$\frac{d^2C}{dy^2}$*). This means that you will also approximate the first derivatives,* $\frac{dC}{dx}$ *and* $\frac{dC}{dy}$*. Assume that the values stored in self.blocks represent concentrations.*

These approximations can be numerically unstable if both the first and second derivatives are calculated using the same interval, i.e. $\frac{df}{dx} = \frac{f(x+h)-f(x)}{h}$ *and* $\frac{d^2f}{dx^2} = \frac{df_x(x+h)-df_x(x)}{h}$ *are using the same interval between x and* $x + h$*. To avoid instability, you will need to balance the derivatives so that they are calculated using different intervals, i.e.* $\frac{df}{dx} = \frac{f(x+h)-f(x)}{h}$ *and* $\frac{d^2f}{dx^2} = \frac{df_x(x)-df_x(x-h)}{h}$*.*

Note: If you need to stop the simulations below, you may do so in terminal via `control-C`.

Question 13: *Run the* `StochasticDiffuser` *(*`python diffuser.py stoc`*) under a variety of initial conditions. You can locate pre-populated conditions at the end of* `diffuser.py`*, and uncomment the condition to run. To change the starting number of particles, edit only the second argument of* `c.setBlock` *(for example, the* `1250` *in* `c.setBlock((length//2,length//2),1250)`*); the diffusion coefficient is set separately, in the* `diffusion =` *line near the top of* `main()`*.*

(a) Include your code for `update` *in* `StochasticDiffuser`*.*
You may remove the comments inside the function if it takes up too much space.

(b) Start with the "Single Particle" simulation. How would you characterize the motion of the particle? ***Hint:*** *The scale bar is indicative of how many particles are in a block.*

(c) Next, run the "Point Mass" simulation. What does the state of the diffuser look like after 100 iterations? After 500?
Hint: *You can edit the plotting script to make it only plot once after N iterations. You may include screenshots if it helps your explanation, but this is not necessary.*

(d) Can you predict where a specific particle will be after many iterations? Can you predict the distribution of particles after many iterations?

(e) Convergence occurs when the diffuser reaches a state of equilibrium. Play around with the diffusion coefficient and starting number of particles. Do either of these parameters affect the rate of convergence? If yes, identify which parameter(s) affect the rate of convergence and describe how the rate of convergence changes.

Question 14: *Run the* `LaplacianDiffuser` *(*`python diffuser.py lap`*) with some of the diffusion conditions in* `diffuser.py`*.*

(a) Include your code for `update` *in* `LaplacianDiffuser`*.*
You may remove the comments inside the function if it takes up too much space.

(b) Play around with the diffusion coefficient and the starting concentration. Do either of these parameters affect the rate of convergence? If yes, identify which parameter(s) affect the rate of convergence and describe how the rate of convergence changes.

(c) Using the "Point Mass" simulation and starting with an initial concentration of $1,250$ *and a diffusion coefficient of* 0.24*, how many iterations does it take for every block (voxel) to have roughly an equal concentration of particles (i.e. to converge)? Note that our criterion for convergence is* $\Delta \text{concentration} < \epsilon \times \text{initial concentration}$ *for all voxels, where* $\epsilon = 0.001$ *(about* $1/1{,}250$*). Provide a brief description of how you determined when the simulation converged.*
Hint: *You can do an approximate calculation using the scale bar to estimate the number of iterations needed for convergence.*
Note*: We will accept a wide range of answers, given a sufficient justification.*

Question 15: *As time goes to $+\infty$, what is the probability that the Laplacian model has a block of concentration 0? What about in the stochastic model — is the probability positive or equal to 0? Justify your answer. (Hint: Do NOT try to explicitly compute this probability. We are asking you to use your intuition based on how these two models work.)*

Question 16: *Based on the answer to the previous question, and any other observations you've made, when do you think it would be appropriate to use the Laplacian Model versus the stochastic model? What are the benefits and drawbacks to each?*

4 Feedback

Again, we'd love some feedback on the assignment! You will receive full credit for this portion for providing any response. Please fill out the assignment feedback form and provide the code word below. We encourage constructive criticism.

Question 17: *What is the feedback form code word?*

5 Challenge Problems

The challenge problems are optional and for extra credit. There are two challenge problems. The two challenge problems are standalone so you may choose to complete either one, both, or neither. The first asks you to build a segmentation pipeline yourself; the second asks you to interpret measurements that someone else's segmentation pipeline produced, at a much larger scale.

5.1 Challenge Problem1: Segmenting Nuclei

The necessary files for this challenge problem are available on the website as a zip file under the Assignments section.

Problem by Laura Miron and Adrian Sanborn. Data by Adrian Sanborn and Cynthia Hao.

In this challenge problem, we have provided microscopy images of fixed human bone cancer cells (U-2 OS) whose nuclei have been stained with the Hoechst fluorescent dye. Your task is to segment the nuclei and compute statistics about their characteristics. The purpose of this problem is to become familiar with the programmatic methods for analyzing image files. We recommend using a conda environment to keep track of the packages you use; see the Assignment 3 Software Guide for details.

Files and Setup

The challenge problem zip file is on the course website under the Assignments section. It contains four `.tif` files, called `nucleus_10x_img*.tif`, and the starter code, `challenge_problem.py`. To view the images, use ImageJ or any other program for scientific images. The starter code provides a function called `load_image` that displays a TIF file and loads it as a numpy array.

Functions provided in the scipy, scikit-image, Pillow, or OpenCV packages may be useful (but are not required). If you choose to use them, install them in your conda environment:

```
pip install scipy
pip install scikit-image
pip install Pillow
pip install opencv-python
```

Note: You may have to use `pip3` instead of `pip`, depending on your conda environment.

Tasks

1. Segment the nuclei in the given images, and display the segmentation as a binary mask. You may use any method you like, including any of these steps:
   (a) **Preprocessing**: smooth, sharpen, or other operations.
   (b) Compute an appropriate threshold value and convert the image to a binary mask. See also **Otsu's method** or **adaptive thresholding** from scikit-image.
   (c) **Fill holes**, e.g. using `scipy.ndimage.binary_fill_holes`.
   (d) Separate objects using a watershed algorithm.
   (e) **Label objects**, e.g. using `scipy.ndimage.label`.
2. Compute the cross-sectional area of each nucleus and plot a histogram of the areas. Each pixel is roughly 0.4 micrometers across.
3. Compute the total Hoechst intensity of each nucleus (this gives a measure of DNA content) and plot a histogram of the intensities.
4. Create a scatter plot of the areas versus the intensities.

Include the following in your submission:

- A binary segmented image of each of the four input files, e.g. nuclei white, background black.
- Your histograms of areas and intensities. Your scatter plot of area versus intensity.
- A complete explanation of your analysis pipeline. Illustrate each step of your pipeline with example images. Why did you design it this way?
- An explanation of other approaches you tried. Why did they perform more poorly? Illustrate with example images.
- A brief discussion interpreting any trends you observe in the nucleus area and intensity data. Are there aspects of the pipeline that could still be improved, and how would this affect these results?

5.2 Challenge Problem 2: Interpreting Segmentation-Based Measurements

In this assignment you segmented a handful of images yourself. Large imaging projects run the same kind of pipeline on hundreds of thousands of cells, and the measurements they produce are only as meaningful as the masks they were computed from. Here you will practice reading such measurements critically and designing one of your own.

The Allen Institute for Cell Science imaged more than 200,000 human induced pluripotent stem cells (a normal, non-cancerous human cell type) in 3D. In each cell line, one type of organelle (e.g. mitochondria, Golgi) is tagged with a fluorescent marker, and each image also has a DNA channel and a cell-membrane channel. Every image was segmented automatically, by filtering and thresholding much like your `threshold` filter, and features such as cell volume and DNA volume were counted from the resulting masks. The Cell Feature Explorer (CFE) website plots these features, one point per cell, and lets you open any cell in a 3D viewer. It runs in your browser, so there is nothing to install. Use the drop-downs beside and under the plot to change the axes, use Color by / Group by to color cells by a category, and click a point and then its thumbnail in the Gallery to open that cell in the 3D Viewer.

Background. The cells were not synchronized, so each was imaged at whatever point it happened to be in its **cell cycle**. A cell spends most of its life in **interphase**, growing larger, with its DNA spread loosely through the nucleus inside a membrane called the **nuclear envelope**. It then divides in **mitosis**, which CFE splits into prophase, early prometaphase, prometaphase/metaphase and anaphase/telophase. During prophase the DNA packs tightly into compact chromosomes, while the nuclear envelope is still intact. At the start of prometaphase the envelope breaks apart, and the chromosomes are then pulled apart as the cell splits in two.

Question 18: *Go to cfe.allencell.org and load the WTC-11 hiPSC Single-Cell Image Dataset (v1). Plot Cell volume on the x-axis and DNA volume on the y-axis, and set Color by and Group by to Interphase & mitotic stages (not "Interphase or mitosis").*

(a) Include a screenshot. The cells fall on two branches. Which stages fall on each branch?

(b) DNA volume drops sharply between prophase and early prometaphase, even though the amount of DNA in the cell has not decreased. Explain why, using the background above. (Hint: what is the DNA segmentation actually outlining before and after the nuclear envelope breaks apart?) What does this tell you about interpreting any feature computed from a segmentation mask?

Next, click Load (top right) and open the Drug Perturbation Pilot Dataset. Each point is now a whole field of view (FOV), plotted by drug and colored by which gene is tagged. Each drug was dissolved in a solvent, and each drug condition has a matching **vehicle** condition in which cells received the solvent alone. We will use **Brefeldin A (BFA)**, which targets the **Golgi**, a compact stack of membranes next to the nucleus that packages proteins for transport. BFA blocks the transport that keeps the Golgi intact, so its membranes merge back into the endoplasmic reticulum, a membrane network that spreads through the cell.

Question 19: *The Golgi is shown in the gene* `ST6GAL1` *cell line.*

(a) Open one BFA FOV and one Vehicle FOV for `ST6GAL1` *in the 3D Viewer. Turn off the membrane channel, switch to the XY view, and include a screenshot of each. Describe the difference in 1–2 sentences. Why is the vehicle, rather than untreated cells, the right comparison?*

(b) Suppose you wanted to measure the BFA effect automatically across every FOV rather than by eye. Propose one feature you would compute from a segmented Golgi channel that would distinguish drug from vehicle. In 2–3 sentences, describe how you would produce that segmentation using the filters from Section 2.1, and one way your pipeline could produce a misleading value.

6 Submission Instructions

Submit your assignment responses to Gradescope under Assignment 3.

As before, type up your solutions in the text editor of your choice. Please include the screenshots of your code for the corresponding questions in the write-up only for the functions we ask. Do not submit any code that has not been explicitly asked for.

Please tag all pages under the corresponding items; without this, we may not be able to give you all the points for a question.