Procedural Cell Shape Model

Seminal work

Advanced Computer Graphics 2020/21, Faculty of Computer and Information Science, University of Ljubljana

Žiga Kleine

***Abstract*—In this report, we will be describing the process of implementing a procedural shape model for modelling cell membrane shapes. We represented the variations in cell membrane structure with a PDM or a point distribution model. We take in several training examples of points sampled from the surface of the cell. The points need to be sampled in an orderly manner, because in the next step we calculate mean shape of all training examples and a covariance matrix representing the covariance in the training shapes. We then perform the principal component analysis on the covariance matrix, extracting its eigenvalues and eigenvectors from it. With the extracted mean and eigenvalues and eigenvectors of the covariance matrix, we can now generate new shapes, based on the statistical model learned from the training example shapes.**

I. Introduction

Procedural modelling of cell shape can be a useful tool assisting the complicated process of medical image segmentation. For this seminal work, we followed a commonly used technique of procedural modelling using a point distribution model, described in [1].

Before we can use a point distribution model to model the shape of the cell membrane, we need to prepare and process the needed data. A point distribution model needs a set of training examples of n cell shapes represented by m points sampled from the cell membrane. The points need to be sampled in an order, so that we can then compare the points between different cells. For ordering the sampled cells, we used two approaches, randomly sampling m points and then ordering them in a manner, similar to how cyllindrical coordinates work, and an approach where we created a sphere of evenly distributed points around the cell, and then fired rays towards the cell from the sphere. We will describe these two methods in more detail in the next chapter. We used cells with already annotated membranes from the Allen institute for cell science web page [2].

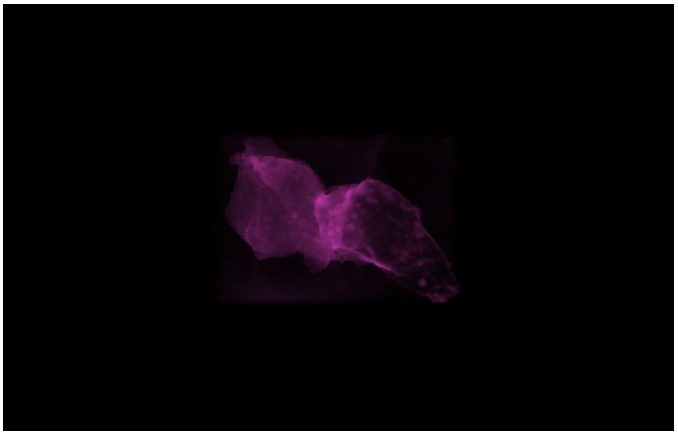

Figure 1. Example of an observed cell membrane shape that we used for training.

II. Methodology

A. Gathering cell membrane training shapes

For the purpose of this experiment, we downloaded 50 different single cell voxel images, which include channels with just the annotated membrane voxels. These annotated voxel images are the basis for our experiment. For the Point distribution model, we needed to gather m landmark points from each of the n training shapes, that were sampled in an orderly manner, so the shapes can then be compared. For that we tried out two different algorithms, that we will describe here.

The first method that we tried out is the method where we first sampled m random points from the annotated membrane, and then sorted them first by their angle in the cylindrical coordinate system, and then by their z-axis. When sorting along the z-axis, we rounded the z-coordinates in bins with the width of 10 units. That way, we got the points that were first sorted along the z-axis into bins of 10 units, and then inside each bin, the points were sorted along the azimuth angle. That way, we got the points that had a similar order inside each sampled training cell shape.

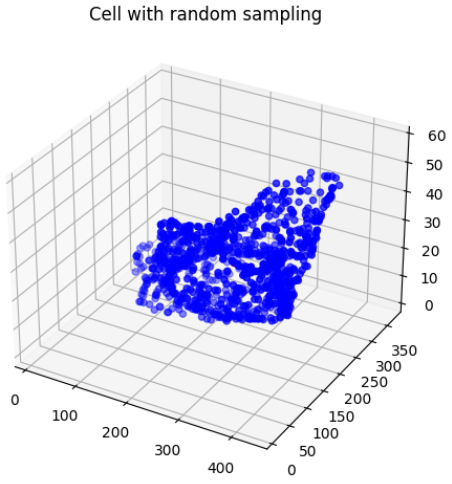

Figure 2. Examples of points sampled from a cell with random sampling sorted.

The second method we decided to try out is to first generate a sphere of m evenly distributed points around around the annotated cell membrane that we wanted to sample. For that we used a sphere drawing algorithm called the fibonacci sphere. The centre of the sphere is the mean point of all of the annotated cell membrane voxels. Then, we fired a ray from each point on the sphere towards the centre of the sphere, using a voxel path tracing algorithm, to detect where the ray intersects the cell membrane. We then used each of these intersections to represent a sampled point on the cell membrane. For generating the sphere of evenly distributed points, we used a fibonacci sphere algorithm, and for path tracing we used a Bresenham's

line drawing algorithm for voxel path tracing [3]. That way we generated another way to get a set of sampled cells in a meaningful order.

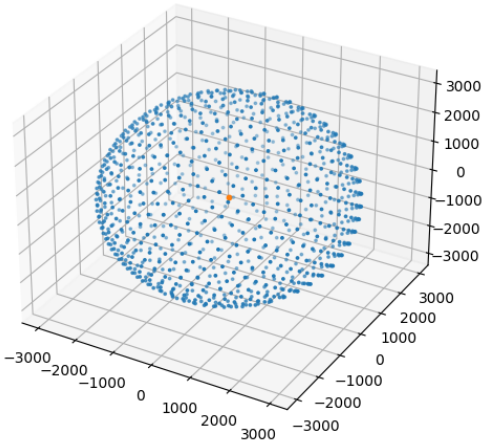

Figure 3. Fibonacci sphere made from 1000 points.

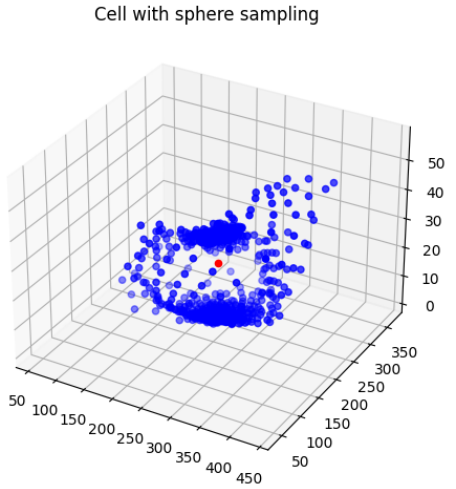

Figure 4. Examples of points sampled from a cell with sphere sampling.

B. Implementing the Point distribution model

The point distribution model is a technique that statistically represents shape variability. The shape variability is learned form m landmark points in n training shapes. We then perform PCA on the training shapes, and then we can generate new cell shapes with a linear combination of eigenvectors that we get.

The first step in computing a PDM is to calculate the mean shape from our training shapes. This is done by using the following equation:

$$\overline{x} = \frac{1}{n}\Sigma_{i=1}^{n} x_i$$

Then, we calculate the covariance matrix of the points in the shape, using the following equation:

$$C = \frac{1}{n-1}\Sigma_{i=1}^{n}(x_i - \overline{x}) * (x_i - \overline{x})^T$$

The next step is then to calculate the eigenvalues and eigenvectors of the calculated covariance matrix. We get $3m$ eigenvalues and $3m$ corresponding eigenvectors. The eigenvector that corresponds to the largest eigenvalue contains the largest portion of shape variations, while others contain decreasingly less and less.

Now that we calculated the mean, the eigenvalues and their corresponding eigenvectors, we can generate new shapes by using the formula below:

$$x_i' = \overline{x} + w_i * q_i$$

where

$$|w_i| \leq \pm 3 * \sqrt{\lambda_i}$$

In the equations above, q_i represents the i-th biggest eigenvalue and λ_i represents the corresponding eigenvector. To ensure that the w_i value is between $-3 * \sqrt{\lambda_i}$, and $+3 * \sqrt{\lambda_i}$, we multiply the mentioned equation with a randomly sampled uniform value between -1 and 1.

III. Results

We ran our algorithm with different parameter values, and then visualized the results using a 3d scatter plot in the matplotlib python library.

We ran the random sampling sorted algorithm on 5 training shapes ($n = 5$), and we sampled 800 point models ($m = 800$), which in turn returned 800 point shapes. In the image 5, we can see the new shapes generated with first, second and third largest eigenvalues and their corresponding eigenvectors to describe shape variability.

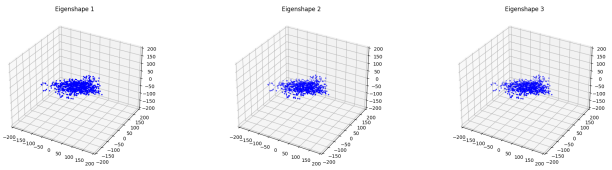

Figure 5. Examples of newly generated cell shapes from the first three eigenshapes using random sorted sampling.

We ran the spherical sampling algorithm on 10 training shapes ($n = 10$), and we sampled 800 point models, which in turn returned 800 point shapes ($m = 800$). In the image 6, we can see the new shapes generated the same way as before, but with the spherical method of sampling of points, that we described earlier. All of the visualisations were generated using a random seed from the random number generator. We could get different reproductible shapes by using different seeds for the random generator.

IV. Discussion

During the experimentation process, we found out that the random sorted sampling works well if we keep the number of training shapes lower, while the sphere sampling works good even with larger training shape datasets. We found out that when using the random sorted sampling with a dataset of training shapes bigger than for example 5, the new generated

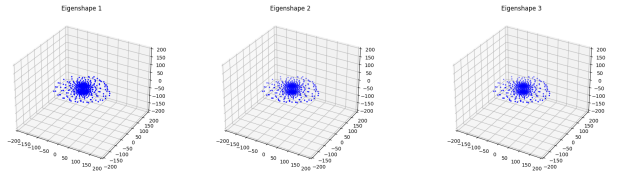

Figure 6. Examples of newly generated cell shapes from the first three eigenshapes using sphere sampling.

cell shapes start to become smaller and more congested around the mean value, while the sphere sampling seems to retain the new shapes better even with more training examples.

We can also observe that the sphere sampling doesn't sample points from the membrane uniformly, the points are more dense where the membrane is closer to the centre of the sphere (the mean of the membrane points) because the rays get closer together, the closer they are to the centre point. That means also that the resulting new generated shapes have a similar structure with congestions of points closer to the membrane. That taken into account, the congestions don't really influence the output cell shape that much, which means that the method works for sampling ordered points from a structure such as the cell membrane. But another drawback that can be observed is that the rays often don't reach all of the complexities in the surface of the cells, because another cell membrane part could have blocked the ray, losing the information about the surface in this point. This was probably the case for a lot of sampled points, because we can observe that the newly generated shapes with the spherical sampling are pretty much always convex, and don't seem to have the surface complexities that can be observed in the cells generated with the random sampling model.

Our approach to the procedural modelling of cell membranes could be improved by using modern neural network based approaches for cell shape modelling, since neural networks are usually better at learning the complexities of a structure like the cell membrane.

The code for the project can be found at https://github.com/zigakleine/ProceduralCellShapeModel.

References

[1] T. Vrtovec, D. Tomaževič, B. Likar, L. Travnik, and F. Pernuš, "Automated construction of 3d statistical shape models," *Image Analysis & Stereology*, vol. 23, no. 2, pp. 111–120, 2004.

[2] A. institute for cell science, "Download cell data (images, genomics, features) - allen cell explorer," https://www.allencell.org/data-downloading.html, 2021, [Online; accessed 20-May-2021].

[3] P. Koopman, "Bresenham line-drawing algorithm," *Forth Dimensions*, vol. 8, no. 6, pp. 12–16, 1987.