Quantitative Image Analysis and Modelling of *Phallusia* Oocytes and *Caenorhabditis elegans* Egg

A thesis submitted towards the partial fullfilment of
BS-MS dual degree programme
from AUGUST 2022 to MAY 2023

by

SIDDHARTH DEOGAM

under the guidance of

DR. CHAITANYA ATHALE

Professor, Biology Department

Indian Institute of Science Education and Research
Pune

Certificate

This is to certify that this dissertation entitled “Quantitative Image Analysis and Modelling of *Phallusia* Oocytes and *Caenorhabditis elegans* Egg” submitted towards the partial fulfillment of the BS-MS degree at the Indian Institute of Science Education and Research, Pune represents original research carried out by “Siddharth Deogam” at “IISER Pune”, under the supervision of “Dr. Chaitanya A. Athale” during academic year August 2022 to May 2023.

Chaitanya Athale

Supervisor:
Dr. Chaitanya A. Athale
Professor
IISER Pune

Date:
22/05/2023

Declaration

I, hereby declare that the matter embodied in the report titled "Quantitative Image Analysis and Modelling of *Phallusia* Oocytes and *Caenorhabditis elegans* Egg" is the results of the investigations carried out by me at the "IISER Pune" from the period 01-08-2022 to 01-05-2023 under the supervision of Dr. Chaitanya A. Athale and the same has not been submitted elsewhere for any other degree.

Siddharth Deogam

Siddharth Deogam
20181006
BS-MS
IISER Pune

Date: 22/05/2023

Acknowledgements

I would like to thank Dr. Chaitanya A. Athale hosting me in his lab and providing me with valuable guidance and resources. I am also grateful to Aman for his patient teaching of simulations, which greatly contributed to my understanding of the subject matter. I would like to acknowledge Dhruv for his mentorship and instruction in image analysis and deep learning. I am grateful to IISER Pune for providing me with an intellectually stimulating and supportive environment in which to pursue my studies. I would like to thank my parents for their unwavering love, support, and encouragement. Finally, I would like to express my appreciation to the KVPY for the fellowship.

Contributions

Contributor name	Contributor role
Chaitanya Athale	Conceptualization Ideas
Siddharth Deogam, Aman Soni, Dhruv Khatri	Methodology
Siddharth Deogam, Aman Soni, Dhruv Khatri, Neha Khetan	Software
Siddharth Deogam	Validation
-	Formal analysis
Siddharth Deogam	Investigation
Chaitanya A. Athale	Resources
Dhruv Khatri	Data Curation
Siddharth Deogam, Chaitanya A. Athale	Writing - original draft preparation
Siddharth Deogam, Chaitanya A. Athale	Writing - review and editing
Siddharth Deogam, Dhruv Khatri	Visualization
Chaitanya A. Athale	Supervision
Chaitanya A. Athale	Project administration
Chaitanya A. Athale	Funding acquisition
Janet Chenevert (IMEV, Villefranche), Marie Delattre (ENS Lyon)	Experimental data

This contributor syntax is based on the Journal of Cell Science CRediT Taxonomy[1]

[1] https://journals.biologists.com/jcs/pages/author-contributions

Contents

Listings

Abstract

Circle packing problem refers to most number of circles which could be fit inside a boundary. Asters or radial microtubules are closely associated with the geometry of a cell. Spatial patterns of microtubule motors are responsible for multiple cellular structures like mitotic spindles. Computational modelling and experimental image analysis combined have shown that these systems can self-organize and form structures even in minimal systems consisting of asters and motors . It has been seen that that microtubules combined with molecular motors can self-organize to form patterns in space that match the rules of the circle packing problem. The sensitivity of the self-organized patterns that emerge from it depends on multiple factors such as concentration, kinetic rates and geometry. The results were obtained in 2D (experimental and simulation) despite having a 3D *Phallusia* oocyte as a sample organism. This report attempts to expand their work from 2D to 3D, by creating a new tessellation and simulation model.

Caenorhabditis elegans is a transparent nematode which is a popular model organism for studying embryonic development. Intraspecific variation in egg shell size and shape are the result of mutations. This thesis deals with the creation of a eggshell detection pipeline based on convolution neural network, U-net. A semi-automatic segmentation algorithm creates ground truth data, which is then used to train U-net, a convolution neural network designed for image segmentation. The pipeline will be used to study the changes in morphometry of eggshell in *C. elegans* and among related species, leading to how the morphometry changed across evolutionary time scales. The segmentation pipeline could also be used to segment *Phallusia* oocytes in the future, owing the the samples approximately round shapes

Chapter 1

Introduction

Tessellation is the covering of a geometric space with shapes such that there are no overlaps or gaps. Voronoi tessellation is a partitioning of a space into voronoi chambers, each chamber or territory contains a single seed point (Weisstein, 2023b). Any random point in a chamber is closer to the seed point than to any other seed points in the space. The vertices of a Voronoi diagram are derived by joining the centers of circumcircles inscribing a Delaunay triangle. Delaunay triangulation is a triangulation where no seed point is inside a circumcircle inscribed by three seed points (Weisstein, 2023a). Delaunay triangulation and Voronoi tessellation of a list of graphs make a dual graph. Voronoi diagrams are used to quantify the space occupied by asters, as described in a previous study (Khetan *et al.*, 2021).

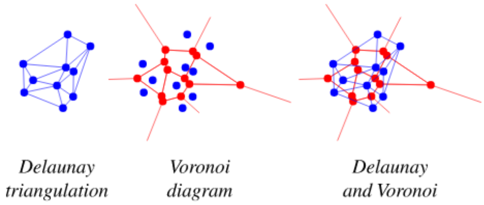


Figure 1.1: **Creation of Voronoi tessellation from Delaunay triangulation in a 2D space** The seed points are marked in blue. Delaunay triangulation is done on them. The circumcenter for a circle joining three triangulated points is shown in red. The circumcenters are joined to create Voronoi cells around seed points. Figure taken from (Weisstein, 2023a)

Asters are radial structures of microtubules with a centre nuclated by a γ-tubulin ring complex. They serve as scaffolds for molecular motors to walk on, leading to numerous interesting properties like self-center localisation and asymmetric force generation which have huge implications in cell physiology.

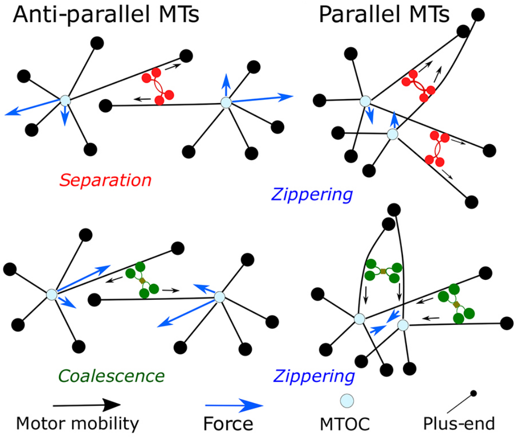


Figure 1.2: **Possible cases of motor-microtubule interaction.** Kinesin5 and dynein are depicted in red and green respectively. Kinesin-5 walks towards the plus end, separating antiparallel and zippering parallel MTs. Dynein walks towards the minus end, coalescing antiparallel and zippering parallel MTs. Figure taken from (Khetan and Athale, 2020)

It was observed in *Phallusia* oocyte that when treated with BI-D1870, a potent ATP-competitive inhibitor of all ribosomal S6 kinase (RSK) isoforms, had asters show tessellation patterns as a property of collective kinesin-5 movement (Khetan and Athale, 2020). Kinesin-5 is a tetrameric molecular motor which can walk on two microtubules at once. These patterns depended on MT length, stability and aster numbers. Packing fraction (ϕ) is defined as the ratio of total area occupied by asters to available area. When $\phi \sim 1.6$, the aster centers lined up in near hexagonal arrangement, which is very reminiscent if circular packing problem. Extending this into 3D would resembleaspecial case of spherical packing problem limited only to the periphery of a larger sphere with intercalation of MTs.

Caenorhabditis elegans is a nematode which is widely used as a model organism to study developmental biology. *C. elegans* embryos are studied using differential interference contrast microscopy, DIC, due to the transparent nature of the embryos (Sulston *et al.*, 1983). DIC is a microscopy technique used to view unstained transparent samples. When light travels through different structures, a path difference is produced. DIC used this path difference to enhance the contrast between such structures. *C. elegans* embryos are sensitive to the toxicity created by molecular markers and high intensity light, both of which do not feature in DIC, enabling the embryos to grow unperturbed.

The first cell division can be observed in *C. elegans* embryo. The male and female pronuclei line up along the posterior and anterior end respectively, forming the anterior-posterior (AP) axis. They move to the center of the embryo and mitosis begins. Spindles are formed which pull the centrosomes along the AP axis. A furrow appears and divides the cytoplasm into two uneven daughter cells. Hence, the first mitotic division is asymmetric.

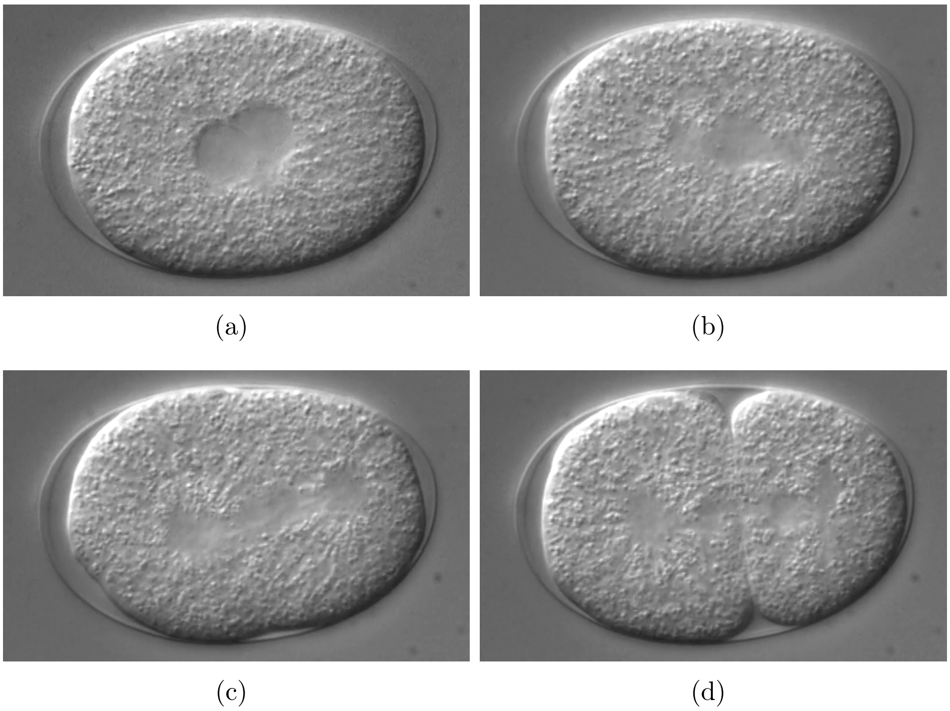

(a) (b) (c) (d)

Figure 1.3: **Progression of mitosis.** (a)DIC image of *C. elegans* embryo taken from (Valfort *et al.*, 2018). The two pronuclei and associated centrosomes move towards the center of the embryo. (b)The aster centers are towards the two poles and the centrosomes are lined up at the equitorial plane, showing metaphase (c)The centrosomes are being pulled apart towards the poles, demonstrating anaphase (d)The cytoplasmic furrow divides the cytoplasm between the daughter cells, this process is called cytokinesis.

Segmentation is the partitioning of image into its constituting objects. Segmentation is done by detecting sharp changes in image pixel value gradient. The segmentation algorithm traces the eggshell during the first mitosis division. Inspiration was taken from ASSET (Blanchoud *et al.*, 2010) to segment out the eggshell using cartesian to polar image transformations. The cartesian to polar transformations are conserved but the segmentation line is drawn using different methods. ASSET draws lines using a shortest path algorithm over scoring functions, such that the path with the strongest and smoothest lines get marked. Whereas our algorithm fits a polynomial line to points with a high edge value while taking the image gradient values of these points as weights. The algorithm generates ground truth data i.e. a binary image which marks the egg and the surroundings. The masks along with their respective images are used to train the convolution neural network.

The convolution neural network is based on U-net (Ronneberger *et al.*, 2015). U-Net has a U-shaped architecture that enables it to capture both high-level and low-level features of an image, while also incorporating spatial information. This means that it can effectively use limited training data to achieve high segmentation accuracy.U-Net uses skip connections between the encoder and decoder blocks, which helps to mitigate the vanishing gradient problem and allows for faster and more stable training. Due to its efficient architecture, U-Net can perform segmentation tasks quickly and with relatively low computational resources, making it ideal for real-time applications. U-net is commonly used in medical image segmentation, object detection, and image classification.

In the following sections the problem of tessellation of asters from 3D datasets, simulating their localization to match experiments and segmenting the egg shell boundary will be addressed.

Chapter 2

Materials and Methods

2.1 Tessellation

2.1.1 Simulation setup

Simulations are performed on Cytosim, a C++ based simulation engine based on Langevin dynamics (Nedelec and Foethke, 2007). Fibers are infinitely thin elastic and non-extensible rods, which behaves like an Euler-Bernoulli beam. The attachment of motors onto microtubules resembles a Hookean spring. Motor-motor interactions are not present. Cytosim version 2.1.2 was used in the simulations (https://gitlab.com/f-nedelec/cytosim). They were run on an Intel i5-7200U CPU @ 2.50GHz × 4 processor which took 10-80 hours to complete.

2.2 Image segmentation

2.2.1 Images of *C. elegans* Embryogenesis

DIC images depicting the time frame of the first asymmetric embryonic division of *C. elegans* embryos were taken from published data described by Dr. Marie Delattre (Valfort *et al.*, 2018). The division lasted about 500s. Each image was taken at an interval of 0.5 s and each image stack consisted of approximately 1000 images which provided a good temporal resolution for tracking the phenomenon. The size of each image was $500 - 600 \times 300 - 400$ pixels.

2.2.2 Eggshell Segmentation

MATLAB ver. R2022b (Mathworks Inc., Natick, MA, USA) was used for image segmentation. Cartesian to polar image transformation and vice versa were carried out using "Cartesian to polar image transform" MATLAB script written by (William Warriner).

2.2.3 Convolution neural network

A prebuilt convolution neural network (CNN) provided with the MATLAB DeepLearning toolbox (DLToolBox). Additionally, the "Computer Vision" toolbox was also used to help train the network. The training was done on an Intel Xeon E5 2620 CPU in ∼2 hours.

Chapter 3

Results

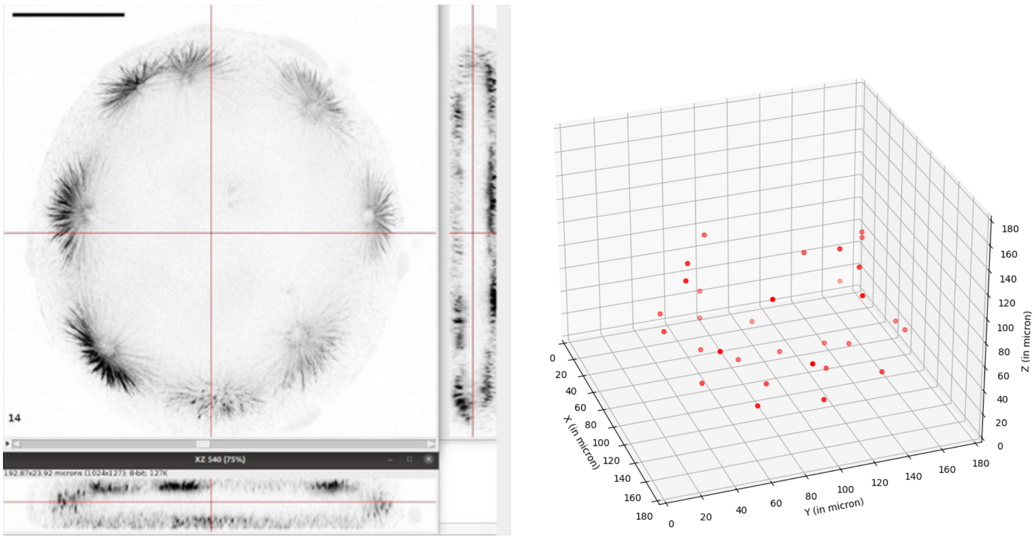
The results are divided into two sections - modelling aster tessellation in *Phallusia* oocyte in 3D and eggshell segmentation of *C. elegans* embryos. The modelling of *Phallusia* oocyte deals with expanding the work of (Khetan and Athale, 2020) from 2D to 3D. The segmentation problem began as an exercise to segment *Phallusia* oocytes. Because the segmentation problem of *C. elegans* eggshell was similar to that of the oocytes, owing to their round shapes and the former is of interest to my lab, it was tackled first.

3.1 Tessellation

Phallusia oocytes stained with anti-α-tubulin antibodies (data obtained from Dr. Janet Chenevert) was treated with drug BI-D1870. A confocal z-stack of the oocyte was taken. The sample showed multiple aster formation in the peripheral cytoplasm, making football like patterns on the surface. It is hypothesized that no asters are seen in the interior because of the presence of a large central vacuole.
The coordinates of the aster centers was obtained using imageJ and then extrapolated onto a 3D space using as the known scale. The sample was compressed along the z-axis because of a cover slip and hence accurate position of aster centers cannot to determined. The distance between each consecutive image was taken to be 5 μm. It can be noticed that the aster centers are present around an empty central space.

3.1.1 Simulation Model

Simulations are performed on Cytosim, a C++ based simulation engine based on Langevin dynamics. Fibers are infinitely thin elastic and non-extensible

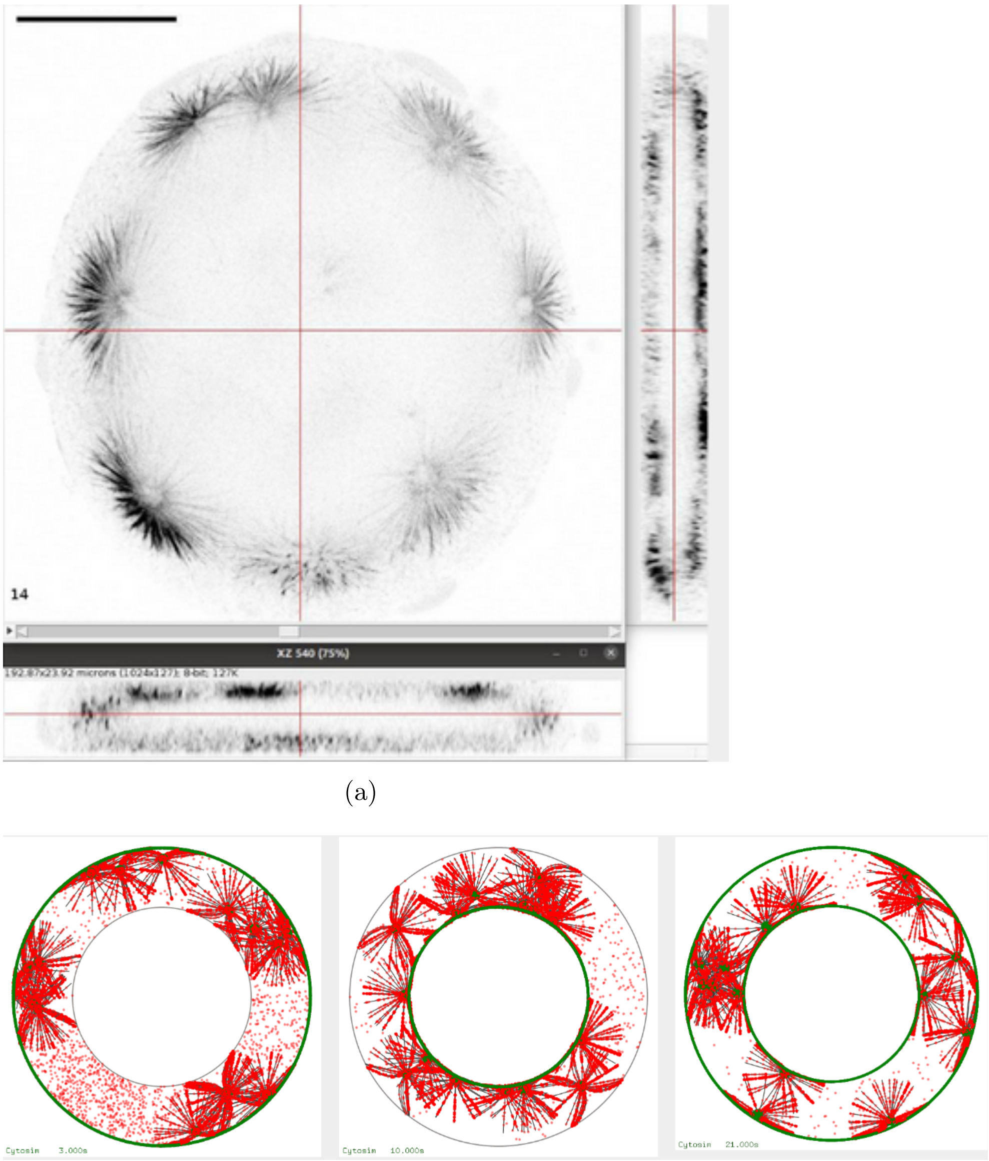


(a)

(b)

Figure 3.1: (a) The 3D confocal z-stack of a representative *Phallusia* oocyte treated with 10 μM BiD and stained for α-tubulin (the grayscale image is inverted). Figure taken from (Khetan and Athale, 2020) (b) The aster centers are projected as red dots on a 3D space. Plotting was done in Python3.

rods, which behaves like an Euler-Bernoulli beam. The attachment of motors onto microtubules resembles a Hookean spring. Motor-motor interactions are not present. The asters of *Phallusia* localise in the peripheral cytoplasm due of the presence of a semi-permeable vacuole at the center. Microtubules and molecular motors being too big to pass through the membrane are excluded from the interior. Cytosim does not provide for the creation of such concentric spaces. Modifications were made to the engine where multiple functions were changed to get a sphere with an inner spherical excluded volume. The simulations done in (Khetan and Athale, 2020) comprised of asters and motors placed within a circular boundary. This did not resemble the structure of the oocyte, hence making these changes were necessary. The middle simulation of figure 3.2 matches that of the confocal data. The aster centers look close to the vacuole with the radial microtubules projecting outward and not inwards. It is hypothesized that the outer surface of the vacuole has dynein attached to it which holds the aster centers close which mimics the *Phallusia* oocyte treated with BiD.



(a)

(b)

Figure 3.2: **Comparison of vacuole in sample and simulation** (a) Confocal image of phallusia oocyte showing the central vacuole. Figure taken from (Khetan and Athale, 2020)(b) Comparision of cytosim simulations. Three different localisation of dynein (in green) is shown. From left to right, dynein is present on the cortex, outer surface of vacuole and both respectively. The aster microtubule patterns from the confocal image matches with the simulation in the middle.

Parameters	Dynein on cortex	Dynein on vacuole	Dynein on cortex and vacuole
Number of Kinesin-5 motors	7070	7070	7070
Number of dynein motors	9000	5000	14000

Table 3.1: Parameters used in simulation in fig 3.2.
Outer radius = 15 μ, inner radius = 10μ, number of asters = 15

3.1.2 Modifications to Cytosim

The ‘space_sphere.cc’ class defines the properties of a spherical or circular space. The inner radius is taken to be 3. An additional condition,

$$(pos^2 <= 9)$$

was added to ‘inside’ function which would return ’false’ when the position vector ‘pos’ had a magnitude squared of less than 9. The ‘project’ function takes the position vector of a particle when it goes outside the boundary and projects it on the boundary, or ‘pos’, the boundary projection of ’pos’ is the closest point on the boundary to ‘pos’. An additional condition was added to ‘project’ function such that when ‘pos’ points inside the inner radius

$$pos^2 <= 9$$

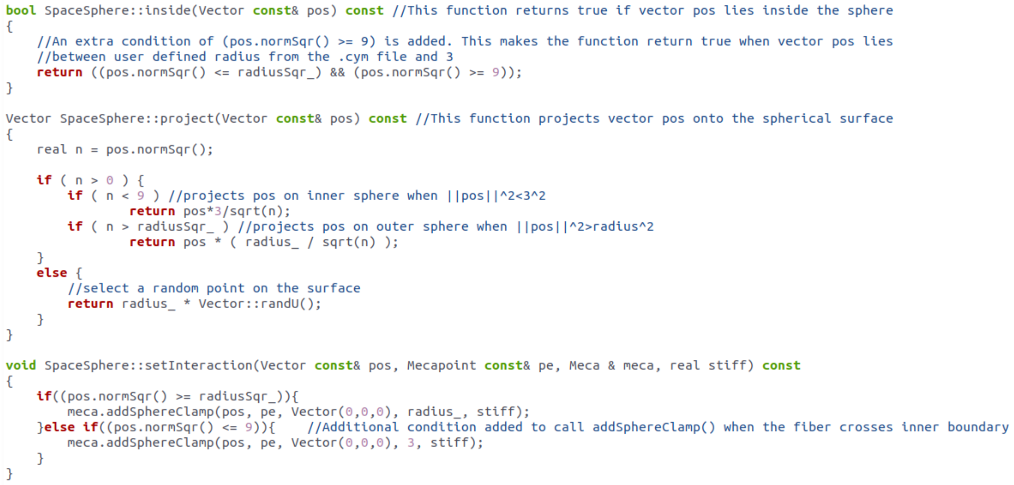
, it gets projected on the inner boundary.
The microtubules get flicked away from the inner vacuole. To prevent this, changes were made when addSphereClamp function. This function adds an additional force on a particle

$$force = weight * (C - P) * \frac{1 - rad}{|PC|}$$

where P is the coordinates of the particle and C is the coordinates of center of circle. The force acts towards the center when P is outside the outer circle and it acts away from the center when P is inside the vacuole.

3.1.3 3D Tessellation

Going from a 2D to a 3D space, the number of motors and asters had to be substantially increased to get tessellaton patterns.



```
bool SpaceSphere::inside(Vector const& pos) const //This function returns true if vector pos lies inside the sphere
{
    //An extra condition of (pos.normSqr() >= 9) is added. This makes the function return true when vector pos lies
    //between user defined radius from the .cym file and 3
    return ((pos.normSqr() <= radiusSqr_) && (pos.normSqr() >= 9));
}

Vector SpaceSphere::project(Vector const& pos) const //This function projects vector pos onto the spherical surface
{
    real n = pos.normSqr();

    if ( n > 0 ) {
        if ( n < 9 ) //projects pos on inner sphere when ||pos||^2<3^2
                return pos*3/sqrt(n);
        if ( n > radiusSqr_ ) //projects pos on outer sphere when ||pos||^2>radius^2
                return pos * ( radius_ / sqrt(n) );
    }
    else {
        //select a random point on the surface
        return radius_ * Vector::randU();
    }
}

void SpaceSphere::setInteraction(Vector const& pos, Mecapoint const& pe, Meca & meca, real stiff) const
{
    if((pos.normSqr() >= radiusSqr_)){
        meca.addSphereClamp(pos, pe, Vector(0,0,0), radius_, stiff);
    }else if((pos.normSqr() <= 9)){    //Additional condition added to call addSphereClamp() when the fiber crosses inner boundary
        meca.addSphereClamp(pos, pe, Vector(0,0,0), 3, stiff);
    }
}
```

Figure 3.3: **Changes made to the source code in space_sphere.cc of cytosim.**
This class defines the properties of a spherical or circular space. The inner radius is taken to be 3. An additional condition was added to 'inside' function which would return 'false' when the position vector 'pos' had a magnitude squared of less than 9.
The 'project' function takes the position vector of a particle when it goes outside the boundary and projects it on the boundary, or 'pos', the boundary projection of 'pos' is the closest point on the boundary to 'pos'. An additional condition was added to 'project' function such that when 'pos' points inside the inner radius, it gets projected on the inner boundary.
The 'setInteraction' function gets called when two objects have to interact. 'addSphereClamp' gets called whenever a fiber goes outside a boundary, a force acts on a protruding segment towards the center of the sphere. The magnitude of this elastic force is a function of boundary stiffness, very similar to a Hookean spring. An additional condition was added such that it would get called whenever 'pos' would point inside the inner boundary, resulting in the force being applied away from the center.

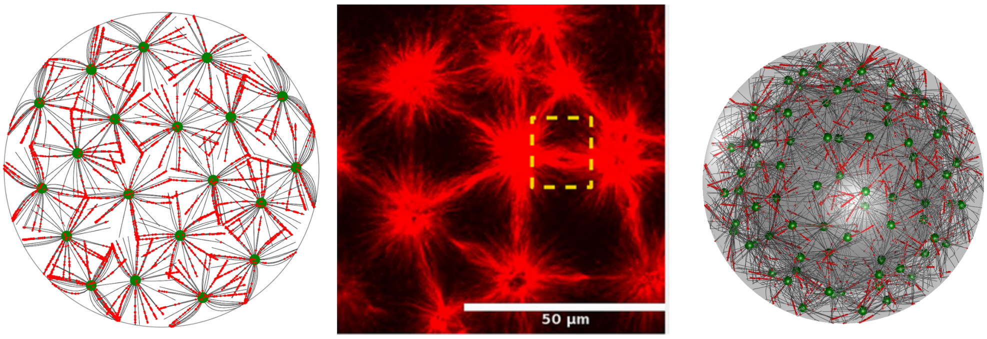
The parameters used in 2D and 3D simulations (Figure 4(a) & 4(c) respectively), to get microtubule bundling (Table 3.2). It acts as a compressing force in packing aster centers and close as possible.

Parameters	2D	3D

Radius (in μm)	15	outer = 15,
		inner = 10.775

Number of Kinesin-5 motors	7070	280000
Number of Asters	20	80
Simulated time (in seconds)	600	300
Time taken to simulate (in hours)	19	150

Table 3.2: Parameters used in simulation.
Scaling of the parameters from 2D to 3D. The parameter values scale by a large amount, resulting in a great increase in computation time.



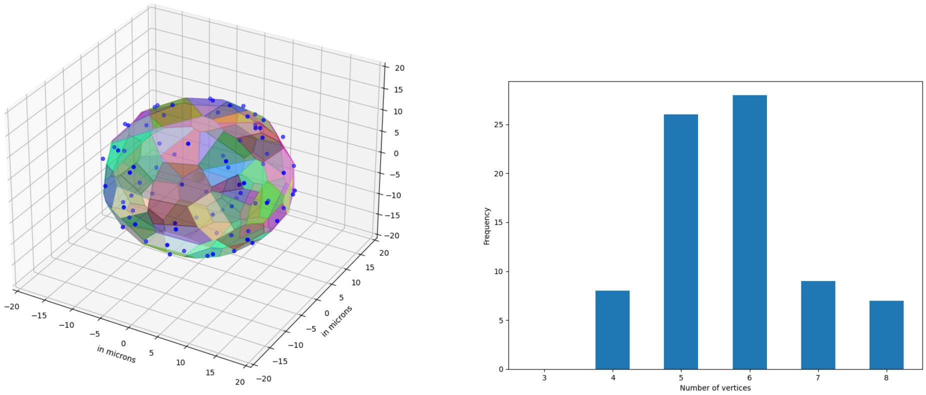
(a) (b) (c)

Figure 3.4:
(a)2D aster tessellation done in cytosim. Few microtubules look bundled together because of bridging done by kinesin-5 motors. (b) Oocyte shows microtubule bundling when treated with BI-D1870. (c) 3D simulation of aster tessellation in cytosim. The asters have almost stabilised into polygonal patterns. Kinesin-5 motors are seen in red which are bridging microtubules.

3.1.4 Finding Optimum Packing Fraction

In (Khetan and Athale, 2020), the packing fraction(ϕ) was defined as

$$\phi = \frac{\text{Area occupied by asters}}{\text{Total a available area}}$$



(a)

(b)

Figure 3.5: Quantification of 3D voronoi model
(a)3D voronoi diagram of fig 3.4 (b). The aster centers are marked in blue. Each voronoi compartment is marked by a different colour. (b)Histogram of the number of vertices in a voronoi polygon

or,

$$\phi = \frac{N \times L^2}{R^2}$$

where N is the number of asters, L is the radius of asters and R is the radius of the cell. In a 3D model,

$$\text{surface area ocupied by an aster} = \pi \times L^2$$

$$\text{total area occupied by asters} = N \times \pi \times r^2$$

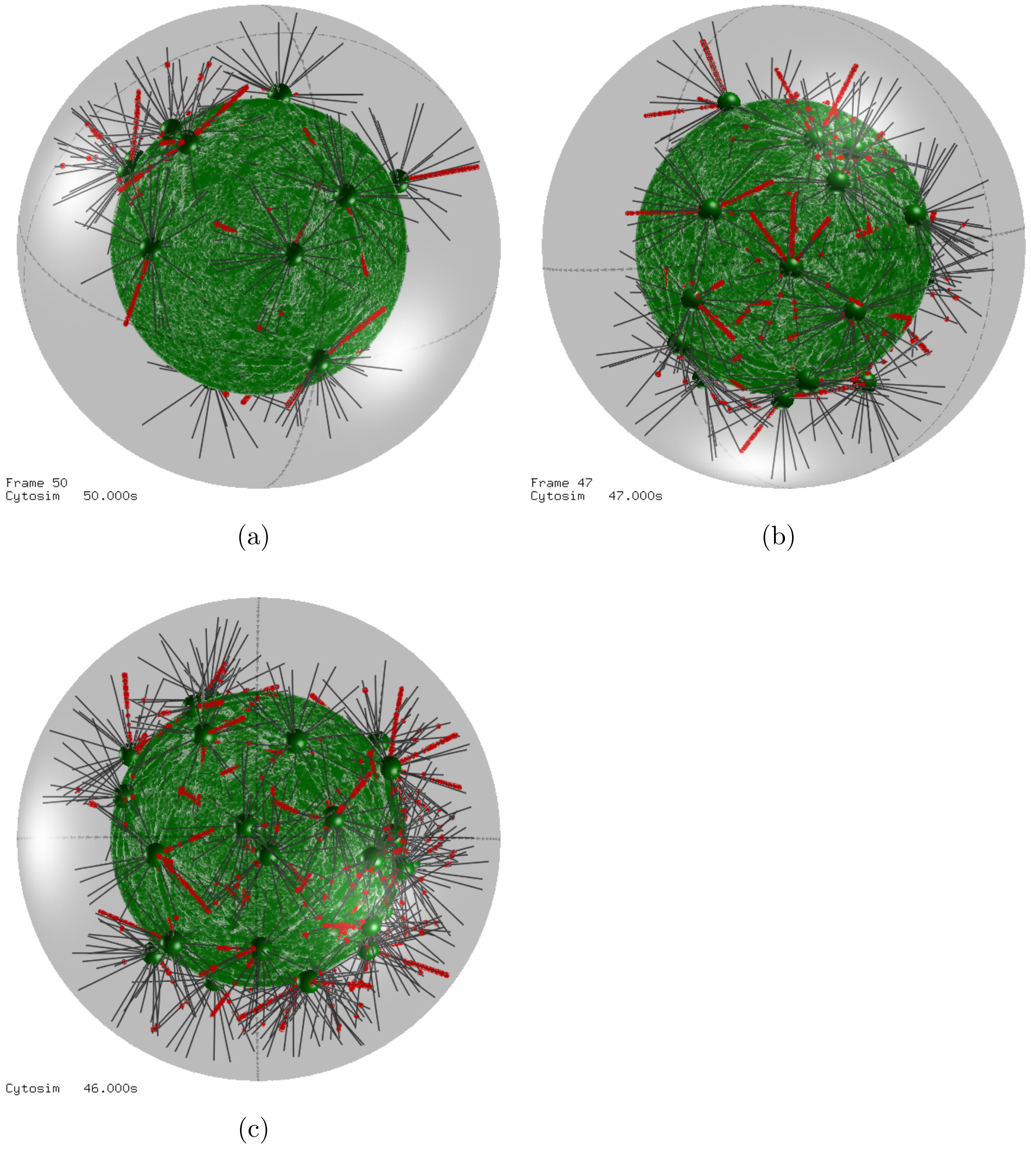
$$\text{surface area available} = 4 \times \pi \times r^2$$

Hence,

$$\phi = \frac{N \times L^2}{4 \times r^2}$$

as the asters are sticking to the vacuole because of dynein, the asters are covering the surface area of a sphere instead of a flat circle. ‘r’ is the radius of the vacuole.

The minimmum packing fraction required for tessellations to be seen is more than 1.7 which is not consistent with the optimum packing fraction of 1.6 from 2D simulations (Khetan and Athale, 2020). When $\phi = 1.2$, the aster centers are too far away to make tessellating patterns around the vacuole. When $\phi = 1.7$, kinesin-5 is seen binding with microtubules of neighbouring asters, making them push away. The packing fraction might need to be



(a)

(b)

(c)

Figure 3.6: Simulations for varying packing fraction. The simulation did not run long enough for the asters to stabilise. Bridging kinesin-5 is shown in red and dynein is shown in green
Outer and inner radius = 10, 6 μm; No. of kinesin-5 = 190000 (a)$\phi = 1.2$, N=14, t=251 s. The asters are too far to form bundles with adjacent asters (b)$\phi = 1.7$, N=20, t=162 s. The asters are just close enough to form bundles. Microtubule bridging is seen (c)$\phi = 2.55$, N=30, t=243 s. The asters are very tightly packed.

Parameters	Values
Radius (in μm)	outer $=$ 10, inner $=$ 6
Number of Kinesin-5 motors	190000
Number of Dynein motors	350000
Radius of aster (in μm)	3.5

Table 3.3: Parameters used in simulation.

increased a bit for the asters to cover the entire vacuole. When Interestingly, in 2D simulations, tessellations could be seen only in the absence of dynein which acts as an antagonist to kinesin-5. In 3D simulation, tessellation can be seen even in the presence of dynein. Albeit, the dynein is not freely diffusing, but stuck to the outer membrane of the vacuole. When $\phi = 2.55$, the asters are packed together densely. This implies that the optimum packing fraction is between 1.7 and 2.55.

3.2 Eggshell Segmentation

3.2.1 Image Acquisition

DIC images depicting the time frame of the first asymmetric embryonic division of *C. elegans* embryos were procured. The division lasted about 500s. Each image was taken at an interval of 0.5 s and each image stack consisted of approximately 1000 images which provided a good temporal resolution for tracking the phenomenon. The size of each image was $500 - 600 \times 300 - 400$ pixels. The images used are from the dataset from (Khatri *et al.*, 2022)

3.2.2 Segmentation Algorithm

The eggshell is a smooth protective structure which surrounds the embryo. The evenness of the eggshell makes The first step involves resizing the rectangular image into a square image to get even sized blank constructs and a more linear eggshell in the polar image.
The second step involves the conversion of the cartesian to a polar image. The MATLAB script by William Warriner was used to carry out this trans-

formation and back. The size of the polar image is $(\pi \times d)$ $\times d/2$ where d is the length of the diagonal of the cartesian image. Since the cartesian is being opened and the resolution of the image is being changed, empty pixels are present between non empty pixels. They are filled using linear interpolation to get a smoother image. Semi-circular blank spaces are present in the polar image which corresponds to the regions outside the cartesian image which also get translated. To remove them, for each pixel column of the image, the pixel value immediately above the null area was copied to fill that specific column.

In the third step, a 'canny' filter is applied on the polar image to get a binary image. The edges are represented in white. For each column, the coordinates of the first positive pixel is taken from the bottom. The image gradients along x ad y are calculated from the polar image. The root mean square of the two gradients are calculated which serves as weights for the coordinates of the edges. A polynomial is fit over the edge points with the root mean square of x and y gradients as weights. This line marks the boundary between the eggshell and the surrounding. The polynomial line becomes closed as the polar image is transformed back to the cartesian image.

Image masks are required to train semantic neural networks. In the polar image from the polynomial fit, all the pixels above the line are designated '1' and the pixels below are designated '0'. The image is transformed to back to cartesian. The eggshell and the regions within are marked while the exterior is not.

3.3 Neural Network

Convolution neural networks (CNN) are widely used as classifiers in computer vision. They take an input image, detect objects by identifying features, and outputs regions demarcating said objects. U-net was used in MATLAB using the 'deep learning' module. The U-net created has 58 layers and 61×2 connections. The classifier segments images into 'embryo' and 'background'. It inputs the 'image datastore' which is the collection of *C. elegans* embryo images, and the 'pixel label library', which is the collection of image masks which aid in semantic classification. U-net outputs a 'combined dataset' which is a combination of the image and new segmented mask. U-net is called so because it's architecture resembles a 'U'. The input image goes into the descending arm. The descending arm consists of convoluting encoders and ReLu (rectified linear units) which reduce the spatial resolution of the feature maps while increasing their depth. After passing through convoluting and ReLu layers twice, the input is passed through a pooling layer which

Training option	Stochastic gradient descent
Learning rate schedule	piece-wise
Learning rate drop factor	0.3
Learning rate drop period	10
Momentum	0.9
Initial learning rate	0.001
L2 regularization	0.005
Maximum epochs	30
Minimum batch size	8
Shuffle	every epoch
Verbose frequency	2
Validation patience	4

Table 3.4: Training parameters for U-net

reduces the spatial dimension of the feature maps by half. This reduces the computational cost and helps the model to focus on the most important features. The ascending layer consists of decoders which take the representation of features upscales them to fit the regions in the image which match the features. The depth concatenation layer is present after two successive convolution and ReLu pairs and a up-convolution and up-Relu layer. The depth concatenation takes inputs with similar dimensions and stacks them. The higher the stack, the stronger the feature. The ascending arm restores the spatial resolution of the image while incorporating the learned features. Skip connections are added between layers which concatenate the feature maps from the encoder with the feature maps from the decoder to produce more accurate segmentation maps.

The training accuracy is 86.36%. The loss is 0.3857 on the validation set. The accuracy can be further increased and the loss decreased by increasing the training data set. Current training set consisted of 48 images which is less than optimum.

The testing set consisted of 4 images. They were segmented by the trained U-net and then compared the resultant semantic segmentation mask with the ground truth mask.

Figure 3.9 (c) is a test image and 3.9 (d) is the segmented image. The 'egg' is in blue and the 'background' is in cyan. Most of the embryo is correctly classified as the 'egg' but a large portion of the periphery is also classified as the embryo. The U-net detects the embryo with high accuracy but the background with low accuracy. This means that the false positive is

Global Accuracy	0.79297
Mean Accuracy	0.75412
Mean IoU	0.62697
Weighted IoU	0.65065
Mean BF Score	0.31941

Table 3.5: Results after testing U-net

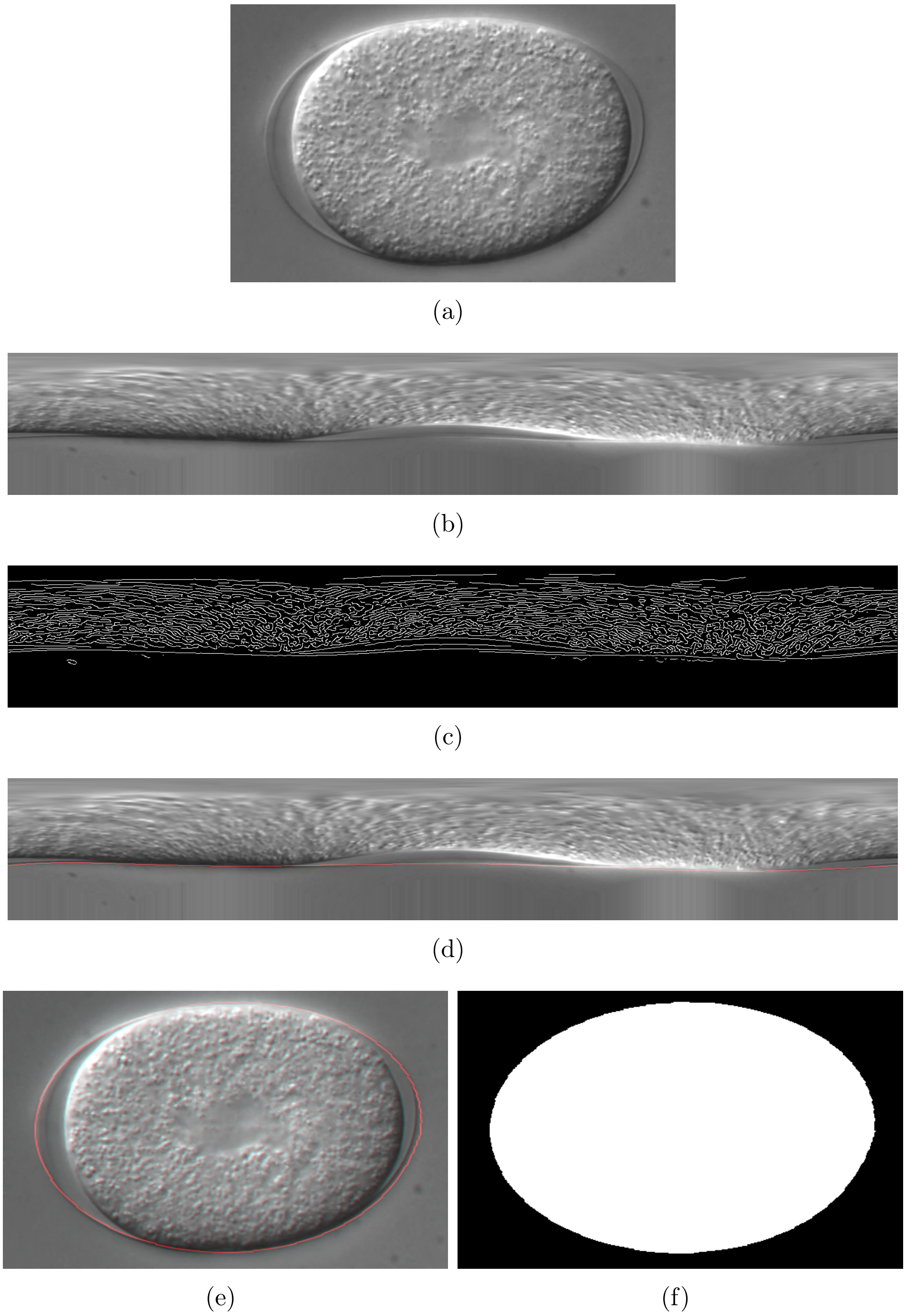
	Accuracy	IoU	Mean BF Score
Egg	0.92831	0.73276	0.19838
Background	0.57993	0.52113	0.44044

Table 3.6: Class metric

	Egg	Background
Egg	0.92831	0.071686
Background	0.42007	0.57993

Table 3.7: Normalized confusion matrix

high. This implies the specificity is low. This can be fixed by increasing the training set and the threshold for embryo detection.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 3.7: **Progression of segmentation.**(a)DIC image of *C. elegans* embryo. The eggshell and the cortex is distinctly visible. Data taken from (Valfort *et al.*, 2018) (b) The cartesian image in (a) in transformed into polar image. The black constructs have been removed by filling the pixel value immediately above the null area was copied to fill that specific column. (c) Binary image is produced by applying a canny filter over (b) (d) Fitting a polynomial line of degree 6, coloured orange. (e) Cartesian transformation of polar image. The polynomial fits along the outer boundary of the eggshell (f)Binary mask of the segmented image. The white part corresponds to the eggshell and the area enclosed within. Black denotes the surroundings.

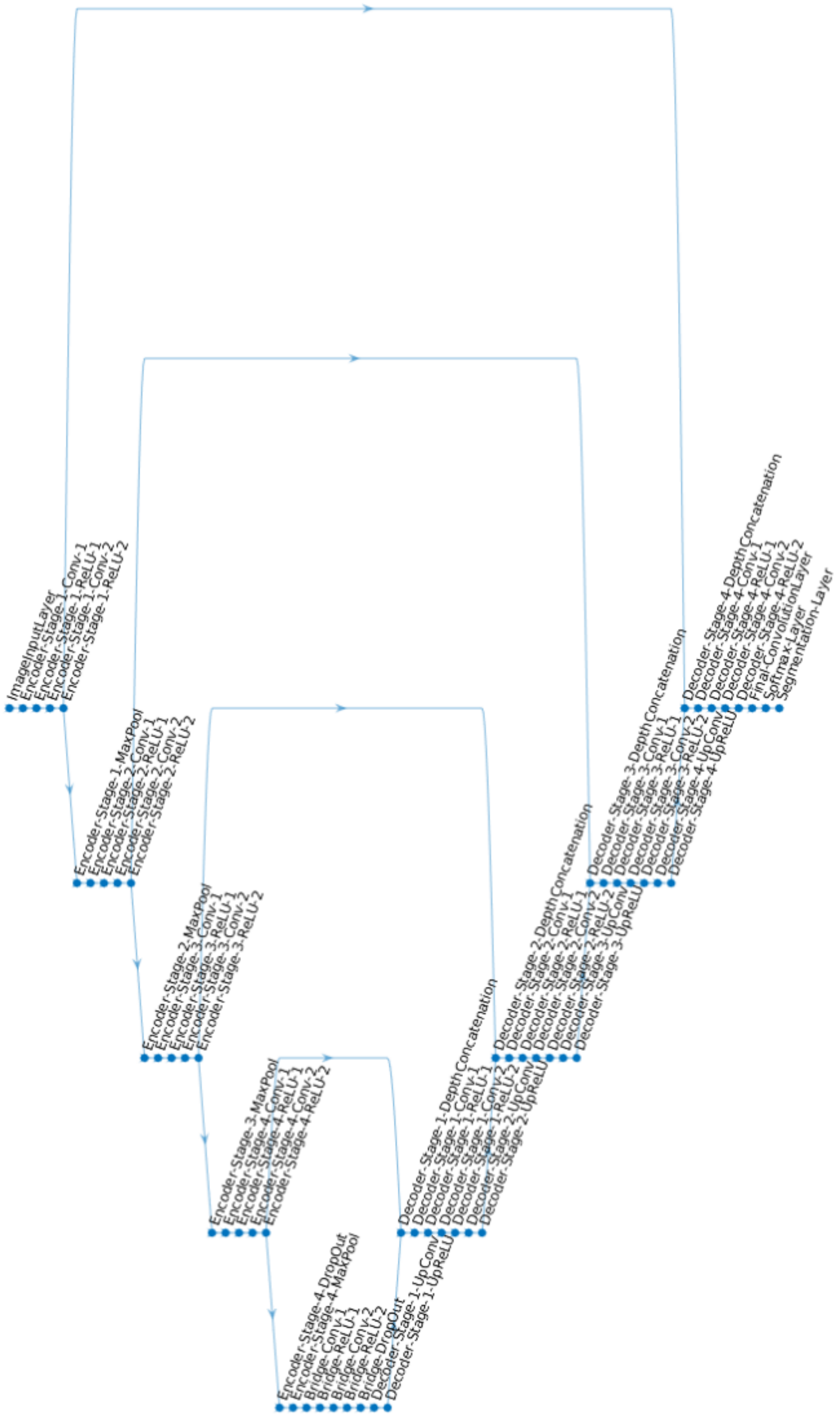
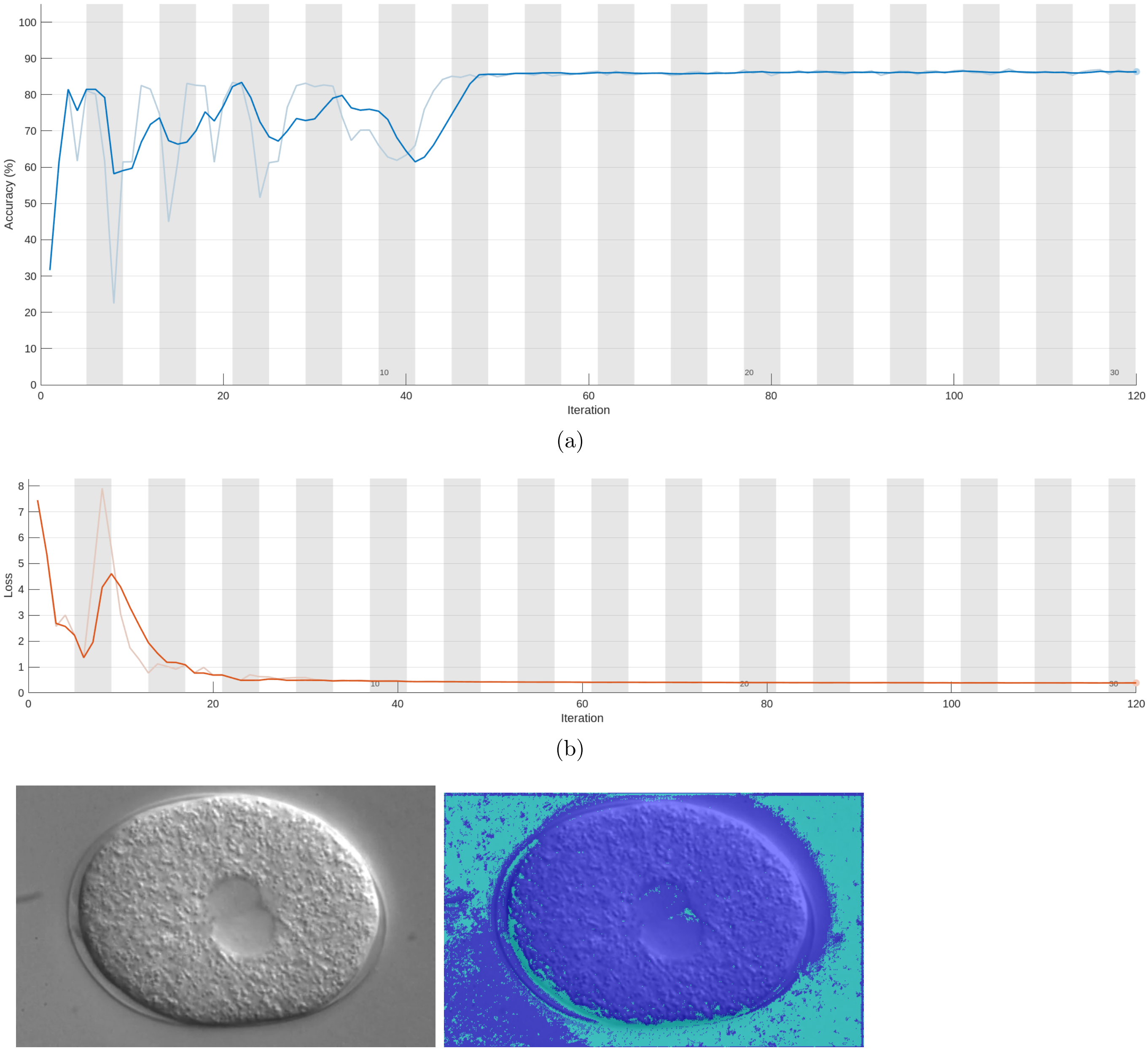


Figure 3.8: The U-net network used. There are a total of 58 layers and 61×2 connections. Figure is made in Matlab R2023a.



(a)

(b)

(c)

(d)

Figure 3.9: Training Result(a) Accuracy plot (b) Loss function plot (c) Test image (d) Segmented image. Detected embryo is in blue, background is in cyan

Chapter 4

Discussion

The changes made to cytosim reciprocate the 2D simulations done previously. The presence of a central vacuole is hypothesized which explains the aster patterns from the confocal data, the aster centers coming very close to the vacuole and the radial microtubules spreading outward and not inward. The parameters are varied to get the optimum packing fraction ϕ. A tessellation model was written in python which would create voronoi cells around aster positions which would help quantify the kind of patterns being formed. The aster tessellations depend on microtubule length, number of asters and the area of volume being covered. The packing fraction was redefined to fit the 3D simulations.

Asters in cells are driven by forces are driven by microtubules and motors. The optimum packing of asters would reveal a novel pattern forming system which could be significant in pattern formation of organisms.

The segmentation problem began as an exercise to segment *Phallusia* oocytes. Because the segmentation problem of *C. elegans* eggshell was similar to that of the oocytes, owing to their round shapes and the former is of interest to my lab, it was tackled first. The segmentation of *C. elegans* embryo was achieved by using an algorithm based on cartesian to polar and vice versa image transformations which was inspired by ASSET's algorithm (Blanchoud *et al.*, 2010). The semi-automatic algorithm generates image masks for classifying the image into the egg and the background. The segmented images are then used to train the convolution neural network.

The accuracy is not high enough owing to the small training data. It can be corrected by increasing the training data. A well trained U-net should be able to segment the eggshell of *C. elegans* and related species. The network would help to study the intra and inter specific morphometry of eggshell. Similarly, in the future, the U-net can be trained to segment the cortex of the embryo and the spindles. The network would then quantify these features within and between species. This would help to further expand the work of Valford (Valfort *et al.*, 2018) in which factors which influence spindle oscillations in the first mitotic division have been studied and used to determine the evolutionary flow of these features.

The segmentation algorithm can be modified to segment the cortex which lies underneath the eggshell, using a similar method as ASSET (Blanchoud *et al.*, 2010). The U-net can also be trained to parallely segment both the eggshell and the cortex. Since cortex undergoes noticable changes during the first mitotic division, their morphology could hold a sway in spindle oscillations. Using DICOT (Chaphalkar *et al.*, 2021), the cytoplasmic granules can be tracked and it could help train the U-net to detect them. Hence, the evolutionary flow of eggshell, cortex and cytoplasmic flow can be mapped by studying their covariation in *C. elegans* and related species.

The algorithm and neural network could be used to segment the cell membrane *Phallusia* oocyte. Additional code for the segmentation of asters can be written. These two along with the code for quantification of 3D tessellation, would model the *Phallusia* oocyte.

Bibliography

Blanchoud S, Budirahardja Y, Naef F, nczy P (2010). ASSET: a robust algorithm for the automated segmentation and standardization of early Caenorhabditis elegans embryos. Dev Dyn 239(12), 3285–3296.

Chaphalkar AR, Jawale YK, Khatri D, Athale CA (2021). Quantifying intracellular particle flows by dic object tracking. Biophysical Journal 120(3), 393–401.

Khatri D, Brugière T, Athale CA, Delattre M (2022). Evolutionary divergence of anaphase spindle mechanics in nematode embryos constrained by antagonistic pulling and viscous forces. Molecular Biology of the Cell 33(6), ar61.

Khetan N, Athale CA (2020). Aster swarming by symmetry breaking of cortical dynein transport and coupling kinesins. Soft Matter 16(37), 8554–8564.

Khetan N, Pruliere G, Hebras C, Chenevert J, Athale CA (2021). Self-organized optimal packing of kinesin-5-driven microtubule asters scales with cell size. J Cell Sci 134(10).

Nedelec F, Foethke D (2007). Collective langevin dynamics of flexible cytoskeletal fibers. New Journal of Physics 9(11), 427.

Ronneberger O, Fischer P, Brox T (2015). U-net: Convolutional networks for biomedical image segmentation. In N Navab, J Hornegger, WM Wells, AF Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, 234–241. Springer International Publishing, Cham.

Sulston JE, Schierenberg E, White JG, Thomson JN (1983). The embryonic cell lineage of the nematode Caenorhabditis elegans. Dev Biol 100(1), 64–119.

Valfort AC, Launay C, mon M, Delattre M (2018). Evolution of mitotic spindle behavior during the first asymmetric embryonic division of nematodes. PLoS Biol 16(1), e2005099.

Weisstein E (2023a). "delaunay triangulation." from mathworld–a wolfram web resource. Mathematica.

Weisstein E (2023b). "voronoi diagram." from mathworld–a wolfram web resource. Mathematica.

Appendix 1

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: sid
"""
#%%
%matplotlib widget
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.spatial import SphericalVoronoi, geometric_slerp
from mpl_toolkits.mplot3d import proj3d
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

f=open("aster_points.txt","r")
lines=f.readlines()
x_temp=[]
y_temp=[]
z_temp=[]
for i in lines:
    x_temp.append(i.split('\t')[0])
    y_temp.append(i.split('\t')[1])
    z_temp.append(i.split('\t')[2])
f.close()

x=[]
y=[]
z=[]
for item in x_temp:
    x.append(float(item))
for item in y_temp:
    y.append(float(item))
for item in z_temp:
    z.append(float(item))

sz = len(x);
#%%
```

```
radius = 15
center = np.array([0, 0, 0])
points = np.zeros((sz,3))
for i in range(len(x)):
    points[i,0]=x[i]*radius/math.sqrt(x[i]**2+y[i]**2+z[i]**2)
    points[i,1]=y[i]*radius/math.sqrt(x[i]**2+y[i]**2+z[i]**2)
    points[i,2]=z[i]*radius/math.sqrt(x[i]**2+y[i]**2+z[i]**2)
#points = np.numpy
sv = SphericalVoronoi(points, radius, center)
sv.sort_vertices_of_regions()
#%%
data=[]
for i in range(len(sv.regions)):
    count=0
    if -1 in sv.regions[i]:
        continue
    for j in range(len(sv.regions[i])):
        count+=1
    data.append(count)
#data.remove(0)

fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(data, bins = [2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5], rwidth=0.5)
plt.xlabel("Number of vertices")
plt.ylabel("Number of occurences")
plt.show()
#%%
# sort vertices (optional, helpful for plotting)
sv.sort_vertices_of_regions()
t_vals = np.linspace(0, 1, 2000)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# plot the unit sphere for reference (optional)
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
x = np.outer(np.cos(u), np.sin(v))
y = np.outer(np.sin(u), np.sin(v))
z = np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_surface(x, y, z, color='y', alpha=0.1)
# plot generator points
ax.scatter(points[:, 0], points[:, 1], points[:, 2], c='b')
# plot Voronoi vertices
#ax.scatter(sv.vertices[:, 0], sv.vertices[:, 1], sv.vertices[:, 2],c='g')
#ax = fig.add_subplot(111, projection='3d')
```

```
for i, region in enumerate(sv.regions):
    if not region:
        continue
    vertices = sv.vertices[region]
    poly = Poly3DCollection([vertices], alpha=0.5)
    poly.set_color([np.random.rand(3)])
    ax.add_collection3d(poly)
ax.set_xlim(-20, 20)
ax.set_ylim(-20, 20)
ax.set_zlim(-20, 20)
fig.set_size_inches(4, 4)
plt.show()


#%%
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i, region in enumerate(sv.regions):
    if not region:
        continue
    vertices = sv.vertices[region]
    poly = Poly3DCollection([vertices], alpha=0.5)
    poly.set_color([np.random.rand(3)])
    ax.add_collection3d(poly)
ax.set_xlim(-20, 20)
ax.set_ylim(-20, 20)
ax.set_zlim(-20, 20)
fig.set_size_inches(4, 4)
plt.show()
```

Listing 4.1: Code used for plotting Voronoi tessellation in 3D

The python code reads the location of aster centers from the .txt file. The centers are then projected onto the surface of a user defined sphere. These points act as the seed points around which the Voronoi regions are built. The construction of Voronoi cells are done using SphericalVoronoi class from scipy package. The code also creates the histogram of the number of vertices of these voronoi cells.

Appendix 2

Modified cytosim

Here is the code for the modified version of cytosim
Changes have been made to the space_sphere.cc class of cytosim to simulate the effect of an empty central vacuole. The .cym files are also provided.

U-net

Here is the code used to create the segmentation U-net network
The MATLAB script for the creating of U-net for segmentation of *C. elegans* eggshell has been provided. The network parameters for training have also been provided.