University of California
Santa Barbara

Segmentation, Tracking, and Shape Modeling for 3D Time-lapse Microscopy Images

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Jiaxiang Jiang

Committee in charge:

Professor B.S. Manjunath, Chair
Professor Kenneth Rose
Professor Nina Miolane
Professor Michael Liebling

December 2022

The Dissertation of Jiaxiang Jiang is approved.

Professor Kenneth Rose

Professor Nina Miolane

Professor Michael Liebling

Professor B.S. Manjunath, Committee Chair

November, 2022

Segmentation, Tracking, and Shape Modeling for 3D Time-lapse Microscopy Images

by

Jiaxiang Jiang

Dedicated to my family and friends.

Acknowledgements

I want to take a moment to thank those who helped me along the way on my dissertation journey.

First and most importantly, I would like to express my sincere gratitude to Professor B.S. Manjunath, who has advised my research with consistent support, patience, and inspiration. His guidance not only helped me find my research direction in computer vision and image analysis field but also taught me how to do research.

Second, I am grateful to Professor Nina Miolane, Professor Kenneth Rose, and Professor Michael Liebling, for being on my dissertation committee, discussing research topics, and offering insightful suggestions. Many thanks to Professor Daniel B. Szymanski and Professor William Smith who provided an excellent environment for interdisciplinary research. I am also thankful to NSF award # 1715544, NSF SSI award # 1664172, and NIH funding # 5R01NS103774-04 that financially supported my Ph.D. study.

It has been a pleasure to collaborate with Amil Khan, Shailja, Michael Goebel, Po-Yu Kao, Samuel A. Belteton, Cezar Borba, Chandrakanth Gudavalli, and Rahul Vishwakarma. Technical discussions with Amil Khan, Michael Goebel, Shailja, Po-Yu Kao, Aditya Jonnalagadda, Fei Xu, Thuyen Ngo, and Utkarsh Gaur are really helpful for coming up with research ideas. Regular meetings with Samuel A. Belteton and Cezar Borba not only help me trim my research topic but also provide me more insights about the biology impact of my dissertation. I am also grateful for Kerrianne Ryan for her time and amazing EM image annotations. I enjoyed the time I spent interacting with all Vision Research Lab members: Austin McEver, Satish Kumar, Devendra Jangid, Raphael Ruschel Dos Santos, Abu Saleh Mohmmed Iftekhar Niloy, Angela Zhang, Ekta Prashnani, Ivan Arevalo, Connor Levenson, Aditya Ramakrishnan, Anmol Kapoor, Griffin Danninger, Carlos Torres, Dmitry Fedorov, Christian Lang, Archith John Bency, Kristian Kvilekval,

Xudong Lin, Zency Young, Tao Deng, Oytun Ulutan, and Lingyun Song.

Finally, special thanks to my family, especially my parents who love me and support me unconditionally. Without them, I could not have gone this far. I would like to thank all my friends who support me on this journey.

Curriculum Vitæ

Jiaxiang Jiang

Education

November 2022	**Doctor of Philosophy** Electrical and Computer Engineering University of California, Santa Barbara, USA.
May 2017	**Master of Science** Electrical and Computer Engineering The Ohio State University, USA.
June 2016	**Bachelor of Science** Electrical Engineering University of Electronic Science and Technology of China, China.

Honors & Awards

2022	ECE Department Dissertation Fellowship, UCSB
2020	2nd Place on Cell Tracking Challenge, ISBI
2019	Conference Travel Grant, UCSB Graduate Student Association
2019	ICIP Conference Travel Grant, IEEE Signal Processing Society
2015	Second of People's Scholarship, UESTC
2014	Province Third Place Award, Mathematics Contest in Modelling
2014	First of People's Scholarship, UESTC
2013	Second of People's Scholarship, UESTC

Publications

Deep Learning Enabled Time-Lapse 3D Cell Analysis, **Jiaxiang Jiang**, Amil Khan, S. Shailja, Samuel A. Belteton, Michael Goebel, Daniel B. Szymanski, B.S. Manjunath, Submitted to Nature Scientific Reports, 2022

Geographic Atrophy Lesion Segmentation Using Multimodal Deep Learning Networks, Theodore Spaide, **Jiaxiang Jiang**, Jasmine Patil, Neha Anegondi, Verena Steffen, Michael Kawczynski, Liz Newton, Christina Rabe, Simon Gao, Aaron Lee, Frank G. Holz, Srini-Vas Sadda, Steffen Schmitz-Valckenberg, and Daniela Ferrara, Submitted to IOVS, 2022

Effects of activation method and temperature to III-nitride micro-light-emitting diodes with tunnel junction contacts grown by metalorganic chemical vapor deposition, Matthew S. Wong, Nathan C. Palmquist, **Jiaxiang Jiang**, Philip Chan, Changmin Lee, Panpan Li, Ji Hun Kang, Yong Hyun Baek, Chae Hon Kim, Daniel A. Cohen1, Tal Margalith, James S. Speck, Shuji Nakamura, and Steven P. DenBaars, Applied Physics Letters, November 2021

Analysis of numerical feature extraction from automated geographic atrophy segmentation, Theodore Spaide, Jasmine Patil, **Jiaxiang Jiang**, Neha Anegondi, Michael Kawczynski, Verena Steffen, Simon S Gao, Investigative Ophthalmology & Visual Science, June 2021

Semi Supervised Segmentation and Graph-based Tracking of 3D Nuclei in Time-lapse Microscopy, S. Shaija*, **Jiaxiang Jiang*** (*denotes equal contribution), B.S.Manjunath, 2021 International Symposium on Biomedical Imaging (ISBI), Nice, France

Improving Patch-Based Convolutional Neural Networks for MRI Brain Tumor Segmentation by Leveraging Location Information, Po-Yu Kao, Shailja, **Jiaxiang Jiang**, Angela Zhang, Amil Khan, Jefferson W. Chen, B.S. Manjunath, Frontier Computational Neuroscience, 2020

Accurate 3D Cell Segmentation Using Deep Feature and CRF Refinement, **Jiaxiang Jiang**, Po-Yu Kao, Samuel A. Belteton, Daniel B. Szymanski, B.S. Manjunath, 2019 International Conference on Image Processing (ICIP), Taipei, Taiwan

Experience

07/2017-Present	Graduate Student Researcher, Vision Research Lab, UCSB
06/2021-09/2021	Graduate Technical Intern, Intel Labs
06/2020-09/2020	Research Intern, Genentech Clinical Imaging Group

Programming Languages

Python, Matlab, C++, C

Service

2021-Present	IEEE Transaction on Computational Imaging Reviewer
2021	ICIP 2021 Reviewer

Abstract

Segmentation, Tracking, and Shape Modeling for 3D Time-lapse Microscopy Images

by

Jiaxiang Jiang

Time lapse 3D images are important resources for biology research because they not only provide 3D structural information but also provide temporal information. Much of the analysis of these data are manual and subjective, and does not scale well with the large amount of imaging data that is routinely collected these days. This is especially true for 3D and time-lapse imagery. Fundamental problems such as detecting cells, sub-cellular features, tracking of such structures in 3D over time, and modeling 3D shapes in robust manner, remain. The problems addressed in this dissertation are motivated by two bio-imaging problems:

1. Understanding the plant pavement cell growth pattern. The pavement cell growth controls the leaf expansion patterns and rates which are the key determinants of the overall photosynthetic rates of the canopy. Time lapse image stacks from 3D confocal imagery are a good resource to study the pavement cell growth process. To better understand this process, machine learning methods are developed to detect and track cellular and sub-cellular features. We developed a deep learning enabled time-lapse 3D analysis pipeline that includes novel boundary tagged 3D segmentation method, and a graph based sub-cellular feature extraction and tracking method. Detailed quantitative evaluation results demonstrate the robustness and state-of-the art performance of the proposed methods.

2. Neuron morphology analysis. Cell morphology especially neuron morphology plays an important role in biology because neuron functions are closely related to neuron

morphology. In this dissertation, we propose a robust computational 3D skeleton model to analyze neuron morphology. It is the first deep learning method to compute a 3D neuron skeleton model directly from discrete 3D surface points for neuron classification. The main innovation is in formulating the learning problem associated with computing the medial axis transform that represents the 3D skeleton. We apply our method on two Datasets, Ciona neuron dataset and C.elegan neuron dataset for classification. It results in an accurate and robust skeleton representation, and achieves state-of-the-art performance in classifying neuron types.

The implementation of the methods developed above and the associated data are made available on GitHub and also as a software service through the UCSB BisQue platform.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cogito, ergo sum

René Descartes, 1637

Life science research cannot be conducted without looking into cells. Microscopy imaging is the primary observation modality for understanding the structure, function and behavior of cells. Cell function and activities are closely related to cell structure [14], and change of 3D cellular structure is a key determinant of the cell function. The modern time-lapse imaging has become a powerful and continuously improving tool for studying the cellular processes and cell-cell interactions with the applications ranging from fundamental aspects of molecular and cell biology to medical practice [15, 16].

Confocal mircrocopy, Fluorescent imaging, transmission electron microscopy (TEM), and scanning electron microscopy (SEM) are some examples of commonly used imaging modalities for cell studies. Those different modalities are suitable for different applications. For example, fluorescent imaging can show single molecules within cells, confocal microscopy is preferred for high-quality images and 3D reconstruction of cells, and TEM is used to view high resolution of thin samples.

These advanced imaging methods generate vast amount of image data. Manual analysis of large amount of image data not only laborious but also subjective. In some cases, such as annotating sub-cellular features in 3D, it is not possible for manual analysis because of lack of appropriate tools to work with such 3D image stacks, and the problem gets more challenging for time-lapse imagery. Further, any such manual analysis, even if possible, is not reproducible. Hence there is an urgent need to develop robust, scalable and reproducible automated methods for analyzing such imaging data.

Our motivation for the methods presented in this dissertation comes from two biology problems: 3D pavement cell growth and neuron morphology analysis. As we explain below, these problems differ from the common computer vision problems that deal with natural images, and offer new opportunities for research at the intersection of bioimaging and image analysis.

3D pavement cell growth The epidermal cell, also known as pavement cell, undergoes a dramatic transformation from a slightly irregular polyhedral cell to a highly convoluted and multi-lobed morphology. The interdigitated growth mode is widespread in the plant kingdom [17], and the process by which lobing occurs can reveal how force patterns in the tissue are converted into predictable shape changes [2]. To analyze the slow and irreversible growth behavior across wide spatial scales, it is important to track and map lobing events in the epidermal tissue. It has been shown that cell walls perpendicular to the leaf surface, referred to as the anticlinal wall, can be used to detect new lobe formations [3, 18].

Manually segmenting the cells and identifying the anticlinal walls is not only laborious but mostly impractical. Even with partial annotations of such data, there is significant subjectivity in the precise localization of the boundaries and cell intersections. In this dissertation, we develop an automatic analysis method to detect, label, and track pavement

cells and their anticlinal walls from time-lapse 3D confocal images.

Neuron Morphology Analysis The importance of neuronal morphology has been recognized from the early days of neuroscience [19]. Neuronal morphology plays a fundamental role in information processing in the nervous system [19]. The shape, structure and connectivity of nerve cells are important aspects of neuronal function. Genetic and epigenetic factors that alter neuronal morphology or synaptic localization of pre- and post-synaptic proteins contribute significantly to neuronal output. [20]. Similar to the pavement cell analysis, this requires 3D boundary analysis which is labor intensive, subjective and does not scale well with any manual analysis.

For the neuron morphology analysis, we propose a novel method that takes 3D surface point clouds as input and builds a shape representation model for neuron classification and computing sub-cellular features. We propose to use a 3D skeleton representation for this purpose. Although there are many neuron analysis methods for neurons [20–25], we are the first to propose to use the skeleton mesh concept for neuron morphology analysis.

We specifically use our method to study the *Ciona* neuron morphology. Ciona sea squirt is one of the most widely studied tunicates in neuroscience [5]. Fig 4.11 shows how *Ciona* looks like in real life. Its brain is closely related to humans with a much simpler neuronal structure. A single *Ciona* larve neural system, it has only about 187 neurons with about 6600 synapses [5]. Studying the *Ciona* brain in depth can reveal the general principles behind the mechanism of how vertebrate brains work [26].

1.1 Challenges

Time-lapse 3D microscopy image analysis is challenging for several reasons. First, the imaging conditions vary significantly from one experiment to another, posing challenges

for basic segmentation and detection methods. Further, the 3D data in the form of image stacks is something that we do not encounter in traditional natural image analysis, so those methods are not directly applicable. A second challenge comes from lack of well annotated datasets. As noted previously, annotating 3D imaging data is very laborious. However, current machine learning methods are data hungry. For example, the well-known image dataset, ImageNet [27], has more than 14 million annotated images while one of the largest time-lapse 3D cell image dataset [10] has 6 plants with 120 image stacks with voxel-wise annotation in the whole dataset. This is further complicated by the unique nature of certain problems, such as detecting the anticlinal walls that require precise boundary point computations and a good understanding of the underlying biology for creating the annotations for training or validation.

1.2 Summary of Contributions

There are mainly four contributions of this dissertation: First, we propose the first deep learning enabled end-to-end fully automated time-lapse 3D cell analysis method for pavement cell growth process. Second, we have curated a membrane tagged imagery with partially (expert) annotated sub-cellular features and fully annotated by our computational method that is now released to the public. Third, we propose a robust neuron morphology analysis method. We utilize this method to explore the neuron morphology and function relationships. Fourth, we have curated a *Ciona* neuron structure dataset that is made available to other researchers, a first of its kind from point cloud data.

1.3 Dissertation Organization

In Chapter 2, an overview of microscopy imaging, datasets, and tools used in this cell research are presented. We introduce (i) different microscope imaging modalities, (ii) the datasets we use in this dissertation, and (iii) the tools for visualization and annotation.

In Chapter 3, we present a method for time-lapse 3D cell analysis. Specifically, we consider the problem of accurately localizing and quantitatively analyzing sub-cellular features, and for tracking individual cells from time-lapse 3D confocal cell image stacks. The chapter discusses three main components of our analysis method: 3D boundary-tagged cell segmentation method, tracking method, and sub-cellular feature method. We describe in detail how we get closed 3D surface while also maintaining high boundary segmentation accuracy. Following segmentation, we compute the adjacency graph from the segmentation mask and utilize the adjacency graph to achieve high tracking accuracy with low computation complexity. We demonstrate the generalizability of the method to nuclei tagged images.. We conclude with a novel method for computing sub-cellular features for the pavement cells that could be used in studying their growth behavior.

In Chapter 4, we provide a method to analyze 3D neuron morphology and explore the relationship between 3D neuron morphology and neuron types. Specifically, a robust unsupervised deep learning method to get the 3D skeleton model from a set of discrete 3D surface points is presented. Following the skeleton model, a skeleton graph introduced to compute cellular and sub-cellular features of the neuron such as length and number of branches and the neuron. At last, a graph embedding network that maximizes mutual information between graph level embedding and node features is used to compute embedding vectors. These graph embedding vectors are then used to explore the relationship between structure and functions of neurons, including neuron type classification.

In Chapter 5, we summarize the main methods and contributions of this dissertation,

and propose potential future work.

Chapter 2

Microscopy Cell Images, Datasets, and Tools

> Man can alter his life by altering his thinking.
>
> *William James*

This chapter introduces the image modalities, datasets, and tools used in this dissertation. The datasets include 3D time-lapse confocal A. thaliana's leaf pavement cell datasets [9, 10], 3D time-lapse fluorescent nuclei microscopy dataset of *C.elegans* developing embryo [12, 13], 3D TEM *Ciona* larva brain [5], NeuroMorpho neuron shape dataset [28], and ShapeNet with 3D CAD models [1]. We also briefly discuss the existing commonly used tools in biology research including ImageJ [29], ITK-SNAP [30], Reconstruct [31], TrakEM2 [32], MeshLab [33], and BisQue [34].

2.1 Microscopy Imaging Modalities

Biomedical research is image driven, and microscopes are the primaty tools to gather such imaging data. In our work we mainly use data from confocal microscopes and transmission electron microscopes (TEM).

2.1.1 Confocal Microscopy Imaging

Confocal imaging is a specific category of the fluorescence imaging techniques. Fluorescence microscopy is often used to capture specific features or molecules of specimens. In most cases, the sample of interest is labeled with a fluorescent substance and then illuminated through the lens with high energy source. In cellular imaging, plasma-membrane or cell nuclei are commonly labeled to be visible with the fluorescent substance. In the traditional wide-field fluorescence microscope, the imaging sample is evenly lighted. The resulting fluorescence detected by microscope's photo detector will include a large amount of unfocused background part. In contrast, a confocal microscope uses point illumination and a pinhole in an optically conjugate plane in front of the detector to eliminate unfocused signal. As only light produced by fluorescence very close to the focal plane can be detected, the image's optical resolution, particularly in the sample depth direction, is much better than that of wide-field microscopes. Since confocal microscope can only image one point in the specimen, scanning over a regular raster (i.e., a rectangular pattern of parallel scanning lines) is required to image the whole specimen. Then successive slices make up a "z-stack" to get a 3D image. Confocal microscopy provides the capacity for direct, noninvasive, serial optical sectioning of intact, thick, living specimens with a minimum of sample preparation as well as a marginal improvement in lateral resolution compared to wide-field microscopy [35].

2.1.2 Transmission Electron Microscopy (TEM)

Transmission electron microscopy (TEM) is a microscopy technique in which a beam of electrons is transmitted through a specimen to form an image. An image is formed from the interaction of the electrons with the specimen as the beam is transmitted through the specimen. The image is then magnified and acquired onto an imaging device, such as a fluorescent screen. TEMs are capable of imaging at a very high resolution than light microscopes, close to atomic scale. Due to TEM's high resolution capability, in cell biology research, TEM is used to study the morphology of cells and their organelles. TEM sample specimens have thickness limitations defined by transmission of the electron beam. Therefore, to image the whole cell, it is common to image slice by slice and then obtain 3D images by registering those slices and stacking them.

2.2 Datasets

The main datasets used in this research include the time-lapse 3D confocal pavement cell datasets, time-lapse 3D fluorescent nuclei microscopy dataset of *C.elegans*, 3D TEM *Ciona* larva brain, and 3D CAD models dataset.

2.2.1 Single Layer Pavement Dataset (Dataset 1) [2, 9] from Plant Biology Group at Purdue University

The image subject is A. thaliana's leaf epidermal tissue and it contains only single layer of cells. This dataset is used to study pavement cell growth process, especially to understand change of cell wall patterns during the growth process. In this dataset, the fluorescence is only on cell membrane and therefore, only cell membranes are tagged in each 3D image stack.

It is a long-term time-lapse dataset. The observation time span is over 12 hours with a spatial (xy-) resolution of $0.212\mu m$ and $0.5\mu m$ thick optical sections. There are 5 sequences of image stacks. Each sequence has 9-20 image stacks and each stack has 18 to 25 slices containing one layer of cells, and the image dimension of each slice is 512×512 pixels. Partial ground truth sub-cellular features are provided for this dataset. Details of this dataset are described in Table 2.1. Example 3D images of this dataset are shown in Fig. 2.1. Example 2D slices from one image stacks are shown in Fig. 2.2

Table 2.1: Single Layer Pavement Cell Dataset [2, 9]. It consists of a long-term time-lapse from *A. thaliana's* leaf epidermal tissue that spans over a 12 hour period with a xy-resolution of $0.212\mu m$ and $0.5\mu m$ thick optical sections. The time step is two hours for sequence#2 and is one hour for all other sequences. Anticlinal cell walls are partially annotated for all sequences. In addition to that, cells are partially annotated for sequence#5

Dataset	Number of Time Points	Image Stack Dimension
Sequence 1	20	$512 \times 512 \times 20$
Sequence 2	9	$512 \times 512 \times 18$
Sequence 3	10	$512 \times 512 \times 30$
Sequence 4	13	$512 \times 512 \times 21$
Sequence 5	20	$512 \times 512 \times 25$

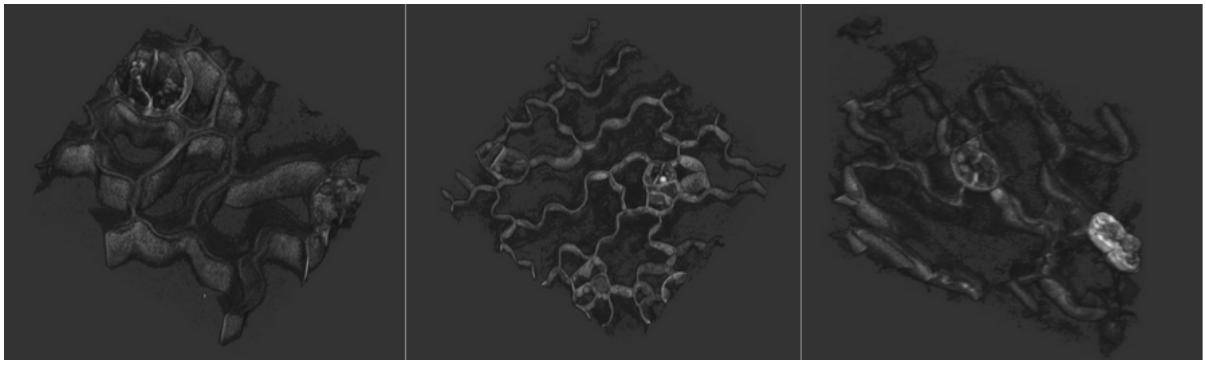

Figure 2.1: 3D example image stacks from Dataset 1. Each image is a 3D rendering of a cell membrane tagged confocal image stack from three different volumes.

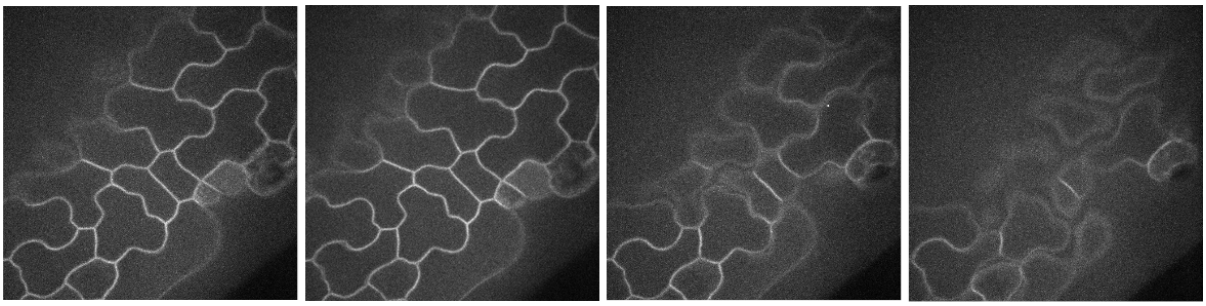

Figure 2.2: Four consecutive image slices from a single volume in Dataset 1. The data is tagged for cell membranes.

2.2.2 Multi Layer Pavement Dataset (Dataset 2) [10]

The second dataset (Dataset 2) contains cells in the shoot apical meristem of 6 *Arabidopsis thaliana* [10]. There are 6 image sequences. Each image sequence has 20 image stacks. In each image stack, there are 129 to 219 slices containing of 3 layers of cells: outer layer (L_1), middle layer (L_2), and deep layer (L_3), and the dimension of each slice is 512×512. The available resolution of each image in x and y direction are $0.22\mu m$ and in z is about $0.26\mu m$. The ground truth voxel-wise cell labels are provided, and each cell has a unique label. Each cell track also has a unique track ID. Details of this dataset are described in Table 2.2. Example 3D images of this dataset are shown in Fig. 2.3. Example 2D slices from one image stacks are shown in Fig. 2.4.

Table 2.2: Multi Layer Pavement Cell Dataset [10]. It contains three layers of cell walls in the shoot apical meristem of *A. thaliana's* that spans over 80 hours with with a xy-resolution of $0.22\mu m$ and $0.26\mu m$ thick optical sections. The time step is 4 hours for all sequences and each sequence has 20 frames. Cells with track IDs are fully provided.

Dataset	Image Stack Dimension
Sequence 1	$512 \times 512 \times 134$
Sequence 2	$512 \times 512 \times 219$
Sequence 3	$512 \times 512 \times 119$
Sequence 4	$512 \times 512 \times 129$
Sequence 5	$512 \times 512 \times 139$
Sequence 6	$512 \times 512 \times 134$

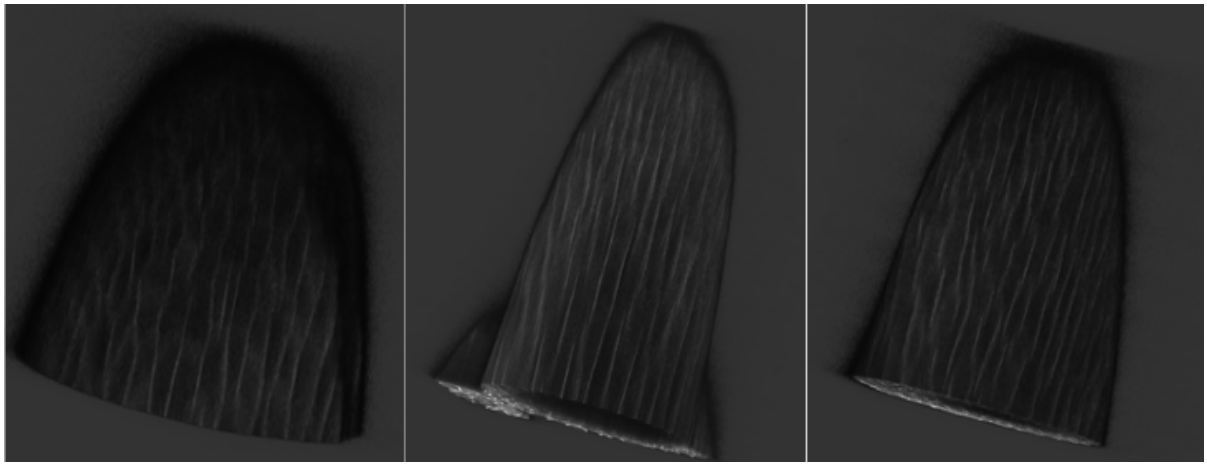

Figure 2.3: 3D example image stacks from Dataset 2. Each image is a 3D rendering of a cell membrane tagged confocal image stack. One image stack has multiple layers of cells.

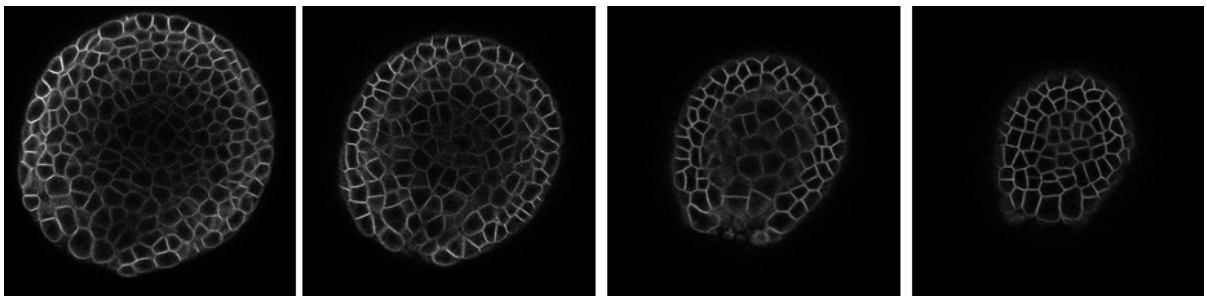

Figure 2.4: Four consecutive image slices from a single volume from Dataset 2. Each image is one xy plane of the same 3D membrane tagged confocal image stack.

2.2.3 Cell Nuclei Dataset (Dataset 3) [11–13]

The 3D time-lapse video sequences of fluorescent nuclei microscopy image of *C.elegans* developing embryo [11–13]. Each voxel size is $0.09 \times 0.09 \times 1.0$ in microns. Time points were collected once per minute for five to six hours. There are two videos in the training set and two videos in the testing dataset. Details of this dataset are described in Table 2.3. Example 3D images of this dataset are shown in Fig. 2.5. Example 2D slices from one image stacks are shown in Fig. 2.6.

Table 2.3: C.elegans Developing Embryo Nuclei Dataset [11–13]. The resolution of each image stack is $0.09\mu m \times 0.09\mu m \times 1.0\mu m$. Sequence 1 and 2 are training set which contains partial nuclei segmentation with track IDs for training. Sequence 3 and 4 are testing set so no annotations available.

Dataset	Time Step (min)	Number of Frames	Image Stack Dimension
Sequence 1	1	250	$512 \times 708 \times 35$
Sequence 2	1.5	250	$512 \times 712 \times 31$
Sequence 3	1	190	$512 \times 712 \times 31$
Sequence 4	1.5	140	$512 \times 712 \times 31$

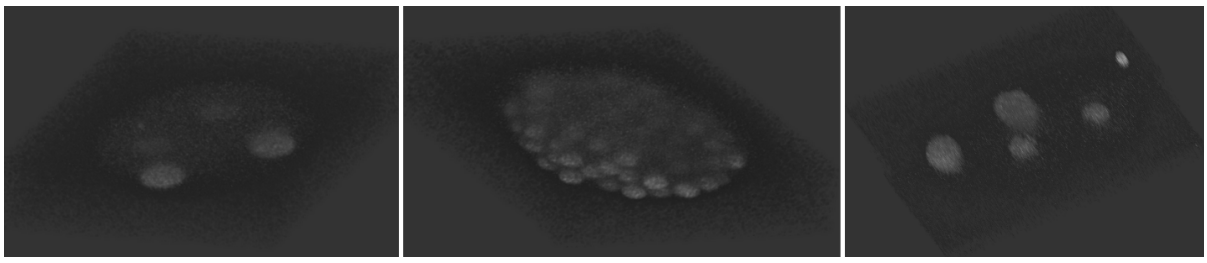

Figure 2.5: 3D example image stacks from Dataset 3. Each image is a 3D rendering of a nuclei tagged confocal image stack.

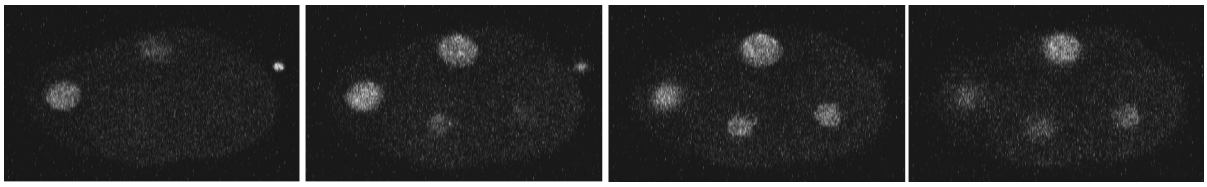

Figure 2.6: Four consecutive image slices from a single volume from Dataset 3. Each image is one xy plane of the same 3D nuclei tagged confocal image stack.

Image Format Dataset 1, Dataset 2, and Dataset 3 contain sequences of image stacks. Each image stack is in 8-bit or 16-bit Tiff image format.

2.2.4 Ciona Larva Neuron Dataset (Dataset 4) [5] from our collaborator: the Smith Lab at UCSB

The fourth dataset (Dataset 4) contains two Ciona larva 3D TEM images. The section thickness for TEM images varies between 35 nm and 100 nm. For each section, xy resolution is 3.85×3.85 nm. Animal 1 contains 7671 sections and animal 2 contains

about 8000 sections. In each Ciona larva, 187 neurons are annotated. Those 187 neurons can be grouped into 31 types. For animal 1, Ciona neurons are manually segmented using Reconstruct [31] and 3D surface point clouds are available. For animal 2, Ciona neuron skeletons are traced using TrackEM2 [32], an ImageJ [29] plugin. Details of this dataset are described in Table 2.4 Example surface point clouds are shown in Fig. 2.7.

Table 2.4: Ciona Neuron Dataset

Animal	xy resolution (nm)	section thickness (nm)	number of sections	annotations
Animal 1	3.85×3.85	35-100	7671	3D surface point cloud of neurons are provided
Animal 2	3.85×3.85	35-100	6928	3D neuron skeletons without skeleton points' radii

Figure 2.7: 3D neuron surface point examples from Dataset 4. This figure shows three sets of surface points to represent three neurons.

2.2.5 NeuroMorpho Dataset (Dataset 5) [28]

NeuronMorpho [28] is a publicly available dataset that is used for neuron morphology research. It has tens of different animals' neurons. So far, it is the largest neuron skeletons dataset with associated metadata. In this dissertation, we take a subset of C.elegans dataset (Dataset 5) from the whole NeuroMorpho dataset to verify our neuron morphology analysis method. Dataset 5 consists of 299 neuron skeletons (with radii) and it is classified into 10 different types. Each neuron with detailed metadata information

such as number of branches and length of neuron. Three example neuron skeletons are shown in Fig. 2.8.

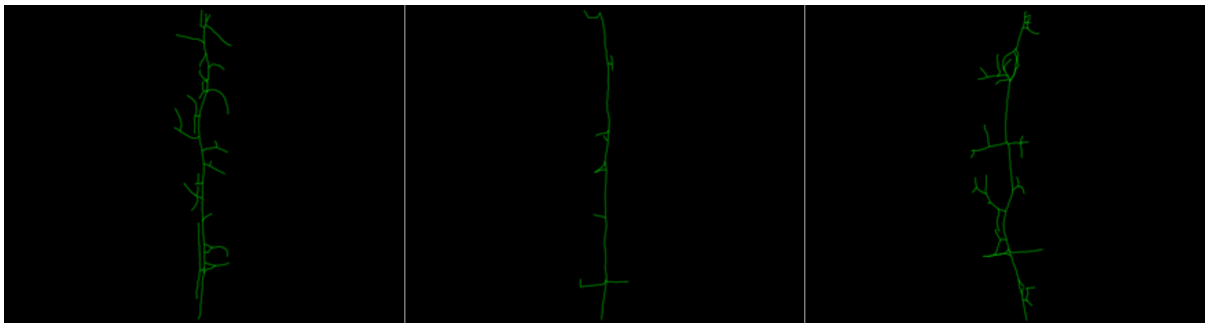

Figure 2.8: Three neuron skeleton examples from the NeuroMorpho Dataset.

2.2.6 ShapeNet (Dataset 6) [1]

ShapeNet is a richly-annotated, large-scale dataset of 3D shapes represented by 3D CAD models of objects. It contains 3D models from a multitude of semantic categories. In this research, we collect 7088 shapes from 8 categories of ShapeNet, the same as in [8]. Each shape has 2000 sampled 3D surface points. 5/6 of the data from each category are used for training and the other 1/6 for testing. The network is trained on all 8 shape categories. Fig. 2.9 shows one 3D example from each category.

2.3 Tools

In this section, we introduce the existing tools used in this research. These tools are used for visualization, registration, segmentation, and tracing cells/neurons.

Figure 2.9: 8 3D CAD shape examples from each category of Dataset 6 [1]. From left to right, top to bottom: airplane, chair, earphone, guitar, lamp, mug, rifle, and table.

2.3.1 ImageJ

ImageJ is open source software for processing and analyzing scientific images. It can display, edit, analyze, process, save, and print 8-bit color and grayscale, 16-bit integer, and 32-bit floating point images. It supports multiple image file formats, including TIFF, PNG, GIF, JPEG, BMP, DICOM, and FITS, as well as raw formats. ImageJ can handle 3D image stacks. A 3D image stack is shown in ImageJ as a series of images that share a single window. It supports standard image processing functions such as logical and arithmetical operations between images, contrast manipulation, convolution, Fourier analysis, sharpening, smoothing, edge detection, and median filtering. It does geometric transformations such as scaling, rotation, and flips. It can also be used to annotate user defined features and save it as ROI files. In this dissertation, ImageJ is mainly used to visualize confocal image stacks and to annotate pavement cell sub-cellular features in Chapter 3. Fig. 2.10 shows the visualization of one image stack from Dataset 1 and their sub-cellular feature.

Another important feature of ImageJ is that it is designed with an open architecture that provides extensibility via Java plugins and recordable macros. Custom acquisition, analysis and processing plugins can be developed using ImageJ's built-in editor and a Java compiler. TrakEM2, which we will introduce later, is an example of ImageJ plugin.

2.3.2 ITK-SNAP

ITK-SNAP [30] is a free, open-source, and multi-platform software application used to visualize and segment structures in 3D biomedical images. It provides manual and semi-automatic segmentation tools using active contour methods. It also offers image navigation for 3D biomedical images. The main advantages of ITK-SNAP include (1) support for multiple 3D image formats, including NIfTI and DICOM, (2) support for

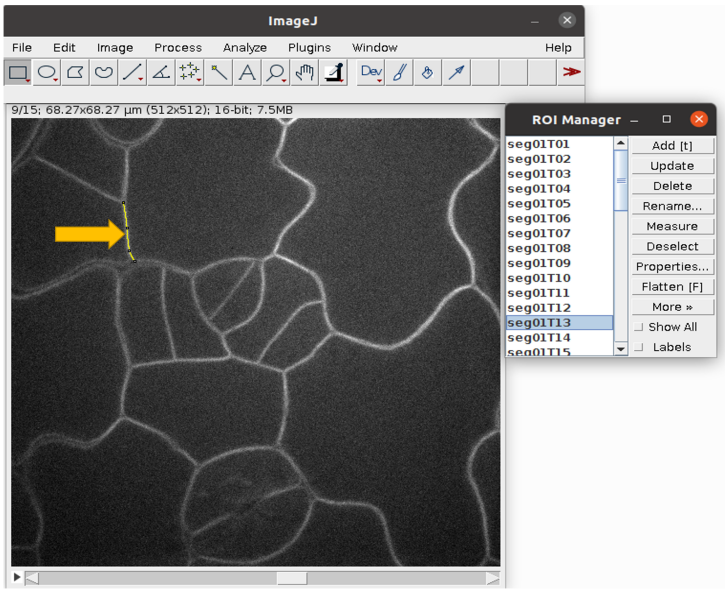

Figure 2.10: An image slice from Dataset 1 visualized using ImageJ. Various subcellular features, such as boundary segments, are manually annotated as shown in the metadata table on the right. Yellow arrow points to one such segment.

multi-channel images, (3) overlaying 3D segmentation image over raw image stacks, and (4) manual segmentation in the sagittal, coronal, and transverse planes at once. Fig. 2.12 and Fig. 2.11 show examples of using ITK-SNAP for visualizing the segmentation mask of pavement cells and nuclei. ITK-SNAP is used for visualization purposes in our research.

2.3.3 Reconstruct

For EM images, especially TEM images, it requires reconstruction of serial section images to form the 3D stack. When each section is cut, mounted and imaged separately, section images must be montaged and realigned to accurately analyse and visualize the three dimensional (3D) structure. Reconstruct is a free editor designed to facilitate montaging, alignment, analysis and visualization of serial sections. The methods used by Reconstructfor organizing, transforming and displaying data enable the analysis of

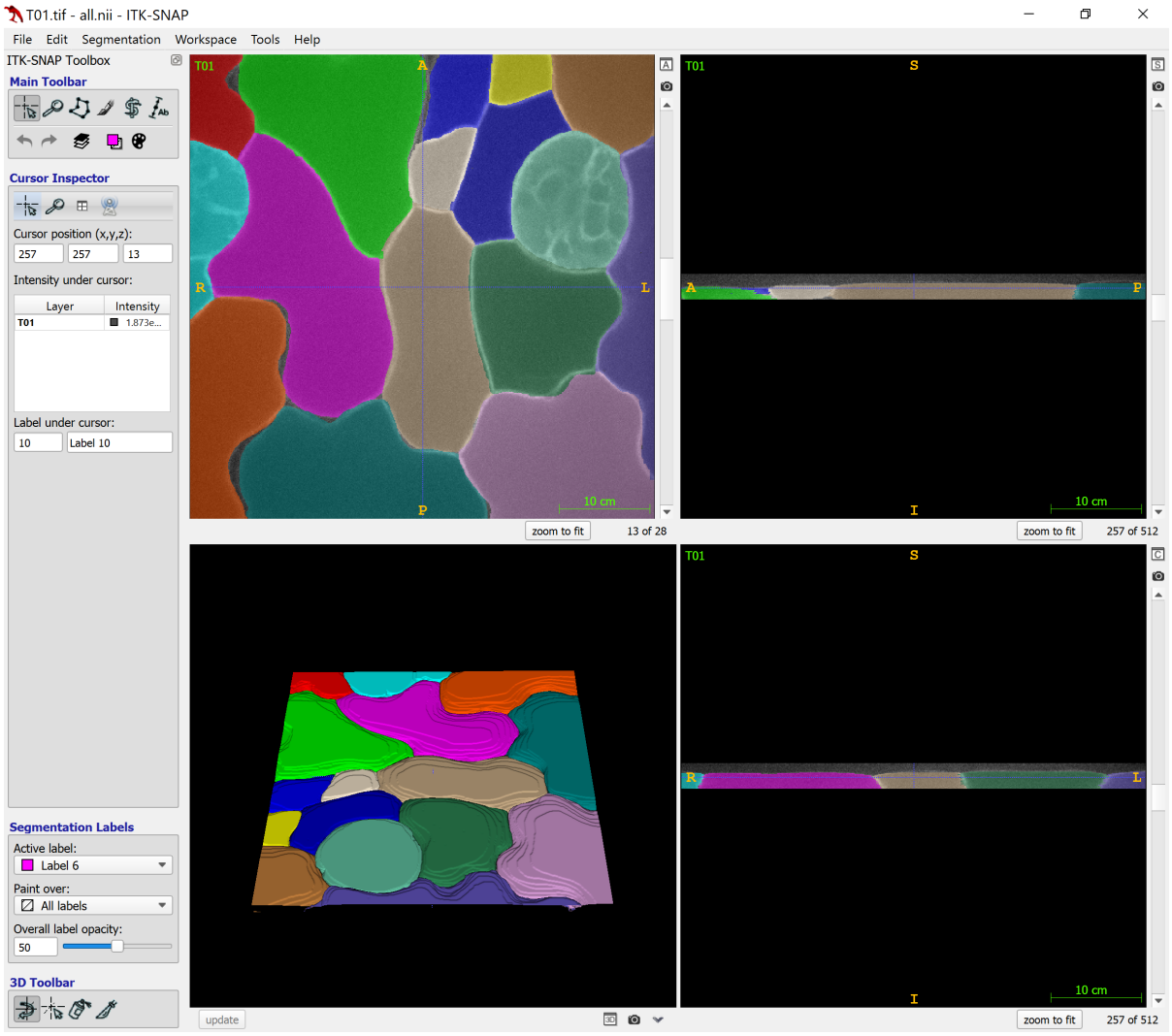

Figure 2.11: Visualizing segmentation mask overlaying on the original pavement cell image stack. Top left, top right, and bottom right windows show the axial, sagittal, and coronal planes of segmentation mask overlaying on the boundary-tagged confocal images, respectively. Bottom left window shows the 3D reconstruction from the labels.

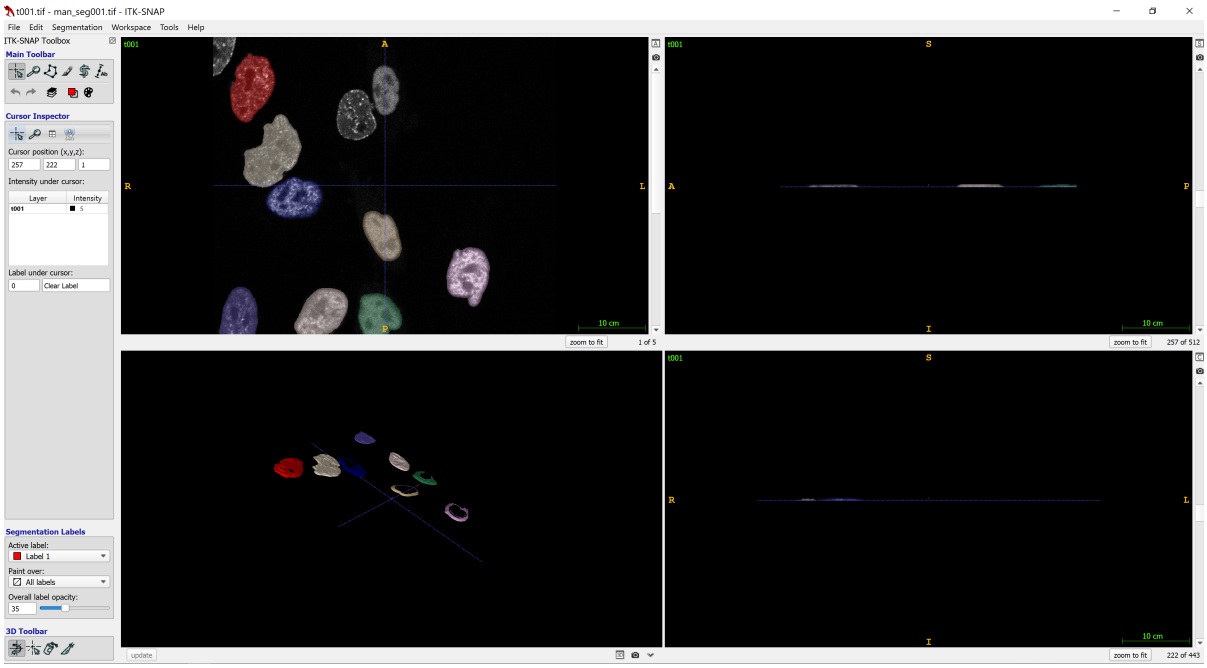

Figure 2.12: Visualizing segmentation mask overlaying on the original nuclei image stack from Dataset 3. Top left, top right, and bottom right windows show the axial, sagittal, and coronal planes of segmentation mask overlaying on the nuclei-tagged confocal images, respectively. Bottom left window shows the 3D reconstruction from the labels.

series with large numbers of sections and images over a large range of magnifications by making efficient use of computer memory. Alignments can correct for some types of non-linear deformations, including cracks and folds, as often encountered in serial electron microscopy. A large number of different structures can be easily traced and placed together in a single 3D scene that can be animated or saved. As a flexible editor, Reconstruct can reduce the time and resources expended for serial section studies and allows a larger tissue volume to be analysed more quickly. Reconstruct is used to annotate each of the animal one neurons from Dataset 4, which in turn is used to visualize the surface mesh of neurons as illustrated in Fig. 2.13

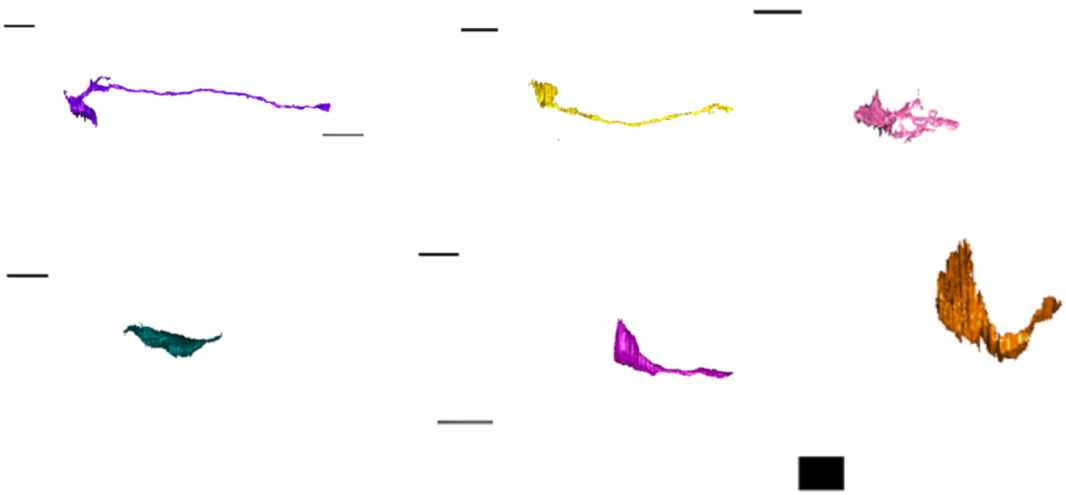

Figure 2.13: Visualizing 3D neuron meshes using Reconstruct. Different colors of the mesh represent different neurons. There are six neurons in this figure.

2.3.4 TrakEM2

TrakEM2 is an ImageJ plugin for morphological data mining, three-dimensional modeling and image stitching, registration, editing and annotation. In this dissertation,

TrakEM2 is used to annotate animal 2 TEM images of ciona dataset. Specifically, TrackEM2 is used to register serial section EM images, trace neurons from the registered TEM images, and visualize 3D annotations of neurons. Fig 2.14 and 2.15 demonstrate example EM images and 3D neurons in TrakEM2.

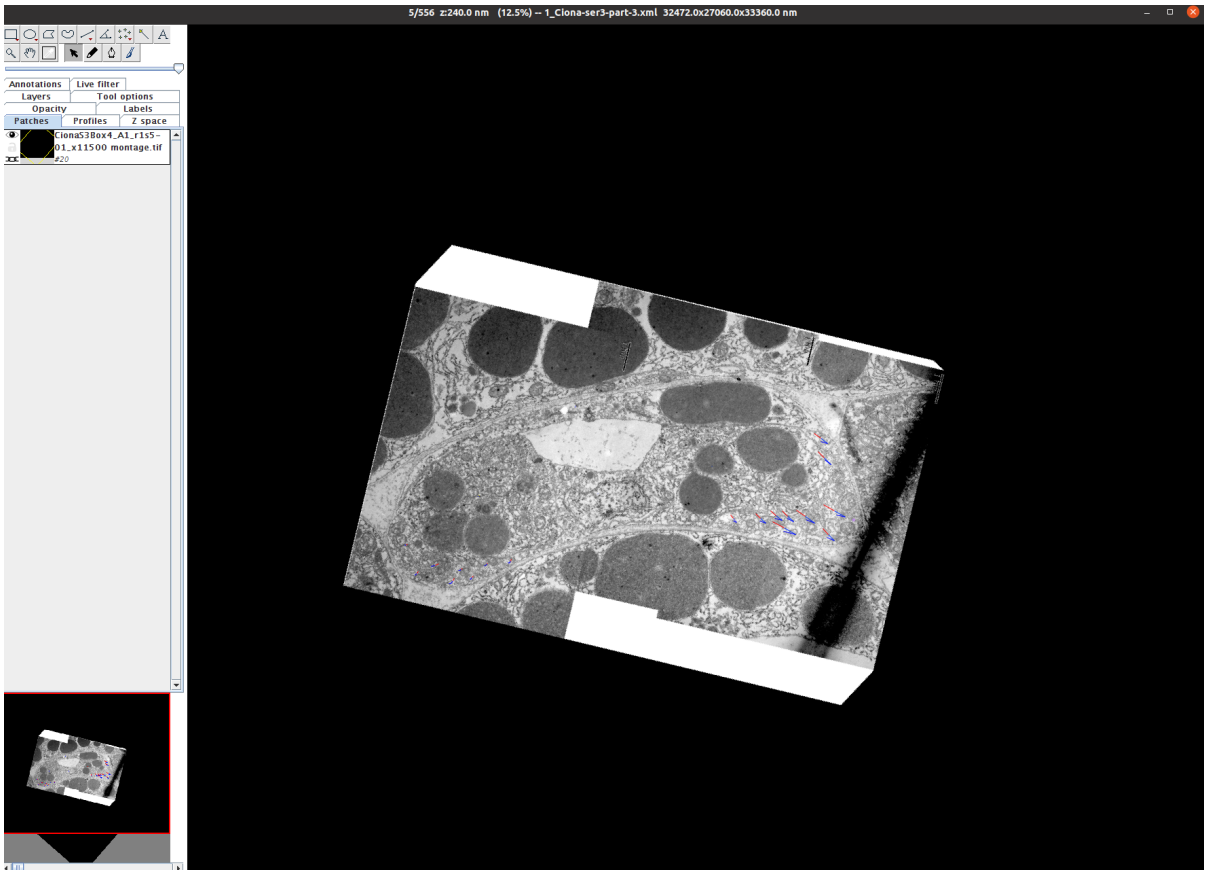

Figure 2.14: Visualizing a set of 3D TEM image stack in TrakEM2. The figure shows one section (slice) of 3D TEM from 3D TEM image stack. This image is also manually registered to a reference image in TrakEM2.

2.3.5 MeshLab

MeshLab is an open source system to process 3D shapes. Specifically, it is oriented to the management and processing of unstructured large meshes and provides a set of

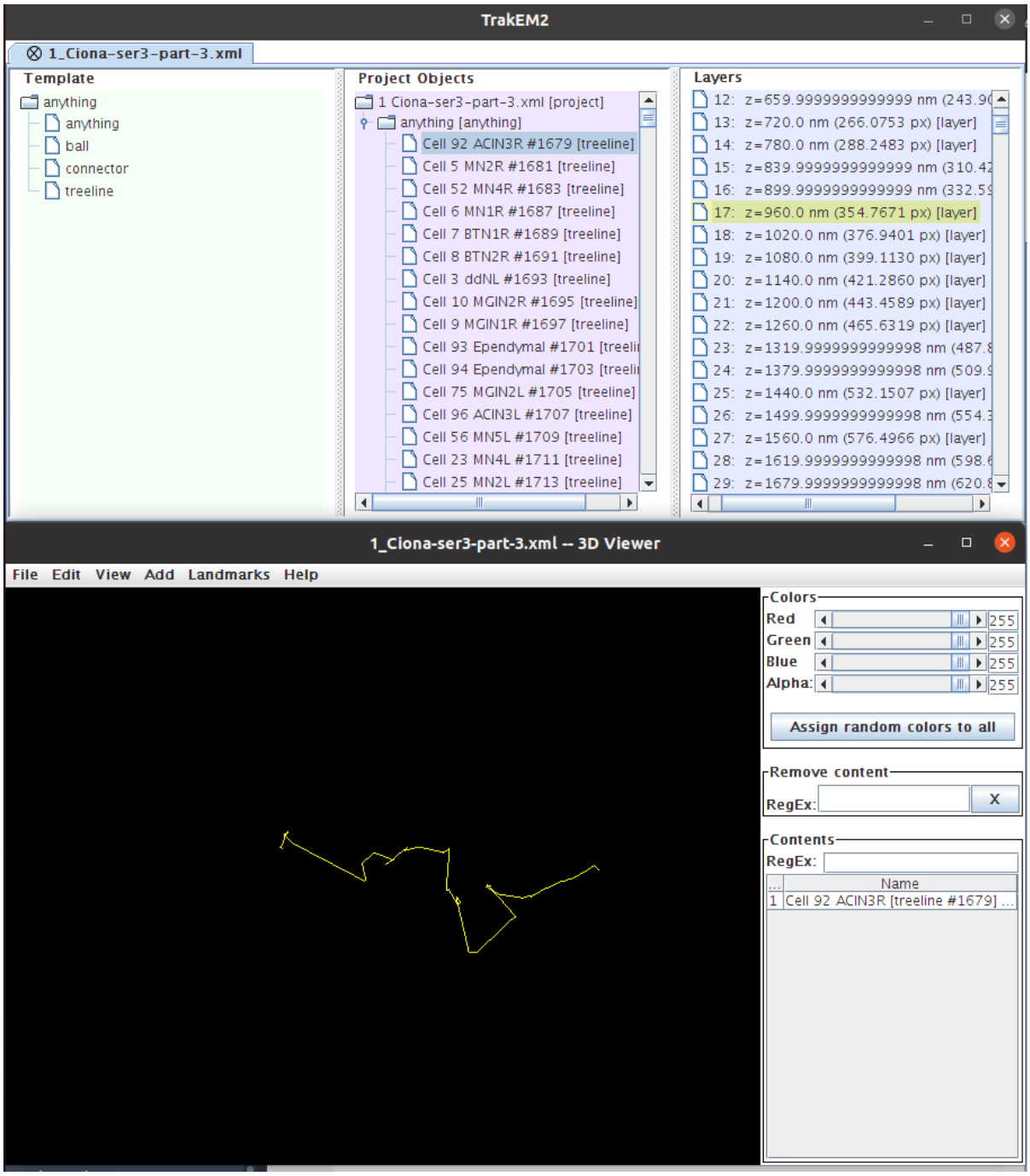

Figure 2.15: 3D visualization of a single neuron skeleton in TrakEM2. Top window shows a list of neuron skeletons and the bottom window shows the skeleton of a neuron called ACIN3R.

tools for editing, cleaning, healing, inspecting, rendering, and converting meshes. In this dissertation, we use it for visualization of point clouds and meshes. Fig. 2.16 and Fig. 2.17 show the visualization of point cloud and mesh in MeshLab.

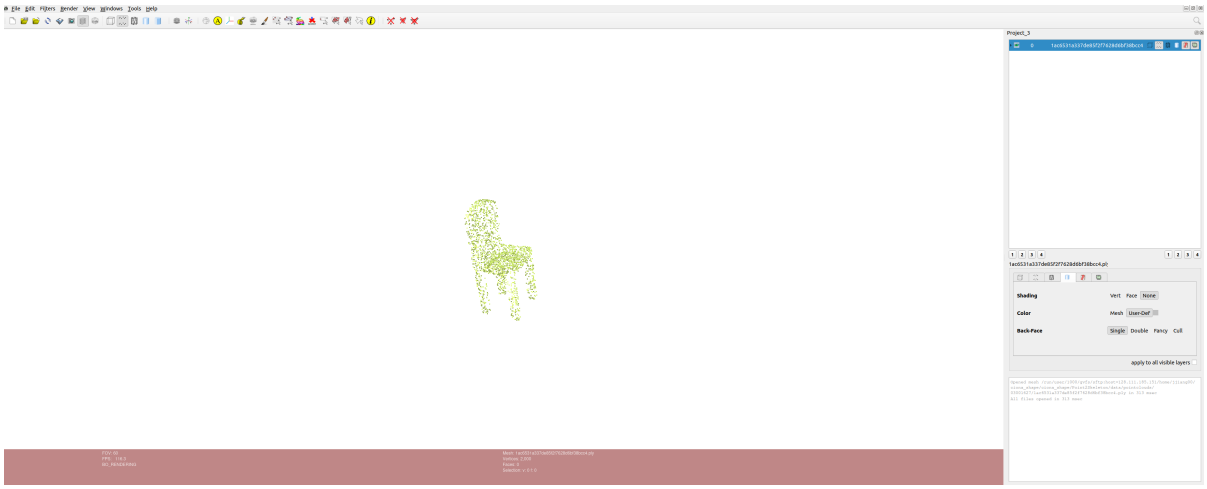

Figure 2.16: 3D visualization an object's point clouds in MeshLab. The object is a chair in this case.

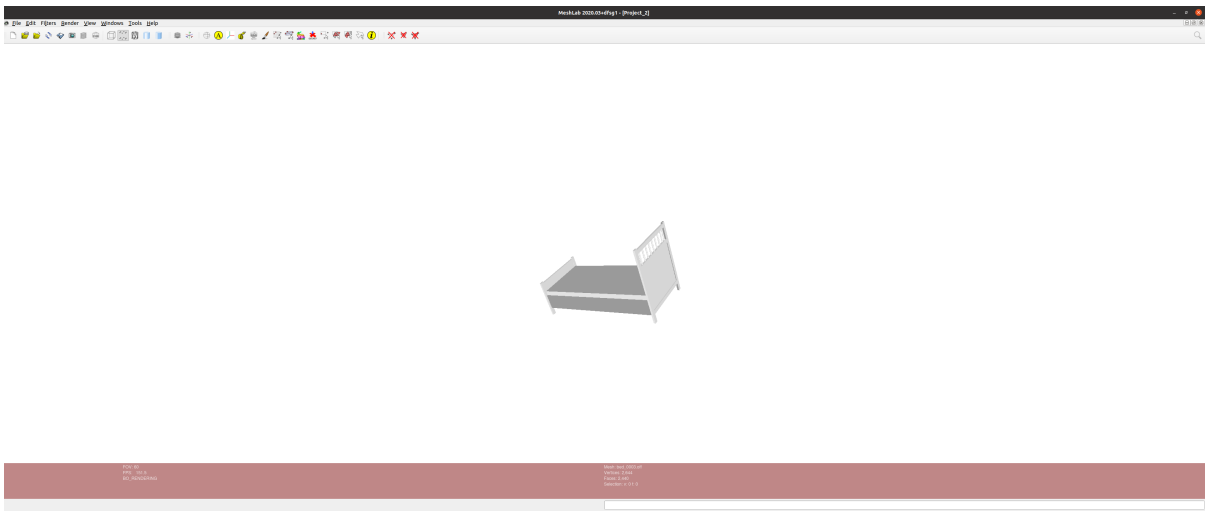

Figure 2.17: 3D visualization of a surface mesh in MeshLab. This is a bed's surface mesh.

2.3.6 BisQue

BisQue is a web-based large scale image processing infrastructure that is developed and maintained at UCSB. It is used to organize, share, annotate, and analyze different images in the cloud. It is designed for the exchange and exploration of biology images and it supports various commonly used biomedical image formats. The bisque system is centered around a database of image and metadata. Novel semantic analysis methods are integrated into the system. Fig. 2.18 shows screenshots of BisQue. We can see data are grouped by datasets and several biomedical image analysis methods available on BisQue.

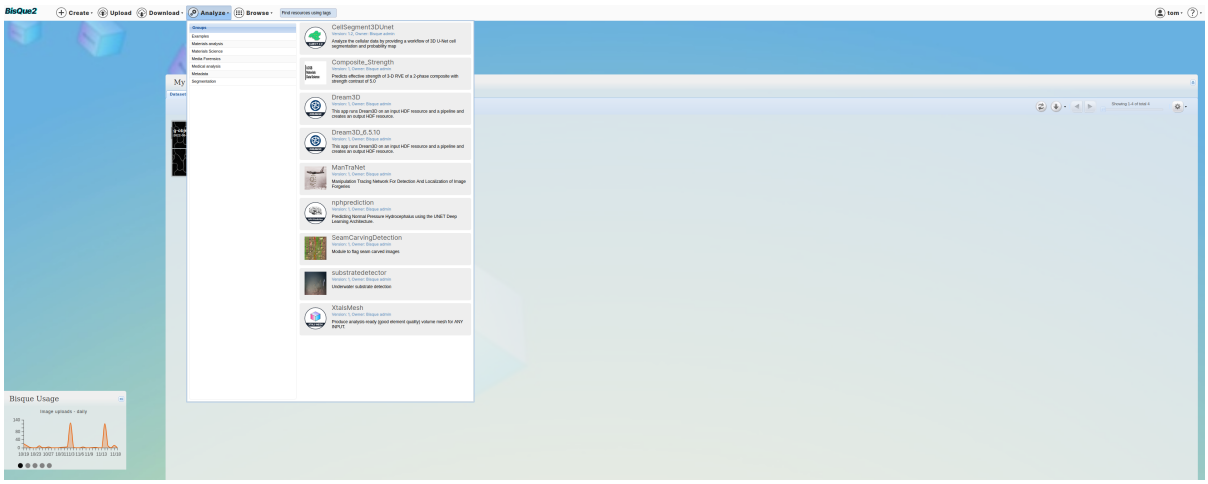

Figure 2.18: BisQue Overview: Datasets and Image Analysis Methods. BisQue can organize a set of data in a dataset and it also provides a set of analysis methods for different datasets.

In this dissertation, we use BisQue for two parts. First, we use BisQue to get and share image processing results with our collaborators. Second, we integrate our analysis method into BisQue as a service for our collaborators or wider communities to use our novel image analysis method. Fig. 2.19 shows our developed method in BisQue.

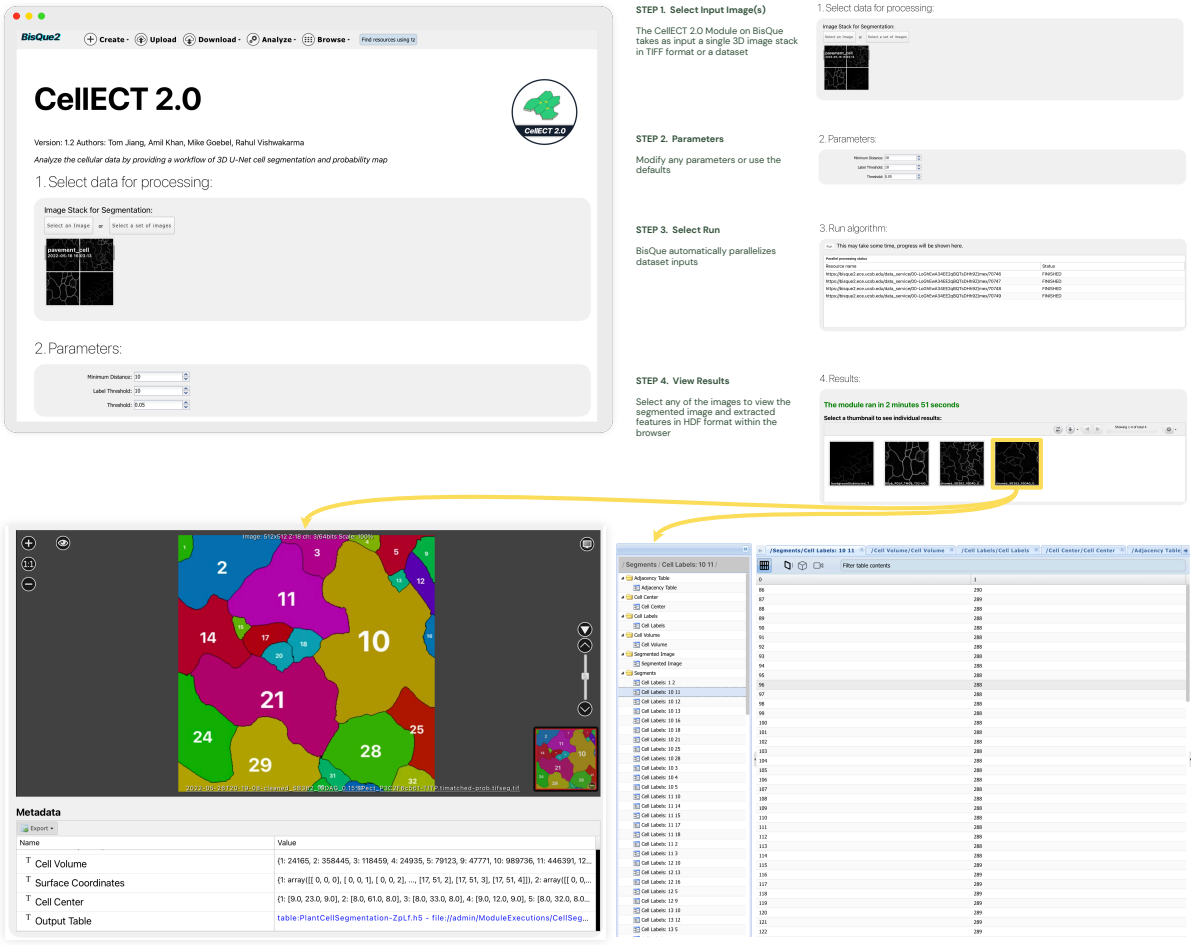

Figure 2.19: Our analysis method is named CellECT2.0 Module in BisQue. Top left image shows we have an example of four 3D stacked images from different time sequences in TIFF format as input to our module. BisQue will automatically parallel process the input images and return the results for each image. Top right shows the running steps and processes of the module. Bottom left shows the segmentation mask. Each segmented cell is labeled with an integer in the segmentation image and each of the outputs have an associated HDF file containing the cellular and sub-cellular features. Bottom right shows the cellular/sub-cellular features in a HDF file.

Chapter 3

Deep Learning Enabled Time-Lapse 3D Cell Analysis

> The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.
>
> *Albert Einstein*

In this chapter, we present a method for time-lapse 3D cell analysis. Specifically, we consider the problem of accurately localizing and quantitatively analyzing sub-cellular features, and for tracking individual cells from time-lapse 3D confocal cell image stacks. The heterogeneity of cells and the volume of multi-dimensional images presents a major challenge for fully automated analysis of morphogenesis and development of cells. This chapter is motivated by the pavement cell growth process, and building a quantitative morphogenesis model. We propose a deep feature based segmentation method to accurately detect and label each cell region. An adjacency graph based method is used to

extract sub-cellular features of the segmented cells. Finally, a robust graph based tracking algorithm using multiple cell features is proposed for associating cells at different time instances. We also demonstrate the generality of our tracking method on C.elegans fluorescent nuclei imagery. Extensive experiment results are provided that demonstrate the robustness of the proposed method.

3.1 Introduction

The sizes and shapes of leaves are key determinants of the efficiency of light capture in plants, and the overall photosynthetic rates of the canopy is a key determinant of yields [36]. The rates and patterns of leaf expansion are governed by the epidermal tissue [37] but understanding how the irreversible growth properties of its constituent jig-saw-puzzle piece cells related to organ level shape change remains as a major challenge.

The epidermal cell, also known as pavement cell, undergoes a dramatic transformation from a slightly irregular polyhedral cell to a highly convoluted and multi-lobed morphology. The interdigitated growth mode is widespread in the plant kingdom [17], and the process by which lobing occurs can reveal how force patterns in the tissue are converted into predictable shape change [2]. To analyze the slow and irreversible growth behavior across wide spatial scales, it is important to track and map lobing events in the epidermal tissue. It has been shown that cell walls perpendicular to the leaf surface, the anticlinal wall as illustrated in Fig. 3.2, can be used to detect new lobe formations [3, 18].

Time-lapse image stacks from 3D confocal imagery provide a good resource to study the pavement cell growth process, and to build a quantitative cell morphogenesis model [2, 9]. 3D confocal microscopy data contain large amount of cell shape and sub-cellular cell wall structure information. Cell analysis requirements include detecting sub-cellular features such as junctions of three cell walls and segments shape of anticlinal cell walls

used to detect lobes, all of which depends on accurate segmentation. These sub-cellular features are illustrated in Fig. 3.2. Currently, these features are usually acquired manually from 3D image stacks. Manual extraction and analysis is not only laborious but also prevents evaluation of large amounts of data necessary to map relationships between lobe formation to leaf growth.

Existing automatic time-lapse cell analysis methods include mainly two steps: (1) Detecting and segmenting cells and cell walls spatially and tracking cells in the temporal dimension, and (2) cellular/sub-cellular feature extraction.

There is an extensive literature on cell segmentation [38–47] and tracking [48, 49]. In [39–41, 45, 50] morphological operations are first used to denoise the images followed by watershed or level set segmentation methods to get the final cell segmentation. In [47], the nuclei information is provided for accurate cell segmentation. However, these methods do not provide accurate localization of the cell wall features that are needed for quantification of the growth process. In [38, 43, 44], they focus on improving the cell boundary segmentation accuracy. In [43, 44], they treat the cell segmentation problem as a semantic segmentation problem, using Generative Adversarial Networks (GAN) to differentiate between boundary pixels, cell interior, and background. These methods provide respectable accuracy on cell boundaries but they are not guaranteed to give a closed cell surface. In 2D cell segmentation, the method proposed in [46] can provide closed 2D surface while maintaining good 2D cell segmentation boundary results. There are no existing method that can provide closed 3D surface while maintaining good 3D cell wall results. It is challenging to do the downstream cell analysis such as cell tracking without a closed 3D cell surface. There are also some methods that work on 2D image slices of 3D image stack individually and then fuse the results to get a 3D segmentation [39, 45]. Finally, post-processing using cell volume or shape heuristic is required for these 2D based methods. Shape heuristics can be difficult to obtain and it restricts the

generality of those methods. For these traditional workflow of cell segmentation, post-processing is subjective, and its parameters are highly dependent on the data. In this chapter, one of the problems we aim to solve is to accurately identify cell boundaries and label individual cells in confocal microscopy image stacks. The goal is to generate closed cell surfaces in the 3D image stack while being able to accurately delineate features of interest such as the inter-cellular spaces and *protrusions*, see Fig. 3.1.

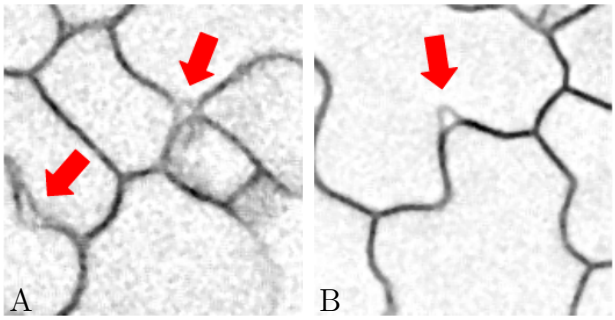

Figure 3.1: (A) Inter-cellular spaces and (B) *Protrusion* are indicated by red arrows.

Based on segmentation or detection of cells, [48,49] rely on Viterbi algorithm to track cells. They require the global optimization which is inefficient to get the cell trajectory because it's time complexity is $\mathcal{O}(TM^4)$ where T is the length of the time sequence and M is the maximum number of cells/nuclei. Also, they do not explicitly tackle cellular events such as mitosis and deaths, and the Viterbi algorithm makes a strong assumption that cells move according to the Brownian motion model, which is known to be inadequate [51].

This chapter presents a robust, time-lapse cell analysis method building upon our work [38]. In [38] we use Conditional Random Field (CRF) to get the improved cell boundaries while maintaining a closed cell surface. To make the segmentation method more robust to different datasets, we propose a modification to [38] that incorporates rotation invariance in the 3D convolution kernels [52]. A segmentation map labeling each individual cell in the 3D stack is thus created and a cell adjacency graph is constructed from this map. The adjacency graph is an undirected weighted graph with each vertex

representing a cell and the weight on the edge representing the minimum distance between two cells. Based on this adjacency graph, sub-cellular features illustrated in Fig. 3.2 are computed. The cells are tracked by comparing the corresponding adjacency graphs in the time sequence. Details of the complete workflow are described in Section 3.2.

We demonstrate the performance of the proposed *segmentation* method on multiple cell wall tagged datasets. To demonstrate generality of our *tracking* method, we evaluate it on both cell wall tagged and nuclei tagged imagery. The methods described in this chapter are published in [38, 52, 53].

In summary, the main contributions of this chapter include

- The first deep learning enabled end-to-end fully automated time-lapse 3D cell analysis method;
- A new 3D cell segmentation network with rotation equivariance that is robust to different imaging conditions;
- CRF based postprocessing method that is sensitive to 3D local boundary-tagged image features;
- A novel graph based method for multiple instance tracking and sub-cellular feature extraction as well as the novel evaluation metrics to evaluate sub-cellular feature extraction accuracy;
- We will release a new membrane tagged imagery with partially (expert) annotated sub-cellular features and fully annotated by our computational method.

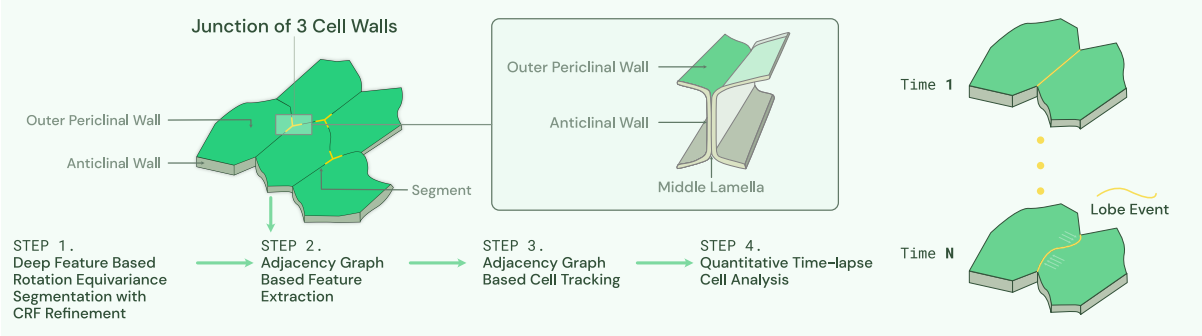

Figure 3.2: Workflow of proposed method. Modified from [2]. Given a sequence of 3D image stacks, deep feature based rotation equivariance deep learning model with CRF refinement is used to segment each cell. Then adjacency graph is built based on segmented image and used for sub-cellular feature extraction and tracking. Sub-cellular features such as junction of three cell walls and anticlinal wall segment are illustrated in the figure. Next detected segments will be used in [3] to detect lobes. This chapter mainly focuses on Step 1 to Step 3.

3.2 Method

Our cell analysis method is illustrated in Fig. 3.2. First, we segment cells from each image stack in the time sequence. Second, the adjacency graph is built based on segmented images and is used to compute sub-cellular features and cell tracking features. Finally, quantitative measurements of the cell segmentation (cell wall, cell count, cell shape), sub-cellular features (junctions of three cell walls detection accuracy, anticlinal wall segment shape), and tracking results are provided.

3.2.1 Segmentation

The cell segmentation workflow is illustrated in Fig. 3.3A with rotation equivariance constrained enforced as shown in Fig. 3.3B. 3D U-Net is a reliable method for semantic segmentation specifically for biomedical images, and 2D rotation equivariance has shown its robustness to input image orientation [54]. Therefore, we first use a rotation equivariance 3D U-Net to generate a probability map of each voxel being a cell wall. The

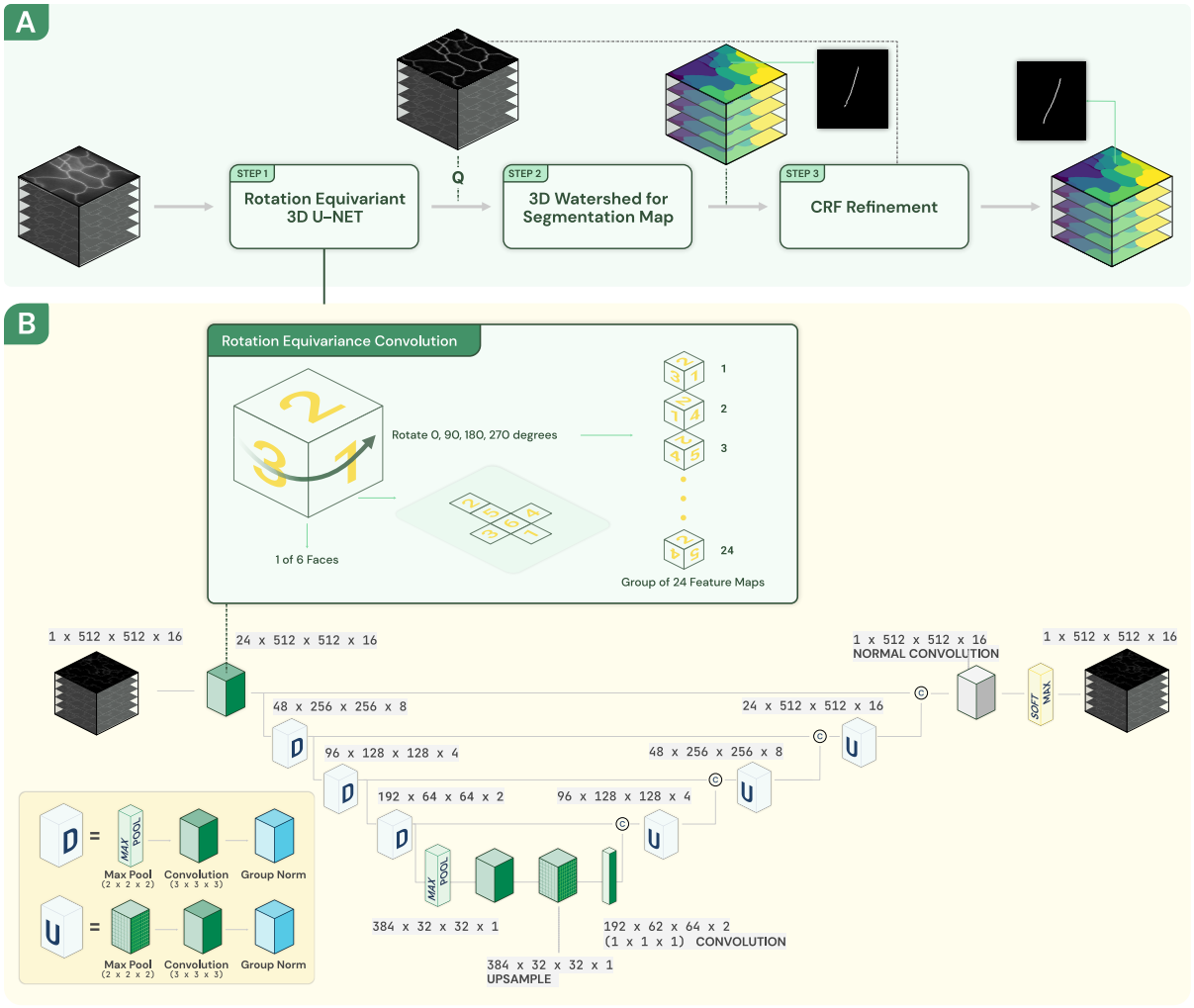

Figure 3.3: A. Segmentation workflow includes rotation equivariant 3D U-Net, 3D watershed, and CRF refinement. B. In 3D equivariant U-Net, all convolution layers are rotation equivariant convolution layers. The raw 3D image stack is truncated into 16 slices and then input to 3D equivariant U-Net.

full 3D U-Net rotation equivariance is achieved by replacing all convolution layers with rotation-equivariant layers described in the next paragraph. Second, to make sure we can get closed cell surfaces, a 3D watershed algorithm whose seeds are generated automatically is applied to the cell wall probability map, and outputs the initial cell segmentation result. The initial cell segmentation boundary is closed but may not be smooth because watershed segmentation is sensitive to noise. Finally, a conditional random field (CRF)

model is used to refine the cell boundaries of the initial cell segmentation. The CRF model takes the cell wall probability map and initial cell segmentation labels as input and outputs a smooth and closed cell wall. In the following paragraphs, we will discuss the details of how to generate the cell wall probability map, our rotation-equivariant convolution layers and the use of CRF to refine the cell segmentation boundary.

Deep Feature Map Generation

We use a rotation-equivariance 3D U-Net to generate the cell wall probability map. We replace all convolutional layers in 3D UNet encoding part with 3D rotation-equivariance convolutional layers. The purpose of doing this is to get more abstract features that are invariant to some 3D rotations.

3D rotation-equivariant layers are a generalization of convolution layers and are equivariant under general symmetry groups, such as the group of four 90° 2D rotations [54]. The corresponding 3D rotation group has 24 rotations as illustrated in Fig. 3.3 (A cube has 6 faces and any of those 6 faces can be moved to the bottom, and then this bottom face can be rotated into 4 different positions). To achieve this, convolution operations on feature maps are operating on a group of features which implies that we should have feature channels in groups of 24, corresponding to 24 rotations in the group.

For the normalization layers in our deep feature map generation, we use group normalization [55]. In our model, the input are the sub-slices of 3D stack images (of dimensions $512 \times 512 \times 16$). Due to computational limitations, we are able to input only 2 batches at a time. With this small batch size, standard batch normalization is unstable and tends to have a higher error so we use group normalization instead. The default number of

groups is set to 4. We use the Huber loss function for training, defined as:

$$L_\delta(y, \hat{y}(I)) = \begin{cases} \frac{1}{2}(y - \hat{y}(I))^2, \text{if } |y - \hat{y}(I)| < \delta \\ \delta|y - \hat{y}(I)| - \frac{1}{2}\delta^2, \text{otherwise} \end{cases} \tag{3.1}$$

where $y \in \{0, 1\}$ is ground truth label representing whether the voxel is membrane, I is the input image, and $\hat{y} \in [0, 1]$ is the regression function which is learned by the 3D U-Net. The reason we use Huber loss is that the network is used to learn the regression function but not the classification function. This loss function is quadratic when the predicted output is close to the ground truth and linear when the prediction is far from the ground truth. δ determines the threshold value between quadratic and linear loss. This loss function is differentiable compared to a mean absolute loss function and less sensitive to outliers than a mean squared loss function. Fig. 3.4(B) shows an example of the membrane probability map generated by the 3D U-Net.

3D Watershed Segmentation

A standard 3D watershed segmentation algorithm is now applied to the 3D probability map. The challenging part of the watershed segmentation is to find seeds for each cell. In most cell images with a stained or fluorescent-tagged nucleus, the nuclei can be used as seeds. However, in surface labeled cell images, nuclei information is not tagged and the seed points need to be automatically generated.

Towards this, we use a standard Otsu's thresholding on the probability map to get a binary image, and then compute a 3D distance transform on this binary image. The distance transform gives the minimum distance with respect to membrane for each voxel. In order to generate one seed within each cell, a H-maxima transform [56] is applied. The H-maximum transform suppresses all maxima in this distance transform map whose

value is less than a threshold H compared to surrounding voxel values. The remaining maxima are the seed locations, to which the 3D watershed method is applied. Fig. 3.4(C) shows the result of 3D watershed segmentation.

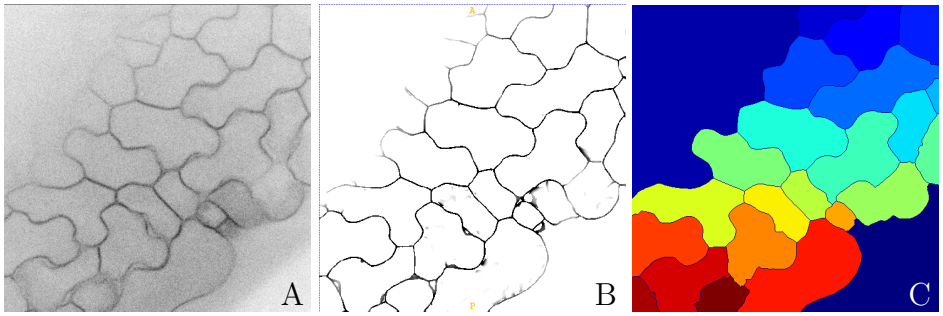

Figure 3.4: (A) Inverted raw image in xy orientation, (B) inverted probability map from the 3D U-Net, (C) initial segmentation result from 3D watershed.

CRF Refinement

The watershed based approach is sensitive to the image signal and likely to smooth out the boundaries and also miss smaller features that are significant in modeling cell morphogenesis. For this purpose, we introduce a final processing step based on conditional random fields (CRFs). The CRF refines the watershed boundaries taking into account the probability map and local voxel boundary information.

For a given cell wall probability map $\mathbf{Q}$ and cell labels $\mathbf{X}$, the conditional random field is modeled by the Gibbs distribution,

$$P(\mathbf{X}|\mathbf{Q}) = \frac{1}{Z(\mathbf{Q})} \exp(-\frac{1}{T} E(\mathbf{X}|\mathbf{Q})) \tag{3.2}$$

where denominator $Z(\mathbf{Q})$ is the normalization factor. T is a constant called the temperature and we set it to be 1. The exponent is the Gibbs energy function and we need to minimize the energy function $E(\mathbf{X})$ to get the final refined label assignments (for notation convenience, all conditioning is omitted from this point for the rest of the paper).

In the dense CRF model, the energy function is defined as

$$E(\mathbf{X}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \tag{3.3}$$

where i and j are the indices of each voxel which iterate over all voxels in the graph, and x_i and x_j are the cell labels of vertices i and j. $i, j \in \{1, 2, ..., N\}$ and N is the total number of voxels in the image stack. $x_i, x_j \in \{0, 1, 2, ..., L\}$ and L is the total number of cells identified by the watershed method (0 is the background class). The first term of eq. 3.3, the unary potential, is used to measure the cost of labeling i_{th} voxel as x_i and it is given by $\psi_u(x_i) = -\log P(x_i)$, where $P(x_i)$ is the probability of voxel i having the label x_i. It is initially calculated based on the cell wall probability map $\mathbf{Q}$ and the label image of the watershed $\mathbf{X}^0$ (The superscript 0 is used to denote the initial cell label assignment after watershed). $P(x_i^0) = 1 - q_i$ if voxel i is inside the cell with label x_i^0 after the watershed or if x_i^0 is the background label, and $P(x_i^0) = 0$ otherwise. q_i is the i_{th} voxel value in the probability map from the rotation equivariant 3D U-Net. $1 - q_i$ represents the probability of voxel being the interior point of the cell. The unary computation is illustrated in Fig. 3.5.

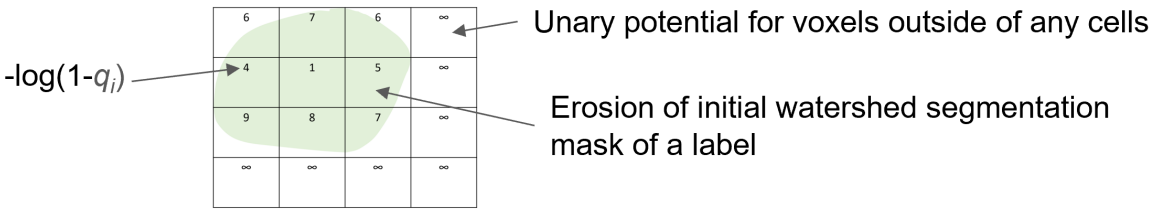

Figure 3.5: Unary potential from initial watershed segmentation mask.

The pairwise potential in eq. 3.3 takes into account the label of neighborhood voxels to make sure the segmentation label is closed and the boundary is smooth [57]. It is

given by:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_m w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) \tag{3.4}$$

where the penalty term $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, and $\mu(x_i, x_j) = 0$ otherwise. $w^{(m)}$ is the weight for each segmentation label $m \in \{0, 1, 2, ..., L\}$, and $k^{(m)}$ is the pairwise kernel term for each pair of voxels i and j in the image stack regardless of their distance that capture the long-distance voxel dependence in the image stack. $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors from the probability map $\mathbf{Q}$. $\mathbf{f}_i$ incorporates location information of voxel i and the corresponding value in the probability map: $\mathbf{f}_i =< \mathbf{p}_i, q_i >$ where $\mathbf{p}_i =< x_i, y_i, z_i >$, and x_i, y_i and z_i are the voxel i in the normalized coordinates in the range $[0, 1]$. Specifically, the kernel $k(\mathbf{f}_i, \mathbf{f}_j)$ is defined as

$$k(\mathbf{f}_i, \mathbf{f}_j) = \gamma_1 \exp(-\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\sigma_\alpha^2} - \frac{||q_i - q_j||^2}{2\sigma_\beta^2}) + \gamma_2 \exp(-\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\sigma_\gamma^2}) \tag{3.5}$$

where the first term depends on voxel location and the corresponding voxel value in probability map. The second term only depends on the voxel location. σ_α, σ_β, σ_γ, γ_1, and γ_2 are the hyperparameters in eq. 3.5. Based on our experiments, we have empirically chosen $\sigma_\alpha = 3$ and $\sigma_\beta = 5$, as these values work over a wide range of experimental data. These two hyperparameters control the degree of nearness and similarity of the probability map within a segmented region. σ_γ is determined by the smallest possible segmentation region (cell size) allowed. γ_1, and γ_2 are weights for the loss function. The detailed explanations of each hyperparameter can be found in [57]. Finally, we pick the best label assignment $\mathbf{X}^*$ as the final cell segmentation that minimizes energy function $E(\mathbf{X})$. An efficient CRF inference algorithm described in [57] is used to find $\mathbf{X}^*$ which is the final cell segmentation mask. In our experiments, σ_γ is set to be 10, and γ_1, γ_2 are set to be 1 and 1.

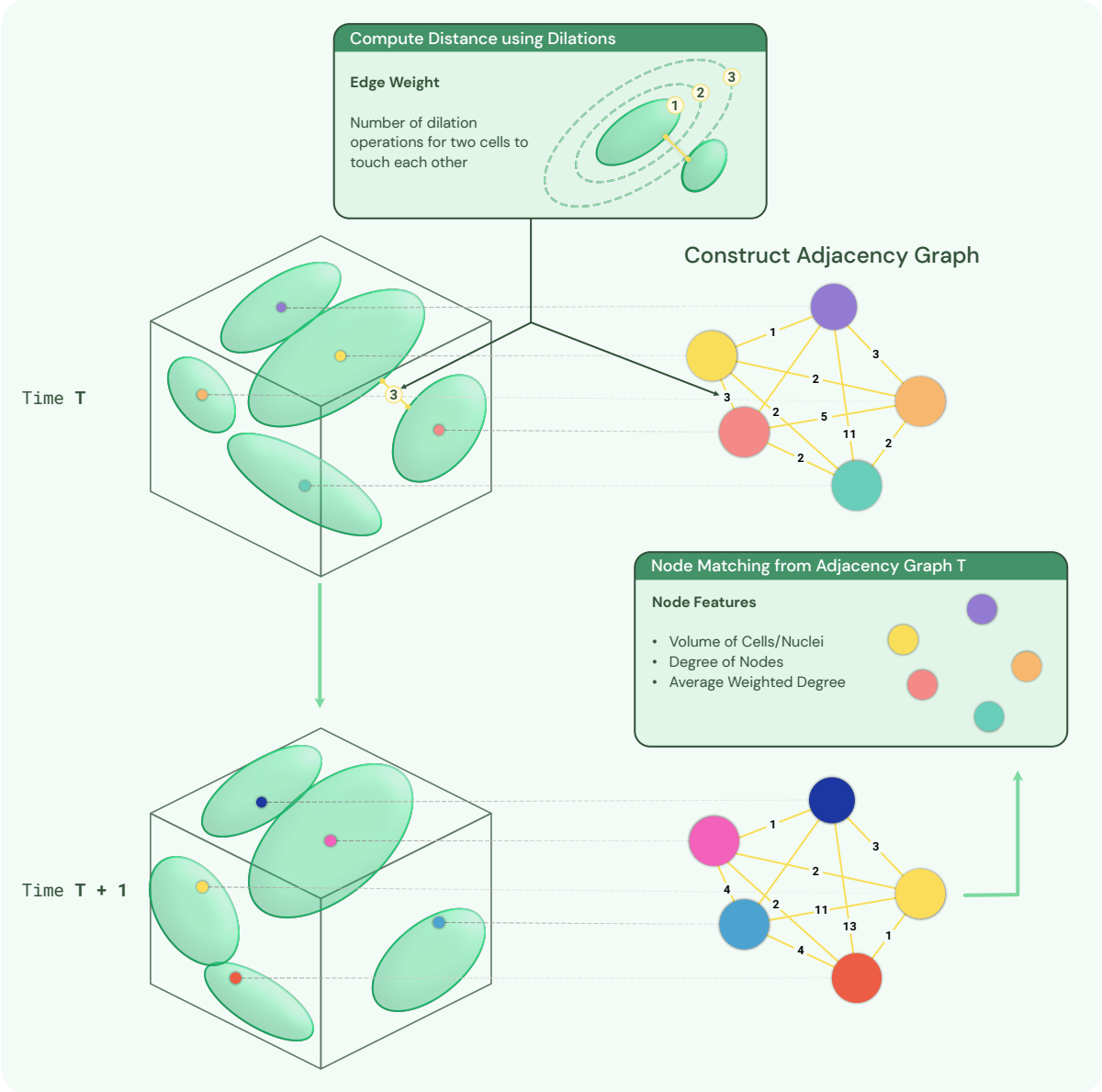

Figure 3.6: Constructing adjacency graph from the segmentation image and tracking cells/nuclei in consecutive frames using adjacency graph node features. Color of nodes denote the label/track of the cell/nuclei. Initially, random labels are assigned for each node in the adjacency graph. For T+1 frame, after node matching for time T, track IDs are assigned to each node in T+1.

3.2.2 Tracking and Feature Computation

After segmentation of 3D image stacks, the cells are detected and labeled in 3D space. Next, we utilize 3D spatial location of cells to build the adjacency graph for sub-cellular feature extraction and tracking as illustrated in Fig. 3.6.

Adjacency graph $G(V, E)$ is a weighted undirected graph. The vertex $v_i \in V$ represents the $i-$th cell. For each pair of vertices (v_i, v_j), there is an edge $e_i \in E$ connecting them. The weight $w_i \in W$ of the edge e_i is the distance between cell i and j. The distance between two cells is computed as the number of morphology dilation operations needed of cells i and j until cell i and j become a single connected component as illustrated in Fig. 3.7 shows. The details of this adjacency graph construction are given in Algorithm 1 below.

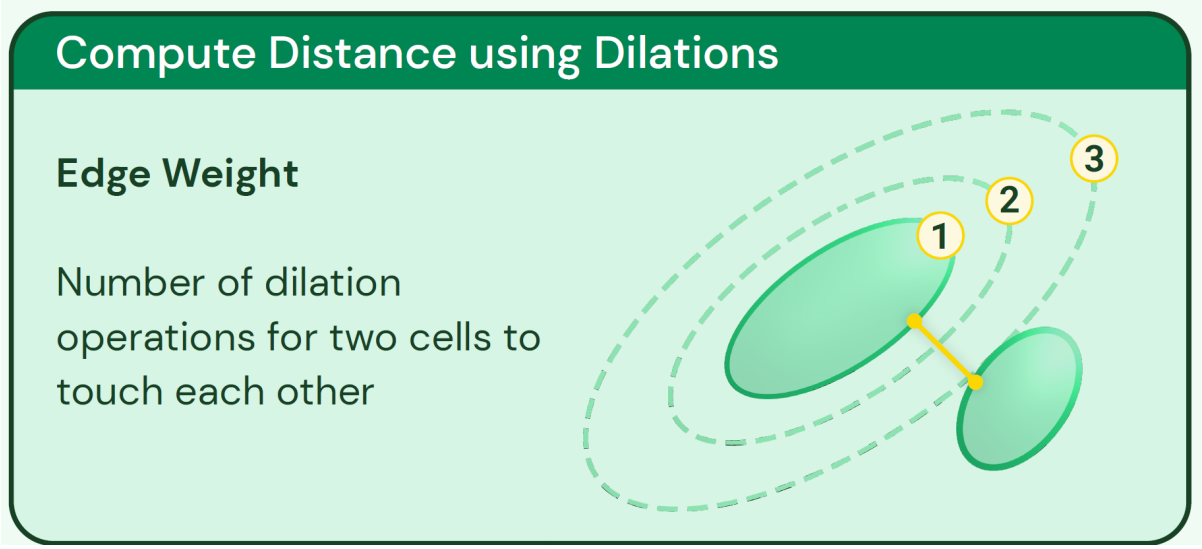

Figure 3.7: Illustration of how we compute distance between cells in adjacency graph.

Algorithm 1 Cell Adjacency Graph Construction

```
function CellAdjacency(Segmented Image Stack)
    Initiate AdjacentGraph
    Initiate drawboard (the same shape as segmentation)to be 0
    for i = 1 to number of cells do
        for j = 1 to number of cells do
            entry of drawboard is set to 1 if the voxel belong to cell i or j
            if i ≠ j then
                3D dilation of drawboard
                connected component analysis on drawboard
                if number of component is 1 then
                    append j to ith entry of AdjacentGraph
                end if
            end if
        end for
    end for
end function
```

Sub-cellular Feature Extraction Using Adjacency Graph

Sub-cellular feature extraction is based on the graph representation of the segmented image. To compute the anticlinal wall segments of cell i, we find all neighbor cells of cell i. The neighbor cells $\mathbf{N}_i$ are defined to be all cells that are at a distance 1 from cell i. The anticlinal wall segments is found by collecting all points in the segmentation image shared by cell i and cell j where cell $j \in \mathbf{N}_i$. To compute the junctions of 3 cell walls, we first pick cell $j \in \mathbf{N}_i$. Then the junctions of 3 cell walls is computed as the points in the segmentation image shared by cell i, cell j, and cell k where cell k is $\mathbf{N}_i \cap \mathbf{N}_j$.

Tracking Using Adjacency Graph

Cell/nuclei tracking is to reconstruct the lineage of cell/nuclei and match related cell/nuclei across the whole time sequence. The tracking process will give a trajectory for each individual nucleus as shown in Fig. 3.8. In the traditional Viterbi cell tracking algorithm, its complexity is $\mathcal{O}(TM^4)$ where T is the length of the video sequence and

M is the maximum number of nuclei. This is because the complexity of the Viterbi algorithm is linear in T, there can be M^2 pairs of nuclei in any two frames, and every such pair can have as many swap arcs between them as there are pre-existing tracks. A general assumption, as shown in [58], is that only certain nuclei events (apoptosis, division, etc) can happen and thus reduces possible swap arcs to some constant. This reduces the whole tracking complexity to be quadratic in the number of cells/nuclei. We propose an adjacency graph-based nuclei tracking algorithm utilizing nuclei relative location information to reduce the complexity to $\mathcal{O}(TM^2)$. As we explain below, our method explicitly takes into account cell division and cells that are not visible at any given time point, thus resulting in an overall improved performance.

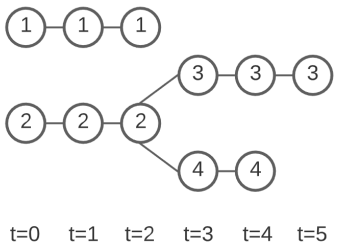

Figure 3.8: Cell/nucleus tracking illustration: number is used to represent the unique ID for each track of nucleus. At t=3, cell/nucleus 2 divides into 2 new cell/nuclei 3 and 4

To make the tracking method more efficient and accurate, we have made the assumption for the cell tracking that in consecutive image stacks, cells should have similar relative location. For this, we will focus on computing features $\mathbf{f}_{loc}$ that represent cell relative location information derived from the adjacency graph.

Cell location feature vector $\mathbf{f}_{loc}$ is a two dimensional vector (N, D), where N is the total number of neighbor cells and D is the average distance from all other cells. Consider the adjacency graph $G(V, E)$ of the segmented image stack. For node i in the graph, the

location feature vector can be expressed as:

$$\mathbf{f}_{loc}^{i} = (N, D) = (deg(v_i), \text{wdeg}(v_i)) \tag{3.6}$$

where $v_i \in V$, $deg(v_i)$ is the cardinality of N_i, and $\text{wdeg}(v_i)$ is the weighted degree of the vertex v_i. The weighted degree of the vertex v_i is defined as:

$$\text{wdeg}(v_i) = \frac{\sum_j w_{ij}}{\text{degree}(v_i)} \tag{3.7}$$

where $\text{degree}(v_i)$ represents the degree of the vertex v_i. Then we compute the cell size by counting number of voxels within the cell. Combining the cell location feature and cell size feature, we get the three dimensional feature vector $\mathbf{f}_{\text{track}}$. The details of algorithm used for calculating $\mathbf{f}_{\text{track}}$ is described in Algorithm 2:

Algorithm 2 Cell Tracking Feature Computation

```
function CellTrackFeature(Segmentation and G(V,E))
    Initiate FeatureVector
    for i = 1 to number of cells do
        cell size S_i = number of voxel inside cell i × voxel resolution
        wdeg(v_i) = (Σ_j w_ij) / deg(v_i)
        append (S_i, deg(v_i), wdeg(v_i)) to FeatureVector
    end for
end function
```

After computing $\mathbf{f}_{track}^{i}$ for all nodes in two consecutive frames, we link two nodes from different frames based on the following similarity measurement $sim(i, j)$ defined as

$$sim(i,j) = \frac{|S_{1i} - S_{2j}|}{S_{1i}} + \frac{|deg_1(v_i) - deg_2(v_j)|}{deg_1(v_i)} + \frac{|\text{wdeg}_1(v_i) - \text{wdeg}_2(v_j)|}{\text{wdeg}_1(v_i)} \tag{3.8}$$

where i and j denote two nodes from two consecutive frames. We define sim so that we

can allow different units of entries in $\mathbf{f}^i_{\text{track}}$. We find i^* and j^* that minimizes sim. i^* and j^* are linked only when their sim is below a set threshold value. If some cell/nucleus in the latter frame has no linked cell/nucleus in the previous frame, it means the new cell/nucleus comes into the field of view. Thus, based on this measure, we can track each cell/nucleus in consecutive frames of the recordings. The complexity for finding all possible links in consecutive frames is $\mathcal{O}(M^2)$. Therefore, the whole tracking complexity is $\mathcal{O}(TM^2)$. In our experiments, the threshold we use is between 0.1 to 0.5.

3.3 Datasets

There are three datasets used in this chapter (Dataset 1, Dataset 2, and Dataset 3). Details of each of these dataset are described in Chapter 2. We use different evaluation for different datasets based on these datasets' imaging subjects and annotations. Table 3.1 summarizes the datasets and their usage in this chapter.

Table 3.1: Datasets Summary and Usage (Note that TRA definition will be described in Results section)

Dataset	Source	Brief Description	Segmentation Evaluation	Sub-cellular Feature Extraction Evaluation	Tracking Evaluation
Dataset 1	Membrane-tagged confocal single layer pavement cells	5 time sequences, each sequence has 9-20 image stacks, and each stack with 18-30 slices	Cell count and cell shape as evaluation metrics	Sub-cellular feature extraction results are provided	Full annotation is unavailable, so TRA score is not provided
Dataset 2	Membrane-tagged confocal multi layer pavement cells	6 time sequences, each sequence has 20 image stacks, and each stack with 119-139 slices	Segmentation boundary evaluation metrics	Sub-cellular feature annotations are not available, so evaluation is not possible	TRA evaluation metric is provided
Dataset 3	Nuclei-tagged C.elegans dataset	4 time sequences, each sequence has 140-250 image stacks, and each stack with 31-35 slices	not applicable	not applicable	TRA score provided

3.4 Results

3.4.1 Cell Segmentation

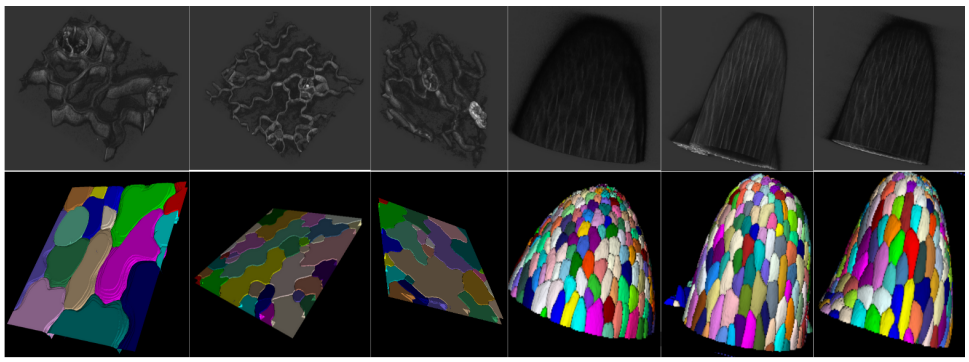

Figure 3.9: The figure shows three 3D segmentation image stacks. The top row is 3D view of confocal images, and bottom row is the 3D view of segmentation results. Left three samples are from Dataset 1 and right three samples are from Dataset 2

Since Dataset 1 does not have fully annotated 3D cell boundaries, we train all machine learning models using the entire Dataset 2. To evaluate different methods' performance on Dataset 2, we use the following train/test split. We randomly divide the whole datasets into three folds (train, validation, and test). To evaluate segmentation performance on test fold, we train models on all image stacks (volumes) of other two folds. We use above strategy three times for each layer and compute average and standard deviation with respect to evaluation metrics in Tables 3.3- 3.5.

We apply our proposed method to the Dataset 1 for the purpose of identifying and analyzing cells based on the segmentation. The segmentation results of our proposed method and other state-of-the-art methods are shown in Fig. 3.10. Our proposed method has visually better segmentation performance with closed cell surface and smooth boundary, and our method is able to identify the inter-cellular spaces and *protrusions* in the 3D cell image stack. For Dataset 1, we do not have full cell annotations, so we only evaluate the cell counting accuracy on this dataset.

Table 3.2: Cell Counting Accuracy for Different Methods. For each time sequence, there is a fixed number of cells. Due to segmentation error, the algorithms can generate different number of cells for different time points of the sequence.In the table, we showed average and standard deviation of number of detection cells for the whole sequence

Sequence	Ground truth	ACME [40]	MARS [41]	Supervoxel Method [42]	Our method
Sequence 1	23	21.5 ±3.2	25.5 ±2.2	24 ± 1.1	**23.5 ± 0.9**
Sequence 2	30	41.1 ±3.1	35.1 ±2.8	32 ± 2.1	**30.1 ± 0.8**
Sequence 3	25	22.6 ±2.1	27.5 ±3.2	24 ± 1.5	**25 ± 0.5**
Sequence 4	18	18.8 ±1.2	18.5 ±1.2	18.2 ± 1.2	**18 ± 0.6**
Sequence 5	28	31.5 ±2.9	24.5 ±2.3	26.2 ± 1.1	**27.8 ± 1**

For each sequence, there are a fixed number of cells for all time points. Therefore, we want segmentation algorithms to generate average cell counting results close to ground truth counting numbers, and the variance of counting results for one sequence should be as small as possible. Details of the cell counting results are in Table 3.2. Clearly, our method has the best cell counting performance.

In order to verify if the output of the segmentation can be used for time lapse sequence analysis, we calculate basic cell shape information from the maximum area plane of the cells to compare with the expert annotations. The maximum area plane of a cell is the image plane which has the largest cell area across all z-slices. The shape information includes area, perimeter, circularity, and solidarity. Fig. 3.11 shows the comparison. Note that not all cells are annotated so that some cell comparisons are missed. The average shape difference is 4.5 percent and the largest shape difference is within 10 percent.

Next, we apply our cell segmentation method on Dataset 2. Boundary precision, recall, and F1 score are used to evaluate the boundary segmentation accuracy. Specifically, given a ground truth boundary image G and a computed boundary image B, we can define the following measurements:

- True Positives (TP): Number of boundary pixels in G for which exist a boundary pixel in B in range R.

- False Negatives (FN): Number of boundary pixels in G for which does not exist a

boundary pixel in B in range R.

- False Positives (FP): Number of boundary pixels in B for whose does not exist a boundary pixel in G in range R

Then boundary precision is defined to be: $\frac{TP}{(TP+FP)}$ and recall: $\frac{TP}{TP+FN}$ In our experiment, we set R to be 5. Tables 3.3 to 3.5 shows the comparison of the final segmentation boundary result using our proposed method and other methods including ACME [40], MARS [41] and a supervoxel-based algorithm [42] on L_1 to L_3 respectively. In terms of cell wall accuracy, our model shows at least 0.03 improvement in the F-score measure on average in terms of cell wall segmentation accuracy.

It is noted that the average segmentation time of our proposed model is significantly shorter compared to the supervoxel-based method [42]. Our proposed method takes approximately 0.8 seconds to segment one 512×512 image slice on average, whereas supervoxel-based method takes approximately 6 seconds on a NVIDIA GTX Titan X with an Intel Xeon CPU E5-2696 v4 @ 2.20GHz. We have also integrated the proposed segmentation method into BisQue. There are three hyperparameters in the BisQue segmentation module. "Minimum Distance"is σ_γ in eq. 3.5. "Label Threshold"relates to the variation in the cell volumes within the datasets. This is used to ensure small regions such as protrusions are not labeled as cells. Intensity values below "Threshold"are ignored. "Threshold"is typically between 0 and 0.1 for a normalized image.

3.4.2 Sub-cellular Feature Extraction

We apply our whole workflow on Dataset 1 to extract sub-cellular features like anticlinal wall segments and junctions of 3 cell walls. Qualitative results of the extracted sub-cellular features are shown in Fig. 3.12.

The quantitative measurement of accuracy of junctions of 3 cell walls is also provided.

Table 3.3: 3D Segmentation Performance on L_1. If there is a detected boundary voxel by algorithms within 5 voxels of a ground truth boundary voxel, then it is a correct detection. If there is not any detected boundary voxel by algorithms within 5 voxels of a ground truth boundary voxel, then it is a miss detection. If there is a detected boundary voxel by algorithms within 5 voxels of a voxel that is not ground truth boundary voxel, then it is a miss detection. Same evaluation metric for L_2 and L_3. Numbers in brackets are standard deviations. Note that ACME and MARS are not based on machine learning methods so we just provide the average number. Supervoxel method follows the same training and testing procedure as that of our method.

Algorithm	Precision	Recall	F1
ACME [40]	0.805	0.966	0.878
MARS [41]	0.910	0.889	0.899
Supervoxel method [42]	**0.962** (0.031)	0.932 (0.038)	0.947 (0.033)
our method	0.961 (0.010)	**0.973** (0.013)	**0.967** (0.012)

Table 3.4: 3D Segmentation Performance on L_2

Algorithm	Precision	Recall	F1
ACME [40]	0.775	**0.980**	0.866
MARS [41]	0.921	0.879	0.900
Supervoxel method [42]	0.910 (0.049)	0.932 (0.051)	0.921 (0.044)
our method	**0.955** (0.012)	0.971 (0.011)	**0.963** (0.012)

Table 3.5: 3D Segmentation Performance on L_3

Algorithm	Precision	Recall	F1
ACME [40]	0.745	**0.976**	0.845
MARS [41]	0.909	0.879	0.894
Supervoxel method [42]	**0.982** (0.055)	0.881 (0.053)	0.929 (0.052)
our method	0.955 (0.011)	0.942 (0.018)	**0.949** (0.013)

We compare our results with 3D corner detection based method [4] on the raw image stack, and applying our 3 cell wall junction detection method using the segmentation image from other state-of-the-art methods [40–42]. The 3 cell wall junction detection results are shown in Table 3.6. If 3 cell wall junctions are detected within 5 voxels of a ground truth 3 cell wall junction, it is a correct detection. Then we define false positive (FP), and false negative (FN) based on the binary detection of 3 cell walls junction. Specifically, FP is the number of computed 3 cell wall junctions whose does not exist a ground truth 3 cell junction in range 5 voxels and FN is the number of ground truth 3 cell wall junctions whose does not exist a computed 3 cell junction in range 5 voxels. Then we define precision, recall, and F1 based on true positive, FP, and FN, The error (E) is defined by the summation of FP and FN and normalized by total number of true 3 cell

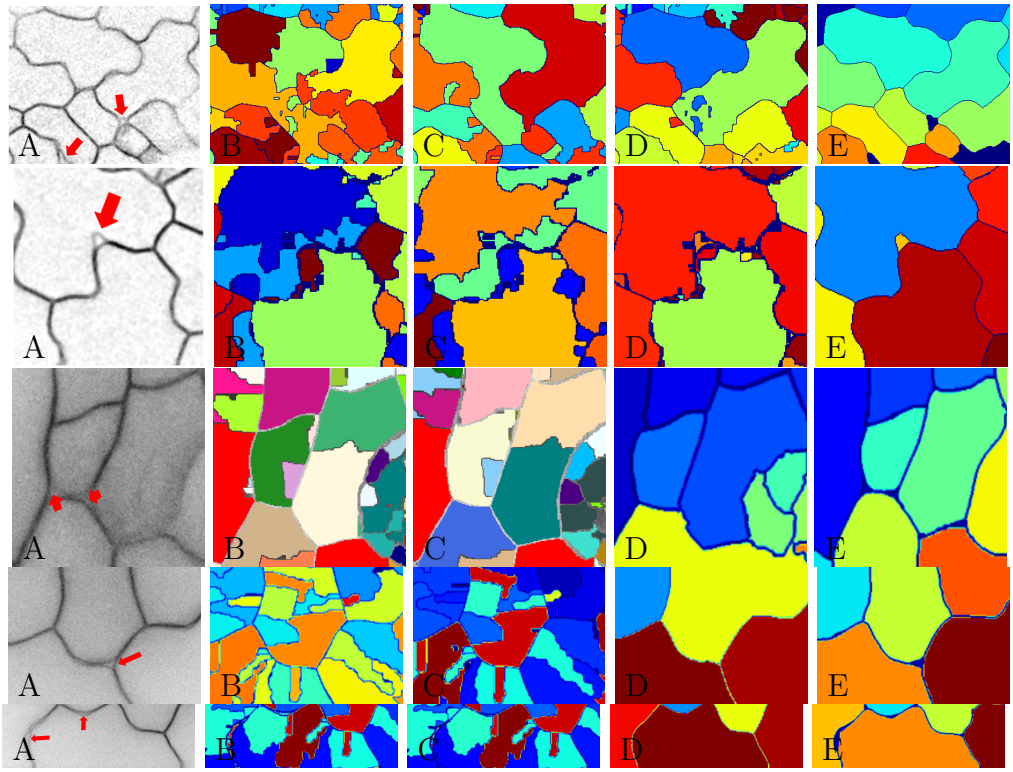

Figure 3.10: The figure shows the segmentation results of the cell image with inter-cellular space or *protrusion* indicated by a red arrow. (A) Inverted raw image in xy orientation, (B) MARS, (C) ACME, (D) supervoxel-based method, (E) proposed method.

wall junctions. The results in the Table 3.6 are average values across all image stacks. From the table, we can see our method has the best 3 cell wall junction detection accuracy in terms of F1. Compared to the method that directly computes 3 cell wall junctions from raw image, our method has significantly better performance in terms of FP. This is because not all corner points are junctions of three cell walls. For example, corner detection based method gives false positive in the case shown in Fig. 3.13. Our graph based method for detecting 3-way junctions uses both the boundary voxel information and the cell labels, including intercellular spaces, thus minimizing false positives.

The anticlinal wall segment is defined by two neighboring junctions of 3 cell walls are also computed. The partial annotation of such segments are provided. We would like

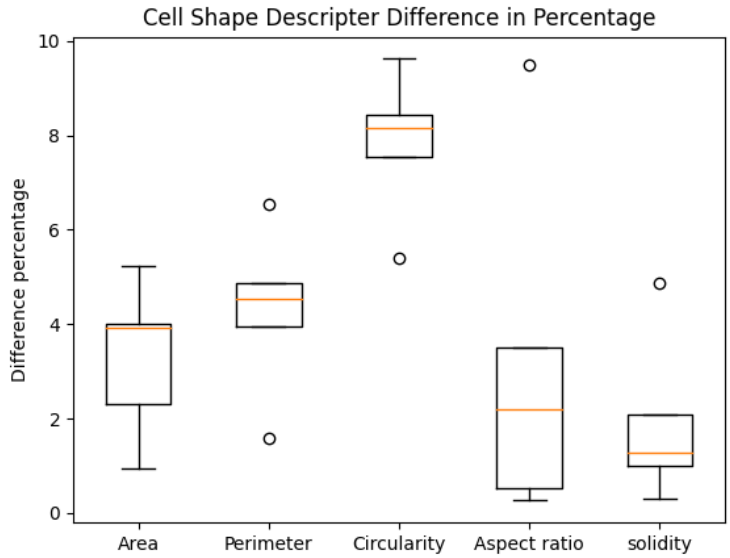

Figure 3.11: 3D segmentation evaluation using cell shape descripter including area, perimeter, circularity, aspect ratio, and solidity (ratio between cell area and its convex hull area). The difference is in terms of percentage.

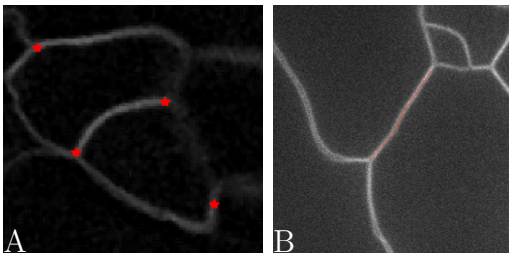

Figure 3.12: A: Extracted junctions of three cell walls, B: Extracted anticlinal wall segment.

to note that such manual annotations are very labor intensive and it is impractical to annotate all anticlinal cell wall segments (see Fig. 3.2) even in a single 3D volume. The practical difficulties include lack of support for 3D visualization and annotation tools for tracing. The ground truth segments were annotated by going through each slice in the image stack, finding the approximate slice where neighboring cell walls touch, and then tracing the segment in that single slice. Each segment in the ground truth is represented by a collection of coordinates of the segment in that image slice. Note that different

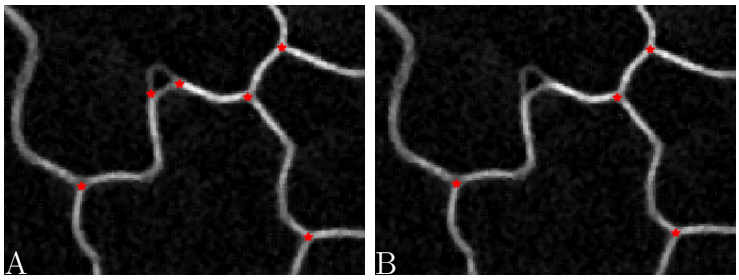

Figure 3.13: Example of computing 3 cell-wall junctions. (A) using method proposed in [4], (B) using our method, Note that (A) has several false positives.

Table 3.6: Quantitative Analysis on Error of Junctions of Three Cell Walls. Precision, recall, and F1 score are used to evaluate the detection of those junctions

Algorithm	Precision	Recall	F1
Corner Detection [4]	0.893	0.962	0.926
ACME [40]	0.829	0.964	0.891
MARS [41]	0.823	**0.980**	0.895
Supervoxel method [42]	0.933	0.911	0.922
Our Method	**0.980**	0.945	**0.962**

segments can be on different slices. In contrast, each of our computed segments can span multiple Z slices, hence providing a more accurate 3D representation than is manually feasible. This also makes it challenging to compare the manual ground truth with the computed results. Fig. 3.14 shows our computed segments compared with the annotated segments.

3.4.3 Evaluation metrics for anticlinal wall segments

We propose a set of evaluation metrics for the detected anticlinal wall segments as there are no prior works on this topic.

1. **End-point Displacement Error (EDE)** in the end points of the two segments. Given two segments P with m points and Q with n points, two end points of P are

p_1 and p_m and two end points of Q are q_1 and q_n. EDE is defined as

$$EDE(P,Q) = \frac{1}{2}(\|p_1 - p_m\| + \|q_1 - q_n\|) \tag{3.9}$$

where $\|\cdot\|$ is l^2 norm.

2. **Fréchet Distance (FD)** [59] between the two segments. Fig. 3.15 illustrates the definition of FD. FD is a measure of shape similarity of two curves and it takes into account the location and ordering of points along the curves. Mathematically, consider two curves P with m points and Q with n points. P contains a sequence of points $(p_1, ..., p_m)$ and Q contains a sequence of points $(q_1, ..., q_n)$. A coupling L between P and Q is a sequence $(p_{a_1}, q_{b_1}), (p_{a_2}, q_{b_2}), ..., (p_{a_z}, q_{b_z})$ of distinct pairs from P and Q such that $a_1 = 1$, $b_1 = 1$, $a_z = m$, and $b_z = n$, and for all $i = 1, ..., z-1$ we have $a_{i+1} = a_i$ or $a_{i+1} = a_i + 1$, and $b_{i+1} = b_i$ or $b_{i+1} = b_i + 1$. Thus the order of those points are kept in the coupling L. The coupling length $\|L\|$ is the length of the longest Euclidean distance in L:

$$\|L\| = \max_{i=1,...,z} d(p_{a_i}, q_{b_i}) \tag{3.10}$$

where d is the Euclidean distance. Then FD F is defined as:

$$F(P,Q) = \min\{\|L\|\} \tag{3.11}$$

where L is a coupling of P and Q.

3. Length Difference (LD), absolute difference in lengths between the two segments

4. Percentage Difference in length (DP), length difference normalized by ground truth length.

Table 3.7: Anticlinal cell wall segment evaluation on Dataset 1 using EDE, FD, LD, DP

Sequence	EDE	FD	LD	DP
Sequence 1	2.80	4.00	2.32	2.90
Sequence 2	6.40	8.40	4.19	6.10
Sequence 3	3.05	3.9	1.74	2.4
Sequence 4	3.02	3.70	2.24	2.3
Sequence 5	2.33	3.40	2.11	2.10

Average EDE between the results using our method and the ground truth is 3.03 voxels, average FD is 3.7 voxels, average LD is 2.24 voxels, and average DP is 2.3 percent. Evaluation results of different time series are shown in Table 3.7 and evaluation result for each segment is in the supplemental materials.

We also apply our tracking method on Dataset 2 and Dataset 3. Table 3.8 shows the quantitative comparison of our method with other state-of-the-art cell/nuclei tracking methods. The evaluation metric we use is tracking accuracy (TRA), proposed in [12]. TRA measures how accurately each cell/nuclei is identified and followed in successive image stacks of the sequence. Ground truth tracking results and tracking results generated from algorithms are viewed as two acyclic oriented graphs and TRA measures the number of operations needed to modify one graph to another. More specifically, TRA is defined on Acyclic Oriented Graph Matching (AOGM) as

$$\mathrm{TRA} = 1 - \frac{\min(\mathrm{AOGM}, \mathrm{AOGM}_0)}{\mathrm{AOGM}_0} \tag{3.12}$$

where AOGM_0 is the AOGM value required for creating the reference graph from scratch. TRA ranges between 0 to 1 (1 means perfect tracking). Our method shows a rough 0.05 TRA measurement improvement on Dataset 2. To demonstrate the robustness of our tracking method, we also apply it on Dataset 3, a cell nuclei dataset, and achieve a TRA of 0.895 which is comparable to state-of-the-art tracking methods on IEEE ISBI CTC2020 cell tracking challenge. State-of-the-art methods ([60] and [58]) are based on

the traditional Viterbi cell tracking algorithm whose complexity is $\mathcal{O}(TM^4)$ where T is the length of the sequence and M is the maximum number of cells/nuclei. This is because the complexity of the Viterbi algorithm is linear in T, there can be M^2 pairs of nuclei in any two frames, and every such pair can have as many swap arcs between them as there are pre-existing tracks. A general assumption, as shown in [58], is that only certain cell/nuclei events (apoptosis, division, etc) can happen and thus reduces possible swap arcs to some constant. This can reduce the whole tracking complexity to be quadratic in the number of nuclei. Specifically, for our proposed tracking method, the complexity is $\mathcal{O}(TM^2)$. Sequence 1 and 2 are the training data released from the challenge and we run the state-of-the-art methods on the individual sequence to get TRA evaluation metric. Sequence 3 and 4 are testing data that is not published by the challenge and TRA values are given by the challenge organization.

Table 3.8: Cell Tracking Performance on Dataset 2 and Dataset 3

Dataset 2	Viterbi Tracker [48]	Cell Proposal [61]	Our method
Sequence 1	0.513	0.492	**0.571**
Sequence 2	0.520	0.512	**0.593**
Sequence 3	0.488	0.532	**0.581**
Sequence 4	0.533	0.498	**0.566**
Sequence 5	0.542	0.525	**0.602**
Sequence 6	0.518	0.542	**0.544**
Dataset 3	KIT-Sch-GE [60]	KTH-SE [58]	Our method
Sequence 1	0.903	**0.942**	0.931
Sequence 2	0.906	0.893	**0.912**
Sequence 3 and 4	0.886	**0.945**	0.895

In summary, we do extensive experiments and use different evaluation metrics to demonstrate the performance of our method. For segmentation, we use cell counting accuracy in Table 3.2, cell shape evaluation metric Fig. 3.11, and cell boundary segmentation accuracy in Tables 3.3-3.5 to show the performance of our method. For sub-cellular feature extraction, we use precision, recall, and F1 score as in Table 3.6 to evaluate 3

cell wall junctions detection performance, and we use EDE, FD, LD, and DP in Table 3.7 to evaluate the segments detection performance. For tracking, we use TRA in Table 3.8 as the evaluation metric.

3.5 Summary

In this chapter, we present an end-to-end workflow for extracting quantitative information from 3D time-lapse imagery. The workflow includes 3D segmentation, tracking, and sub-cellular feature extraction. The 3D segmentation pipeline utilizes deep learning models with rotation equivariance. Then an adjacency graph is built for cell tracking and sub-cellular feature extraction. We demonstrate the performance of our model on multiple cell/nuclei datasets. In addition, we also curate a new pavement cell dataset with partial expert annotations that will be made available to researchers.

3.6 CELLECT2.0

The proposed segmentation method is implemented as a computational module in BisQue [34, 62]. We first dockerize the method and then put it as a module on BisQue. Users can run the CellECT2.0 module using the following steps: (1) Navigate to BisQue on their web browser and create an account, (2) Upload their own data in TIFF format or use suggested example dataset, (3) Select an uploaded TIFF image or use our example, (4) Set hyper parameters of the module (default value to run Dataset 1) and select `Run` and the BisQue service will compute the segmentation results and display it in the browser. The runtime for a $512 \times 512 \times 18$ image is approximately one minute using a CPU node with a 24 core Xeon processor and 128GB of RAM. The module also supports GPU process.

We have also enabled the better visualization function in BisQue. It is useful to visualize 3D segmentation mask especially for instance segmentation mask. The segmentation mask is overlayed on the original image and users can toggle on/off for some instances in the segmentation mask as Fig. 3.16 shows. The segmentation mask can also be viewed in 3D as shown in Fig. 3.17.

Cellular and sub-cellular features are also provided on Bisque by HDF files. Those features provided by BisQue are summarized in Table 3.9.

Table 3.9: Cellular and Sub-Cellular Features Provided by BisQue

Different features	**Example Values or Explanation**
Cell Volume	number of voxels inside a cell, for example, 358,445
Neighboring (Adjacent) cells	for example, cells 1,3,11,14 are neighboring cells of cell 2
3D cell surface	list of coordinates of surface of points for one cell
Three cell wall junction points	for example, junction point of cell 2,3,11 is (12,207)
Cell Center	for example, center of cell 2 is (8,61,8)
Segments	list of coordinates of points along that segment

The full cellular and sub-cellular results for this image stack can be accessed here.

3.7 Segments evaluation

Some examples of segments evaluation are summarized in Table 3.7. The evaluation metrics include: Euclidean error of start (end) points of segments, Fréchet Distance, and length difference of segments. In Table 3.7, the first column is segment ID. Segment ID includes two parts: Sequence ID and the segment ID within the sequence. Sequence ID starts with "LGMTPM" and ends with a 2-digit number to differentiate different sequences. Segment ID contains information about which frame this segment comes from. For example, "LGMTPM01 Seg03T28" means this segment comes from frame 28 of sequence 01.

Table 3.10: Segments evaluation results. The results include segments' end points location accuracy, segments length accuracy, and Frechet Distance for segments shape accuracy

	detected start point x	detected start point y	detected start point z	detected end point x	detected end point y	detected end point z	GT start point x	GT start point y	GT end point x	GT end point y	euclidean distance error start point	euclidean distance error end point	Frechet Distance	GT Length	Detected Length	difference	% change
LGMTPM01 Seg03T28	391	215	3	485	203	3	392.7	216.2	481.8	203.3	2.0	3.2	3.9	95.1	93.4	1.7	1.8
LGMTPM01 Seg06T24	121	144	4	195	142	5	124.9	145.3	196.0	139.0	4.1	3.2	4.1	85.6	87.2	1.6	1.9
LGMTPM01 Seg02T01	169	356	2	234	360	4	172.8	358.3	233.3	360.3	4.5	0.7	4.5	66.6	70.3	3.6	5.4
LGMTPM01 Seg05T01	303	274	6	350	320	2	303.5	273.3	346.8	320.0	0.9	3.3	3.3	70.7	72.9	2.2	3.1
LGMTPM01 Seg08T01	200	59	1	299	82	2	204.3	59.8	297.3	81.8	4.3	1.8	4.3	100.6	98.1	2.4	2.4
LGMTPM02 Seg05T23	207	138	4	217	77	5	210.2	143.8	213.3	82.0	6.6	6.2	8.4	68.9	73.1	4.2	6.1
LGMTPM03 Seg02T32	178	212	4	246	183	4	182.7	212.3	243.7	182.7	4.7	2.4	4.1	77.8	80.3	2.5	3.2
LGMTPM04 Seg03T17	286	327	5	328	392	7	289.7	329.3	325.2	395.7	4.3	4.6	4.7	85.2	83.3	2.0	2.3
LGMTPM05 Seg03T01	45	220	3	145	224	5	48.8	219.3	145.3	220.3	3.8	3.8	3.7	106.5	108.1	1.6	1.5
LGMTPM05 Seg03T02	11	198	2	100	200	4	13.5	196.5	101.5	198.0	2.9	2.5	3.0	98.0	100.0	2.0	2.0
LGMTPM05 Seg03T01	12	201	3	100	200	5	13.0	202.5	102.3	203.5	1.8	4.2	2.9	99.3	97.0	2.2	2.3
LGMTPM05 Seg03T03	13	200	10	103	200	8	14.0	199.8	102.5	201.0	1.0	1.1	3.6	98.5	102.1	3.6	3.7
LGMTPM05 Seg03T04	17	208	9	102	211	7	15.5	205.5	104.5	206.3	2.9	5.4	3.7	99.0	103.1	4.1	4.1
LGMTPM05 Seg03T05	16	210	8	105	212	8	15.8	209.3	105.3	210.3	0.8	1.8	3.5	99.5	101.2	1.7	1.7
LGMTPM05 Seg03T06	18	184	6	105	185	8	17.0	182.5	106.8	184.0	1.8	2.0	3.2	95.8	98.1	2.3	2.4
LGMTPM05 Seg03T07	19	216	7	104	218	9	17.0	215.8	106.8	216.5	2.0	3.1	3.3	99.8	101.2	1.5	1.5
LGMTPM05 Seg03T08	18	221	6	105	221	8	17.0	220.0	106.5	220.8	1.4	1.5	3.6	99.6	102.3	2.7	2.7
LGMTPM05 Seg03T09	18	200	6	110	201	7	19.3	199.0	109.5	199.8	1.6	1.3	3.1	100.3	101.3	1.0	1.0
LGMTPM05 Seg03T10	24	192	5	110	195	7	22.0	193.5	112.3	194.0	2.5	2.5	3.3	100.3	102.3	2.1	2.0
LGMTPM05 Seg03T11	23	197	6	115	200	9	22.0	196.9	113.3	197.8	1.0	2.9	3.4	101.3	101.2	0.1	0.1
LGMTPM05 Seg03T12	25	200	11	117	199	9	25.0	198.5	115.3	198.3	1.5	1.8	3.3	100.3	101.4	1.0	1.0
LGMTPM05 Seg03T13	25	201	10	114	198	8	24.3	197.3	116.3	200.5	3.8	3.4	3.2	102.1	104.3	2.2	2.2
LGMTPM05 Seg03T14	27	200	9	118	200	7	25.5	198.5	118.0	200.3	2.1	0.3	3.6	102.5	103.3	0.7	0.7
LGMTPM05 Seg03T15	26	199	7	119	202	7	29.0	197.8	122.5	198.3	3.3	5.1	3.2	103.5	101.5	2.0	2.0
LGMTPM05 Seg03T16	29	195	11	122	199	12	31.3	196.3	125.0	197.8	2.6	3.3	3.4	103.8	100.3	3.5	3.3
LGMTPM05 Seg03T17	32	220	10	126	220	11	33.8	221.0	127.0	222.8	2.0	2.9	3.5	103.3	101.3	2.0	1.9
LGMTPM05 Seg03T18	16	230	9	113	230	10	17.0	229.3	111.8	230.8	1.3	1.5	3.7	104.8	106.4	1.6	1.5
LGMTPM05 Seg03T19	35	231	8	130	232	9	34.8	230.0	129.3	230.8	1.0	1.5	3.6	104.5	107.9	3.4	3.2
LGMTPM05 Seg03T20	37	220	9	130	224	10	37.5	221.3	132.3	222.3	1.3	2.9	3.9	104.8	106.8	2.0	1.9
LGMTPM05 Seg03T21	31	226	8	129	230	8	30.5	225.5	126.5	227.0	0.7	3.9	3.1	106.0	101.3	4.7	4.4
LGMTPM05 Seg03T22	47	220	9	144	223	10	48.8	219.3	145.3	220.3	1.9	3.0	4.0	106.5	105.9	0.6	0.6
LGMTPM04 Seg01T01	365	360	12	433	425	13	366.0	358.0	429.3	424.0	2.2	3.9	3.1	101.4	99.3	2.2	2.1
LGMTPM04 Seg01T02	366	362	13	430	427	14	365.8	362.5	429.0	429.5	0.6	2.7	3.2	102.1	104.0	1.8	1.8
LGMTPM04 Seg01T03	365	370	12	428	436	12	368.0	371.0	431.3	438.0	3.2	3.8	3.2	102.1	104.7	2.5	2.5
LGMTPM04 Seg01T04	367	378	11	429	449	11	367.3	379.3	431.5	445.3	1.3	4.5	4.0	102.1	101.2	0.9	0.9
LGMTPM04 Seg01T05	365	381	12	435	448	13	367.8	383.3	431.3	450.5	3.6	4.5	4.2	103.5	100.9	2.6	2.5
LGMTPM04 Seg01T06	363	375	14	429	453	15	368.3	380.0	433.0	450.0	7.3	5.0	3.6	105.4	102.3	3.0	2.9
LGMTPM04 Seg01T07	369	385	11	433	450	12	369.3	385.5	430.8	450.3	0.6	2.3	3.4	99.3	98.5	0.8	0.8
LGMTPM04 Seg01T08	370	381	11	432	455	11	370.3	385.3	432.8	451.1	4.3	4.0	3.1	100.7	103.7	3.0	3.0
LGMTPM04 Seg01T09	371	388	10	431	452	10	371.0	389.0	433.0	450.0	1.0	2.8	3.9	98.9	96.2	2.6	2.7
LGMTPM04 Seg01T10	373	389	9	433	450	9	375.0	389.3	430.3	451.0	2.0	2.9	3.2	92.8	95.0	2.2	2.4
LGMTPM04 Seg01T11	378	389	11	433	451	9	376.3	389.3	430.3	452.3	1.8	3.1	3.6	93.0	90.0	3.0	3.2
LGMTPM04 Seg01T12	375	389	9	432	455	9	377.8	389.0	430.3	451.0	2.8	4.4	4.9	91.2	96.1	4.9	5.3
LGMTPM04 Seg01T13	366	389	8	443	460	9	370.3	391.3	440.8	458.3	4.9	2.9	4.6	107.1	103.2	3.9	3.6
LGMTPM04 Seg01T14	370	390	10	439	454	10	372.5	392.3	438.5	451.0	3.4	3.0	4.8	98.3	96.1	2.2	2.2
LGMTPM04 Seg01T15	373	386	9	438	453	9	374.3	387.5	436.0	451.0	2.0	2.8	3.2	98.6	101.0	2.4	2.5
LGMTPM04 Seg01T16	372	391	9	430	452	11	375.3	389.5	430.3	451.0	3.7	1.0	3.3	92.5	93.9	1.4	1.5
LGMTPM04 Seg01T17	373	394	9	436	452	9	375.0	392.3	435.0	451.0	2.7	1.4	3.2	94.0	95.2	1.3	1.4
LGMTPM04 Seg01T18	378	389	9	435	450	9	375.3	386.0	430.3	451.0	4.1	4.9	3.5	95.2	92.0	3.1	3.3
LGMTPM04 Seg01T19	379	389	9	437	455	9	377.5	385.8	435.3	451.0	3.6	4.4	3.7	95.1	98.1	3.0	3.1
LGMTPM04 Seg01T20	378	389	9	438	453	9	377.9	388.0	436.3	451.0	1.0	2.7	3.9	95.9	96.1	0.2	0.2
LGMTPM04 Seg01T21	377	389	9	434	452	9	374.3	389.0	431.0	451.0	2.7	3.2	4.1	94.0	92.2	1.8	1.9
LGMTPM04 Seg01T22	370	387	9	439	448	9	369.3	386.0	440.3	451.0	1.3	3.3	5.1	106.3	102.0	4.3	4.0
LGMTPM04 Seg01T23	376	389	9	433	449	9	375.1	388.3	430.3	451.0	1.2	3.4	3.2	93.3	92.6	0.7	0.8
LGMTPM04 Seg01T24	372	389	9	433	449	9	375.3	389.3	430.3	451.0	3.3	3.4	4.0	92.6	92.7	0.1	0.1
LGMTPM03 Seg01T01	210	371	8	243	417	10	212.5	369.7	243.0	419.5	2.8	2.5	4.2	72.4	73.3	0.8	1.2
LGMTPM03 Seg01T02	215	372	9	245	419	11	213.3	370.0	246.5	421.5	2.6	2.9	3.9	71.3	73.6	2.3	3.2
LGMTPM03 Seg01T03	216	375	9	249	420	10	214.5	371.3	248.0	422.8	4.0	2.9	3.6	71.3	70.0	1.3	1.9

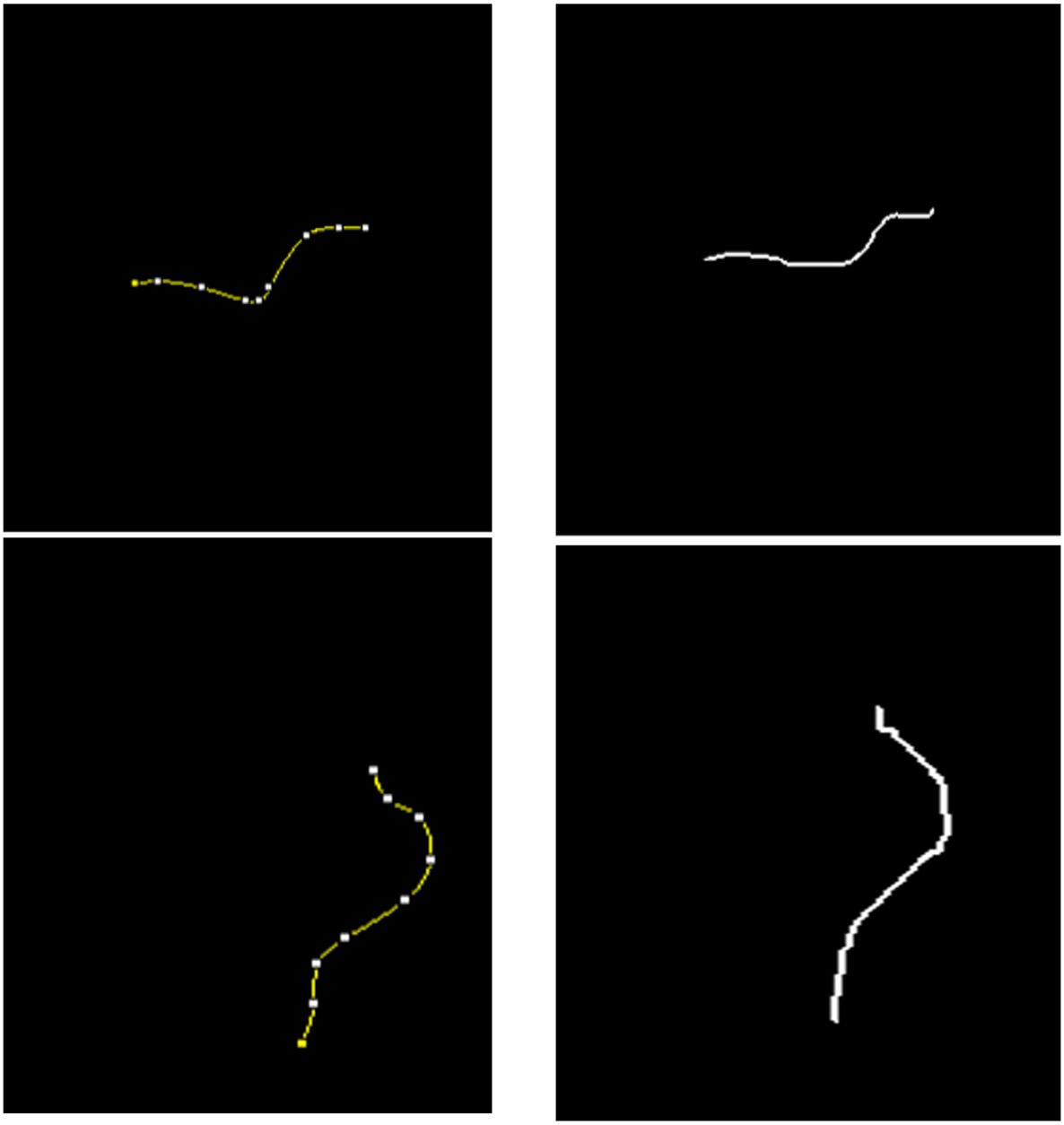

Figure 3.14: Expert annotated segments (Left) and our computed segments (Right)

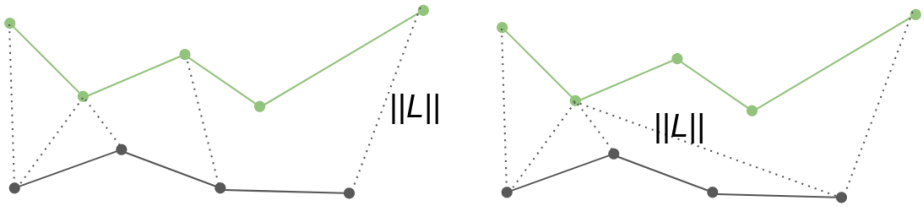

Figure 3.15: The figure shows two examples of coupling L. Dashed lines represent distinct pairs. $\|L\|$ is the length of the longest distance of those pairs. Finally, FD is the minimum of those $\|L\|$.

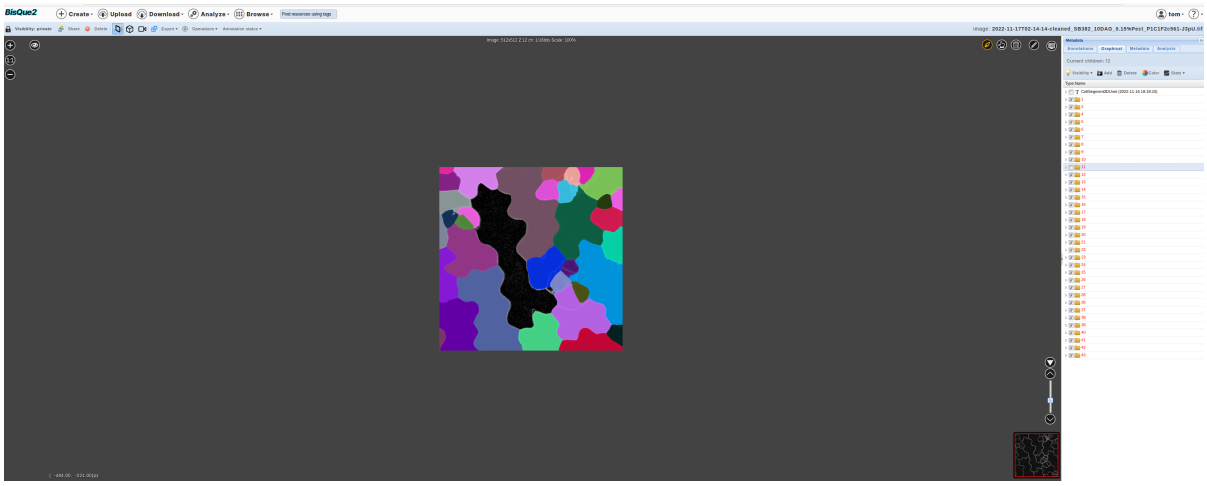

Figure 3.16: 3D segmentation visualization in BisQue. Segmentation mask is overlayed on the original image and users can toggle segmentation mask on/off.

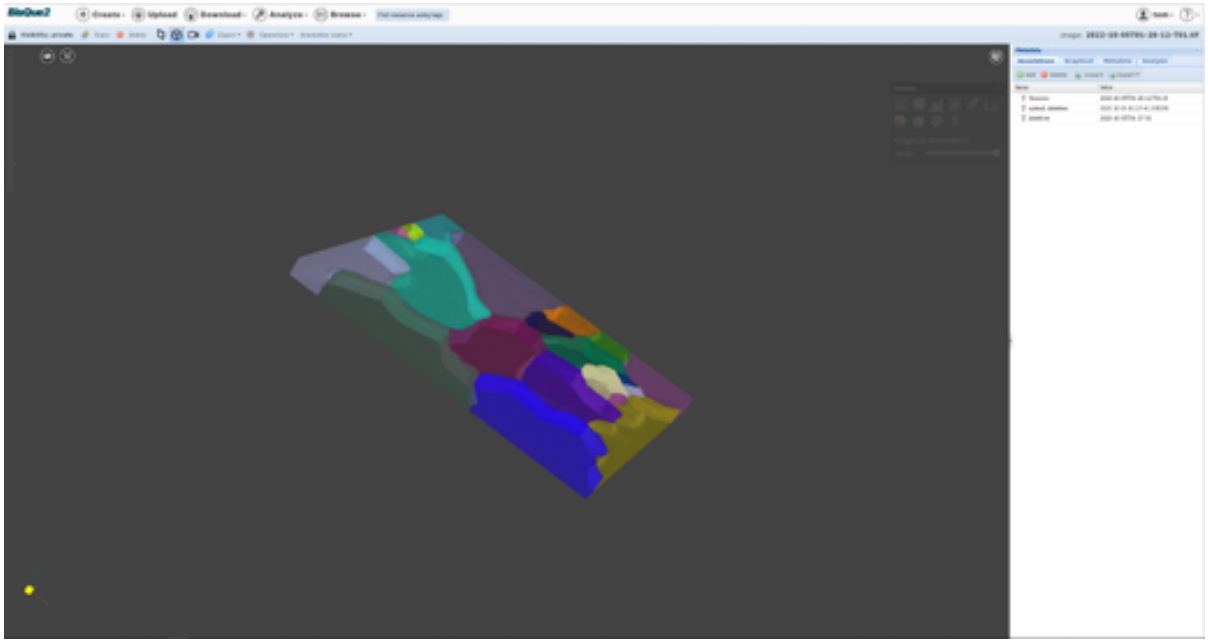

Figure 3.17: 3D visualization in BisQue from our segmentation method. Here a 3D segmentation mask is presented.

Chapter 4

Neuron Morphology Analysis

> Many of life's failures are people
> who did not realize how close they
> were to success when they gave up.
>
> *Thomas A. Edison*

This chapter focuses on 3D neuron shape analysis. We consider the problem of finding an accurate representation of neuron shapes, extracting sub-cellular features, and classifying neurons based on neuron shapes. In neuroscience research, the skeleton representation is often used as the compact and abstract representation of neuron shapes. However, existing methods are limited to getting and analyzing "curve"skeletons which can only be applied for tubular shapes. This chapter presents a 3D neuron morphology analysis method for more general and complex neuron shapes. First, we introduce the concept of skeleton mesh which is used to represent general neuron shapes and propose a novel method for computing mesh representations from 3D surface point clouds. A skeleton graph is then obtained from skeleton mesh and is used to extract sub-cellular features. Finally, an unsupervised learning method is used to embed the skeleton graph

for neuron classification. Extensive experimental results are provided that demonstrate the robustness of our method to analyze neuron morphology.

4.1 Introduction

The importance of neuronal morphology has been recognized from the early days of neuroscience [19,63]. There are three obstacles in automatic neuron morphology analysis. First, we need to have a good shape representation of each neuron. Skeleton representations are widely used in neuroscience [21–24, 64] as they provide a compact and abstract shape representation. Mathematically, skeletonization or medial axis transform (MAT) has a rigorous definition for arbitrary shapes. The skeleton of a shape is defined as a collection of interior points that have at least two closest points on the surface of the shape. We refer to those interior points as skeleton points and each skeleton points are associated with a radius. Fig. 4.1 shows an example of MAT. However, in reality, it is not an easy task to get skeleton representation directly from images. Most automatic segmentation or manual segmentation outputs surface point clouds of neurons. Thus, we need to compute the 3D neuron skeleton from 3D surface point clouds. The skeleton representation further enables computing sub-cellular features such as length and number of branches of neurons as well as classification of neurons.

The main contribution of this chapter is a robust and efficient method for computing a skeleton representation from a set of 3D surface points. This 3D skeleton representation can be used for a quantitative analysis of neuronal cell structures, including sub-cellular feature calculations and for neuron type classification based on 3D shapes. Fig. 4.2 shows the three main contributions of this chapter.

There is an extensive literature that tries to solve the neuron skeleton extraction problem [21, 22, 25, 65]. In [22], the skeleton representation is computed from a 3D mesh

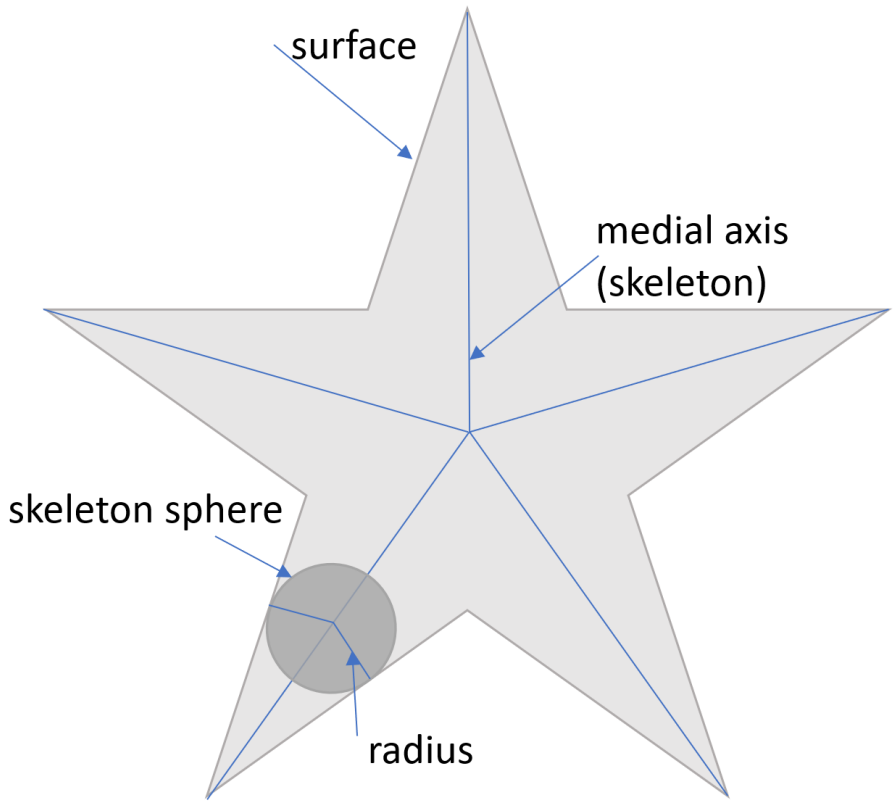

Figure 4.1: Illustration of MAT by using a 2D shape example.

by using a traditional morphological thinning algorithm [66]. This method has two main drawbacks. First, thinning algorithm is sensitive to noise of 3D mesh. Second, in reality, we usually get discrete 3D surface points of neurons from the segmentation step, and constructing 3D mesh from those discrete 3D surface points will introduce additional noise. To make the skeleton extraction model more robust, [25] proposes to use deep learning network to learn skeleton representation. The main idea of the paper is to use the deep learning network to predict skeletons from features of multiple spatial scale layers. This model still takes a continuous surface as input, as opposed to discrete surface points. Further, this is a supervised method and it requires training samples. However, in theory, skeletonization is a well defined mathematical process for continuous surfaces

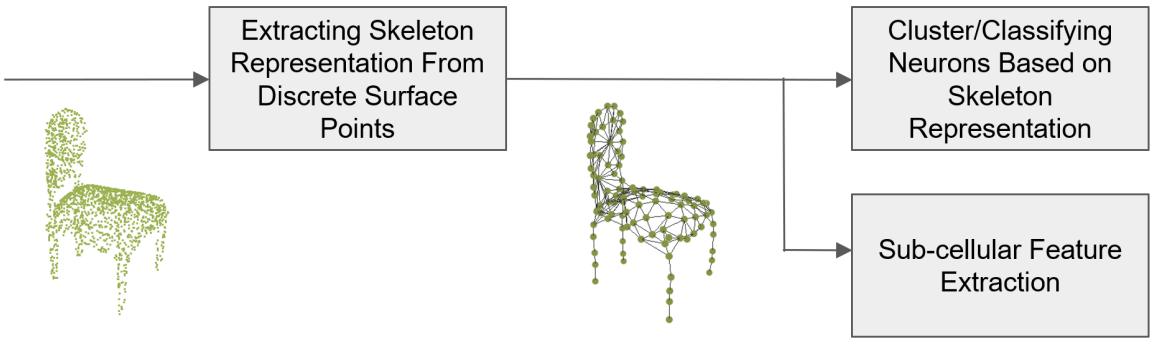

Figure 4.2: Three main contributions of our neuron morphology analysis work.

and should be computable directly. All above methods are used to get the skeleton model from continuous surface. In [65], they propose extracting skeleton representations directly from discrete surface points by using a 3D discrete distance transform. However, this only works well for curve skeletons and only tubular structures have curve skeletons. General 3D shapes will result in surface skeletons as shown in Fig. 4.3. In practice, skeleton mesh is used to represent the surface skeleton. There is no existing method to extract skeleton mesh from surface point clouds for neuron morphology analysis.

There are also methods to analyze neuron shapes using the skeleton representation. For skeleton classification task, [22] proposes to compare similarity of skeletons by using local skeleton features. It breaks a neuron skeleton into short segments and characterize segments by location and direction of that segments. However, this method only works well for curve skeletons but not surface skeletons. In [67], they only consider features from skeleton points for the classification and it does not fully utilize the skeleton mesh information. We also need to extract sub-cellular features such as number of branches from the skeleton mesh because these features are important in the scientific description of neurons as Fig. 4.4 shows.

To analyze general neuron shapes, this chapter presents a robust 3D neuron morphol-

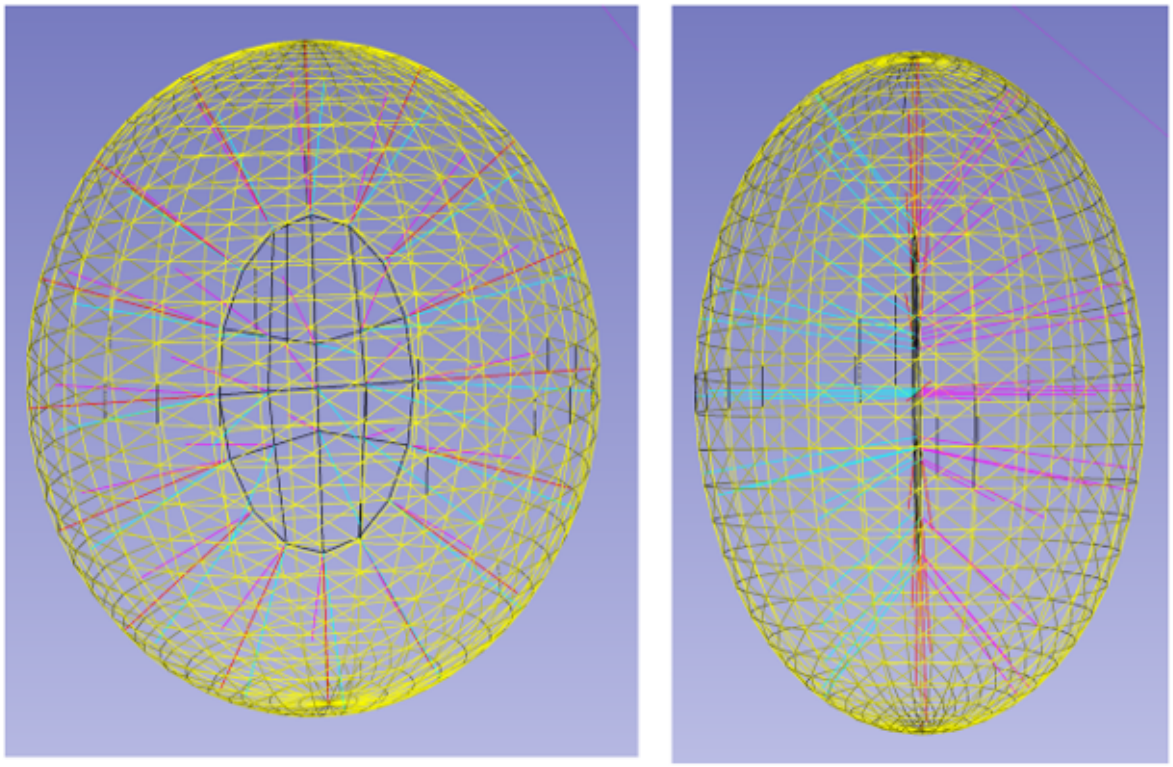

Figure 4.3: Visualization of a 3D ellipsoid shape and its surface skeleton from two points of view. Yellow triangle mesh represents object surface. Black contour represents the outline of the skeleton surface. Magenta and Cyan line segments represent two closest surface points from the skeleton point. Two colors are used to differentiate different directions.

ogy analysis framework based on the surface skeleton representation of neurons. In [8], the authors propose an unsupervised deep learning skeleton mesh extraction method. However, this method does not work well when neurons have concave shapes. Our skeleton mesh extraction method is built upon [8], and by using estimated surface norm of point clouds as part of the optimization function, we address this drawback. Next the skeleton mesh is converted to an undirected graph called skeleton graph. Inspired by [68], we embed the skeleton graph by maximizing mutual information, and then classify neurons based on the embedding of each skeleton. To compute cellular/sub-cellular features of neurons from the skeleton representation, we also utilize the skeleton graph. A simple

Cell ID	Cell Type	NT	Ciliated	Soma location					Morphology	Cell body volume	Number of pre-synaptic sites	Number of post-synaptic sites	Number of putative gap junctions
				Side	Brain region	Z	X	Y					
1	cor-ass BVIN	?	ciliated to bp	L	anterior BV	0.83	9.5	30.6	Single simple axon with slight expansion at terminal in PBV	94.7	2	21	-
2	cor-ass BVIN	?	ciliated to bp	L	anterior BV	1.25	16.5	32.8	Simple cell with single unbranched axon to posterior	100.9	12	14	2
3	BVIN	?	ciliated	L	anterior BV	0.17	23.2	31.9	Simple cell body with single axon that expands forming short branched at terminal in PBV	24.4	6	5	2
4	PNIN	?	no vacuole or membrane contact, does not enter canal	DM	anterior BV	0.59	35.5	28	Simple elongate cell body with thick axon that branches to form 2 colateral axons, one of which branches to form two branches both axon branches terminate at same	96.3	54	30	24

Figure 4.4: Typical scientific description of neurons include number of branches as important morphology features. The description of neurons is from [5].

but effective recursive algorithm is proposed to get number of branches and length of neurons.

We apply our neuron morphology analysis method to classify *Ciona* neurons. The *Ciona* sea squirt is one of the widely studied tunicates in neuroscience [5]. Its brain is closely related to vertebrates with a much simpler neuronal structure. In a single *Ciona* larva, it has only about 187 neurons with about 6600 synapses [5]. Studying the *Ciona* brain in depth can reveal the general principles behind the mechanism of how vertebrate brains work [26]. Therefore, *Ciona* nervous system is ideal for the analysis of neuron circuits with respect to animal behavior.

We also present our results on the NeuroMorpho [28] dataset. In summary, the main contribution of our paper include

- A robust and efficient skeleton mesh extraction method with novel cost function by using properties of MAT. To the best of our knowledge, this is the first one to use skeleton mesh instead of the curve skeleton to analyze neuronal shapes
- A 3D *Ciona* neuron dataset that can be used for neuron morphology analysis.

4.2 Method

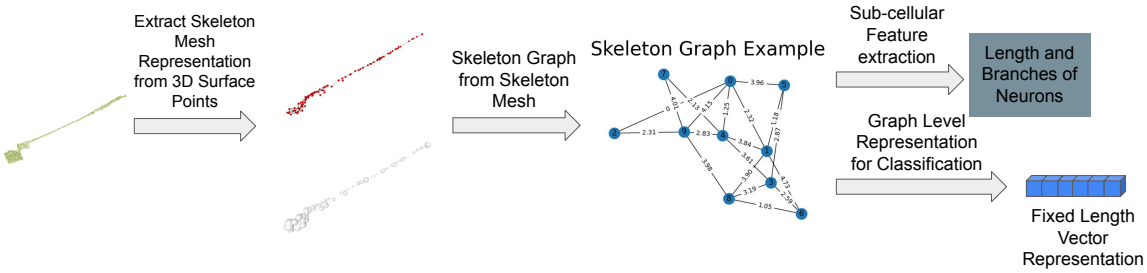

Figure 4.5: Overview of our proposed neuron morphology analysis pipeline. Given a surface point cloud as input, we extract the skeleton mesh. The skeleton mesh includes skeleton points with their radii as well as the connection of skeleton points. Then we construct the skeleton graph. Each node in a skeleton graph represents a skeleton point, and edge in the graph represents the connection between skeleton points. Next, we propose the graph analysis method to get length and number of branches of neurons based on the skeleton graph. We also use the skeleton graph for classification task by embedding it into a fixed length vector.

Fig. 4.5 illustrates our overall neuron morphology analysis method. Given a set of surface point clouds as input, we introduce an unsupervised deep learning method to get the skeleton mesh representation of each neuron. This is achieved by using the properties of the traditional medial axis transform (MAT). The skeleton mesh representation includes skeleton points with radii as well as the connection of those skeleton points as shown in Fig. 4.5. Second, the skeleton representation of each neuron is transformed into a skeleton graph. The skeleton graph is shown in Fig. 4.5. Each node in the skeleton graph represents a skeleton point. If there is an edge between two nodes, it means those

two skeleton points are connected. The weight of the edge represents distance between the two skeleton points. Each node in the skeleton graph represents a skeleton point. If there is an edge between two nodes, it means those two skeleton points are connected. The weight of the edge represents distance between the two skeleton points. Radii as well as the location of each skeleton point are attributes of each node. Next, length and number of branches of neurons are computed from the skeleton graph. To compare different shapes of neurons, the graph level representation learning method is used to embed the skeleton graph. The representation learning method is an unsupervised method by maximizing mutual information of the skeleton graph.

4.2.1 Skeleton Mesh from 3D Surface Point Cloud

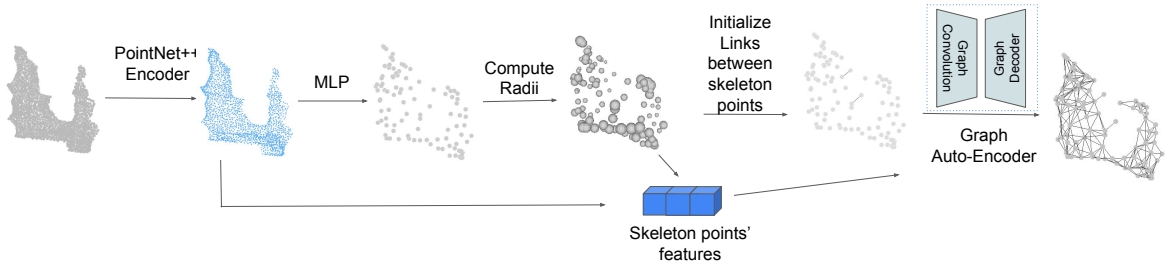

Figure 4.6: Overview of neuron skeleton representation method. Given 3D surface point cloud as input, PointNet++ [6] is used to extract features of the input point cloud. Then a geometric transformation learned by MLP will predict the skeleton points location with their radii. After skeleton points prediction, two simple priors are used to initially connect some skeleton points, and a graph auto-encoder is used to predict all links that connect skeleton points.

Our unsupervised 3D neuron skeleton extraction method is built upon the method in [8] as illustrated in Fig. 4.6. Given a 3D surface point cloud as input, PointNet++ [6] is used as the encoder to obtain the sampled surface points with features. Next, a Multi-Layer-Perceptron (MLP) is used to learn the geometric transformation to predict the skeleton points with their radii using linear combination of the MLP input points.

Compared to [8], we propose to use properties of skeleton points as the prior knowledge to make the geometric transformation learning process more robust to general shapes. After getting skeleton points with radii, a graph auto-encoder is used to predict links between skeleton points.

skeleton points prediction

Mathematically, given a set of 3D surface points $\mathbf{P} \in \mathbb{R}^{M\times 3}$ where M is a number of points, we want to predict N skeleton spheres $\mathbf{s}_i =< \mathbf{c}_i, r_i >$ where $\mathbf{c}_i$ is the center of each sphere and r_i is the radius of each sphere.

As illustrated in Fig. 4.6, we first use PointNet++ [6] as the encoder to obtain a set of sampled input points $\mathbf{P'} \in \mathbb{R}^{M'\times 3}$ and their associated contextual features $\mathbf{F} \in \mathbb{R}^{M'\times D}$. $M'(M' < M)$ is the number of feature points after PointNet++ and D is the feature dimension of each feature point.

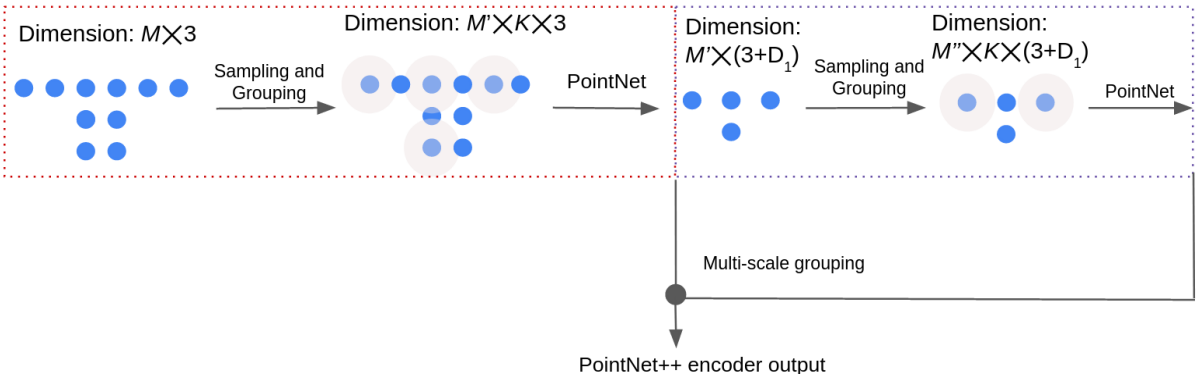

Figure 4.7: PointNet++ Encoder. Each dashed box represents a set abstraction level. The PointNet++ encoding features are multi-scale grouping features from multiple spatial scales.

PointNet++ groups points and extract point features hierarchically as Fig. 4.7 shows. It contains a number of set abstraction levels. For each set abstraction level, there are three layers: Sampling layer, Grouping layer, and PointNet layer. For the first set

abstraction level, the input is $\mathbf{P}$, a set of M number of 3D surface points. Therefore, the input size to the first set abstraction level is $M \times 3$. Next, Sampling layer is applied. The iterative farthest point sampling (FPS) is used to get M' sampled points. In the grouping layer, those M' sampled points are used as the centroid points. Then for each centroid, all M points within a radius are viewed as neighbor points and are grouped into that local region. Therefore, each centroid has K neighbors and K can vary for different groups. After Sampling and Grouping layers, PointNet [69] layer is used to extract features for each local region. The sampling layer, grouping layer, and PointNet layer consists one set abstraction level, and we stack such abstraction levels to form a hierarchical architecture to get features at different spatial scales. Next, multi-scale grouping is applied to concatenate the features from different spatial scales.

As Fig. 4.6 shows, a Mult-Layer-Perceptron (MLP) is used to estimate the geometrical transformation to get a set of skeleton spheres' center points $\mathbf{C}$, $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$. The geometric transformation we use is a convex combination of input points $\mathbf{P}'$. MLP with softmax function is used to estimate the weight $\mathbf{W} \in \mathbb{R}^{M' \times N}$ of the convex combination in eq. 4.1.

$$\mathbf{C} = \mathbf{W}^T \mathbf{P}' \qquad \text{subject to} \qquad \forall j \in \{1, ..., N\} \quad \sum_{i=1}^{M'} W_{i,j} = 1 \tag{4.1}$$

As shown in [8], the same weight matrix $\mathbf{W}$ can be used to compute $r(\mathbf{c}) \in \mathbf{R}$ using eq. 4.2

$$\mathbf{R} = \mathbf{W}^T \mathbf{D} \tag{4.2}$$

where $D \in \mathbb{R}^{M' \times 1}$ is a vector collecting all $d(\mathbf{p}', \mathbf{C})$. $d(\mathbf{p}', \mathbf{C})$ is the closest distance of one surface point $\mathbf{p}'$ to all skeleton points $\mathbf{C}$ and is defined as $d(\mathbf{p}', \mathbf{C}) = \min_{\mathbf{c} \in \mathbf{C}} ||\mathbf{p}' - \mathbf{c}||_2$

A set of loss functions are defined in [8] to train the MLP. The loss function includes sampling loss L_s, point-to-sphere loss L_p, and radius regularizer loss L_r. The first two losses are based on the recoverability of skeleton representation. The last loss term is to encourage larger radii to avoid instability induced by surface noise.

For the sampling loss, we sample points on the surface of each skeleton sphere and measure the Chamfer Distance (CD) between the set of sampled points $\mathbf{T}$ and the input points $\mathbf{P'}$ as in eq. 4.3:

$$L_s = \sum_{\mathbf{p'} \in \mathbf{P'}} \min_{\mathbf{t} \in \mathbf{T}} ||\mathbf{p'} - \mathbf{t}||_2 + \sum_{\mathbf{t} \in \mathbf{T}} \min_{\mathbf{p'} \in \mathbf{P'}} ||\mathbf{t} - \mathbf{p'}||_2 \tag{4.3}$$

Point-to-sphere loss L_p loss measures the reconstruction error by explicitly optimizing the coordinate of skeleton points and their radii:

$$L_p = \sum_{\mathbf{p'} \in \mathbf{P'}} (\min_{\mathbf{c} \in \mathbf{C}} ||\mathbf{p'} - \mathbf{c}||_2 - r(\mathbf{c}_{p'}^{min})) + \sum_{\mathbf{c} \in \mathbf{C}} (\min_{\mathbf{p'} \in \mathbf{P'}} ||\mathbf{c} - \mathbf{p'}||_2 - r(\mathbf{c})) \tag{4.4}$$

where $\mathbf{C}$ is a set of predicted skeleton points, $\mathbf{P'}$ is a set of sampled surface points from PointNet++, $r(c)$ is a radius of skeleton point $\mathbf{c}$, and $\mathbf{c}_{p'}^{min}$ is the cloest skeleton points to a point $\mathbf{p'}$.

Radius regularizer loss L_r is defined in eq. 4.5 where $r(\mathbf{c})$ is a radius of the skeleton point $\mathbf{c}$. By minimizing this loss, it encourages large radii of skeleton points to make the skeleton points prediction more stable.

$$L_r = -\sum_{\mathbf{c} \in \mathbf{C}} r(\mathbf{c}) \tag{4.5}$$

However, based on above three losses, predicted skeleton points can be outside of a shape if the shape is concave. Therefore, we introduce the skeleton-to-surface norm loss

L_n to encourage the skeleton points to be inside the shape. L_n is a term to measure the reconstruction error by utilizing the property of spoke direction in MAT. Fig. 4.8 illustrates a spoke of a skeleton point in MAT. The length of a spoke of the skeleton point $\mathbf{c}$ is $r(\mathbf{c})$. This is also one of our main contribution compared to [8] for skeleton points prediction. In theory, the direction of the spoke should be perpendicular to the object surface at the surface point [67]. Also the spoke direction should be pointing outside of a shape. To capture this property, we define L_n:

$$L_n = \sum_{\mathbf{c}\in\mathbf{C}}(1 - \mathbf{n}_{\mathbf{p}_c^{'min}} \cdot \frac{\mathbf{p}_c^{'min} - \mathbf{c}}{||\mathbf{p}_c^{'min} - \mathbf{c}||_2}) + \sum_{\mathbf{p}'\in\mathbf{P}'}(1 - \mathbf{n}_{p'} \cdot \frac{\mathbf{p}' - \mathbf{c}_{p'}^{min}}{||\mathbf{p}' - \mathbf{c}_{p'}^{min}||_2}) \tag{4.6}$$

$\mathbf{p}_c^{'min}$ is the closest surface point to the skeleton point c and $\mathbf{n}_{\mathbf{p}_c^{'min}}$ is the estimated surface norm of the surface point. $\cdot$ is the dot product between two vectors. To estimate the norm of each point in the 3D surface points, the adjacent points are found first and then principal axis of the adjacent points using covariance analysis are calculated. More details of the norm estimation of each surface point are described in [70]. L_n encourages the skeleton points to be located within a shape even the shape is concave.

Links prediction

After predicting skeleton points, our target is to predict links to connect skeleton points to form the skeleton mesh. In theory, for any pair of skeleton points, if all points that are on the line connecting the two skeleton points are also on the skeleton surface, there should be links between those two points. We adapt the graph auto encoder (GAE) as used in [8] to predict links between skeleton points. GAE takes input the initialized adjacency matrix A_{ini} of the skeleton mesh graph G_{mesh} and the skeleton points' features.

Based on two priors, we initialize a set of reliable links and links are divided into known existing links, known absent links, and unknown links. These two priors are the

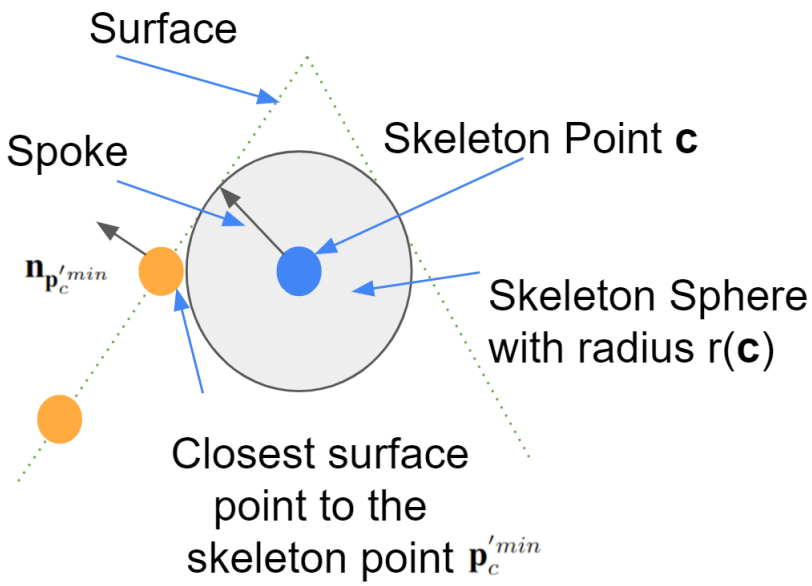

Figure 4.8: Spoke is a vector connecting a skeleton point and that skeleton point's one of two closest surface points. The vector points from the skeleton point to the surface point. Green lines represent the surface of an object, blue dot is one skeleton point, and the arrow represents a spoke. Spoke is perpendicular to the object surface at the surface point.

topology prior and the recovery prior.

- Topology prior: There is a link between two closest skeleton points and this link is marked as the known existing link. There will be no link between two farthest nodes and it is marked as known absent link.

- Recovery prior: If a point $\mathbf{p}'$ has two closest skeleton points $\mathbf{c}_i$ and $\mathbf{c}_j$, then there is a link between $\mathbf{c}_i$ and $\mathbf{c}_j$ and the link is marked as known existing link.

All other links are marked as unknown.

In general, after using these two priors, there are missing links. Therefore, the graph auto-encoder [8,71] is used to predict links. The encoder is a series of graph convolutional network layers with residual blocks and a simple inner product decoder to produce the predicted adjacency matrix $\hat{\mathbf{A}}$ with each entry as 0 (there is no link) and 1 (there is a

link).

The input of the encoder is the initialized graph (skeleton points with features, adjacency matrix of the graph). The adjacency matrix $\mathbf{A}$ is used to represent the input adjacency matrix. The skeleton points' features is concatenation of $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{W}^T\mathbf{F}$. $\mathbf{C}$ are coordinates of skeleton points, $\mathbf{R}$ are radii of skeleton points, $\mathbf{W}$ is the learned weights from the MLP, and $\mathbf{F}$ is the contextual features of the surface points from PointNet++. The loss function is a Masked Balanced Cross-Entropy (MBCE) loss as proposed in [72]. Given the input column of adjacency matrix $\mathbf{a}_i$ and the output column of estimated adjacency matrix $\hat{\mathbf{a}}_i$, MBCE is defined in eq. 4.7 as:

$$L_{MBCE} = \frac{\mathbf{m} \bigodot L_{BCE}}{\sum_j m_j} \tag{4.7}$$

$\bigodot$ is the element wise product. $m_j \in \mathbf{m}$ and m_j is the boolean function: $m_j = 1$ if the jth element of $\mathbf{a}_i$ is unknown, else $m_j = 1$. L_{BCE} is defined in eq. 4.8 as:

$$L_{BCE} = -\mathbf{a}_i log(\sigma(\hat{\mathbf{a}}_i)) \cdot \zeta - (1 - \mathbf{a}_i) log(1 - \sigma(\hat{\mathbf{a}}_i)) \tag{4.8}$$

L_{BCE} is the balanced cross-entropy loss with weighting factor $\zeta = 1 - \frac{\#\text{known present links}}{\#\text{known absent links}}$, and σ is the sigmoid function. ζ is used as a weight multiplier for the positive class to solve the sample imbalance problem because in the skeleton mesh, most pair of skeleton points will not have links between them.

4.2.2 Sub-cellular feature extraction from the skeleton graph

Skeleton model can be represented as the skeleton graph $G(V, E)$ where V represents all skeleton points and E represents connection between skeleton points. Note that, the main difference between the skeleton graph and the graph we use to predict links between

skeleton points is that we also have weights for edges in the skeleton graph. The weight of the edge represents distance between skeleton points.

Neuron length computation from skeleton graph

We formulate the neuron length computation problem as finding the longest simple path in the skeleton graph G. A simple path in the graph is a path that does not have repeat nodes. In G, the length of the path is the sum of all edges' weights along the path. Note that our skeleton graph can have loops. To solve this problem, for each node, we find the longest simple path from that node and denote as $path(i)$ for node i. To avoid getting stuck during the loop, we mark any node when we visits as shown in the algorithm below. Next, we find i^* that maximizes $path$. We use the following recurrent algorithm to find the longest simple path from one node in the skeleton graph.

Algorithm 3 Find the longest simple path from node i

```
D[j] represents the longest path from j to i. Initially, D[j]=0 for all j
function LongestPath(i, currLength)
    if Node i is visited then
        return
    end if
    change i status to visited
    if D[i] <currLength then
        D[i]=currLength
    end if
    for all nodes j that is connected to i do
        LongestPath(j, currLength+edgeweight[i,j])
    end for
    change i status to not visited
end function
```

Neuron branch calculation

After finding the longest simple path, we are able to identify a set of nodes on that path. Those nodes are possible branching nodes. We name a set containing all possible

branching nodes as **B** For each node $i \in B$, we find the longest simple path from that node i which does not contain any other nodes in B. Therefore, the branch is identified as the longest simple path.

4.2.3 Skeleton Model Comparison

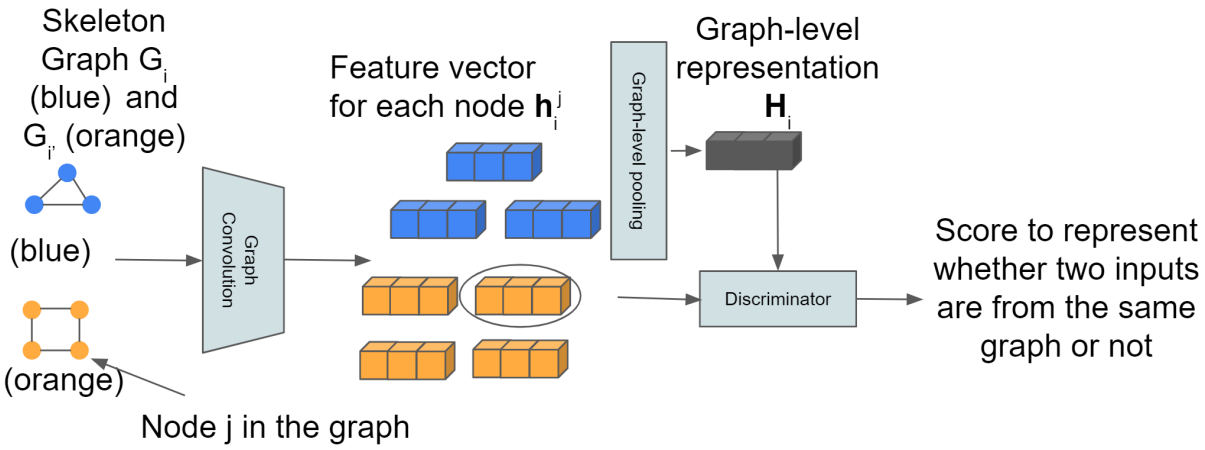

Figure 4.9: We use two example skeleton graphs (blue and orange) to demonstrate how we embed the skeleton graph. Each node of a skeleton graph is encoded into a feature vector by using graph convolution layers. A fixed length graph level feature vector (global representation) is obtained by graph-level pooling operation of each node feature vector. The discriminator takes inputs both global representation and patch representation to decide whether they are from the same skeleton graph. In this toy example, there will be 14 global-patch pairs.

We cluster neuron morphology by comparing different skeleton graphs. Specifically, we embed the skeleton graphs and then cluster neurons based on their embeddings. We embed the skeleton graph based on InfoGraph [68] as illustrated in Fig. 4.9. The embedding process is in an unsupervised manner.

First, graph convolutional layers are used to generate node features which we also call patch representation $\mathbf{h}_i^j$ (i is the skeleton graph index and j is the node index of the skeleton graph i). Then graph-level pooling is used on all patch representations to get the

graph level representation (global representation) $\mathbf{H}_i$. We define our mutual information (MI) estimator on global-patch pairs over the given graph dataset $\mathbf{G}$ in eq. 4.9

$$MI = \sum_{i \in K} \frac{1}{K} \sum_{j \in G_i} I(\mathbf{h}_i^j, \mathbf{H}_i) \tag{4.9}$$

where K is the total number of graphs in the dataset and $G_i \in \mathbf{G}$.

MI is the mutual information estimator modeled by the discriminator T. The Jensen-Shannon MI estimator proposed in [68] as eq. 4.10

$$I(\mathbf{h}_i^j, \mathbf{H}_i) = \mathbb{E}[-sp(-T(\mathbf{h}_i^j, H_i)] - \mathbb{E}[-sp(-T(\mathbf{h}_{i'}^j, H_i)] \tag{4.10}$$

where $\mathbb{E}$ is the expectation (here it is just average operation) and $sp(z) = log(1 + e^z)$. i and i' denote two graph samples from the dataset $\mathbf{G}$. The discriminator T estimates global-patch representation pairs by passing two representations to different non-linear transformations and then takes the dot product of the two transformed representations. Both non-linear transformations consist 3 linear layers with ReLU activation functions. Therefore, the discriminator will output a score between $[0, \infty)$ to represent whether the input patch (node) is from the input graph. If the input global/patch pairs are from the same graph, we refer to them as positive samples, otherwise negative samples. We randomly sample pairs as input to the discriminator.

4.3 Dataset

4.3.1 *Ciona* Neuron EM Dataset from UCSB MCDB the Smith Group [5]

This dataset (Dataset 4) contains two *Ciona* larva 3D TEM images. Fig. 4.10 shows how we use *Ciona* to study the neuron activities with respect to environmental changes. The section thickness for TEM images varies between 35 nm and 100 nm. For each section, xy resolution is 3.85×3.85 nm. Animal 1 contains 7671 sections and animal 2 contains about 7000 sections. In each Ciona larva, 187 neurons are annotated. Those 187 neurons can be grouped into 31 types. Fig. 4.11 shows how *Ciona* neurons are located in their brains. For animal 1, *Ciona* neurons are manually segmented using Reconstruct [31] and 3D surface point clouds are available. For animal 2, *Ciona* neuron skeletons are traced using TrackEM2 [32], an ImageJ [29] plugin. Details of this dataset are described in Table 4.1

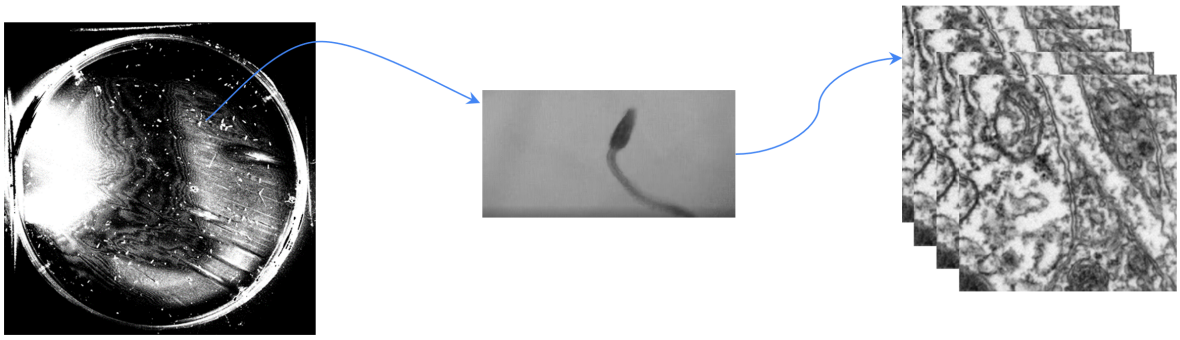

Figure 4.10: This figure illustrates *Ciona* larva under environmental changes (light on/off), real *Ciona* larva, and *Ciona* larva brain EM images.

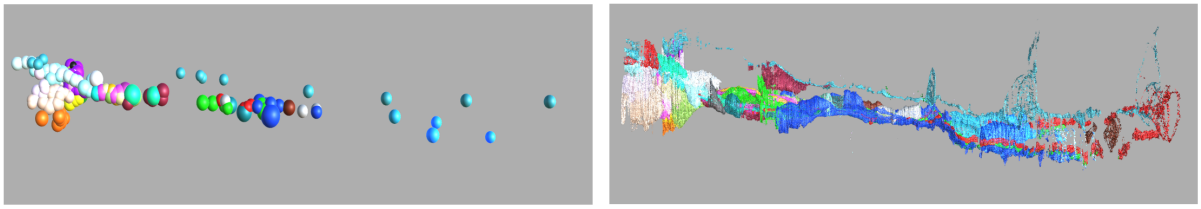

Figure 4.11: Neurons in *Ciona*. Different colors represent different types of neurons. Left: each neuron is represented as a sphere; right: each neuron is represented by their surface points.

Table 4.1: Details of *Ciona* Dataset. It contains two *Ciona* animals, one with surface point cloud annotated and one with skeleton annotated.

Animal	xy resolution (nm)	section thickness (nm)	number of sections	annotations
Animal 1	3.85×3.85	35-100	7671	3D surface point cloud of neurons are provided
Animal 2	3.85×3.85	35-100	6928	3D neuron skeletons without skeleton points' radii

Figure 4.12: Three examples of surface point clouds from *Ciona* Dataset

4.3.2 *C.elegans* Neuron Dataset from NeuroMorpho [28]

NeuronMorpho [28] is a publicly available dataset that is used for neuron morphology research. It has tens of different animals' neurons. So far, it is the largest neuron skeletons dataset with associated metadata. In this paper, we take a subset of *C.elegans* dataset (Dataset 5) from the whole NeuroMorpho dataset to verify our method. Dataset 5 consists of 299 neuron skeletons (with radii) and it is classified into 10 different types. Each neuron with detailed metadata information such as number of branches and length of neuron.

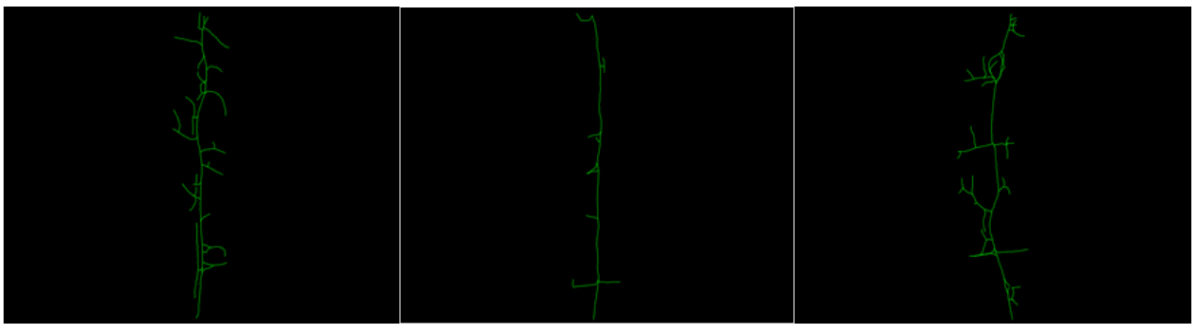

Figure 4.13: Example skeletons from NeuroMorpho Dataset

4.3.3 ShapeNet 3D CAD models [1]

ShapeNet [1] is a richly-annotated, large-scale dataset of 3D shapes represented by 3D CAD models of objects. It contains 3D models from a multitude of semantic categories. In this paper, we collect 7088 shapes from 8 categories of ShapeNet, the same as in [8] for a fair comparison. Each shape has 2000 sampled 3D surface points. This dataset is used to train the skeleton extraction model and test the robustness of our skeleton extraction model and the skeleton graph embedding method. Examples from ShapeNet are shown in Fig. 4.14.

Figure 4.14: Example CAD shapes from ShapeNet

4.3.4 Skeleton Model from 3D Surface Point Cloud

We apply our method on animal 1 neurons from Dataset 1 for the purpose of building a shape model to analyze neuron morphology. We train our skeleton model extraction method using all 3D models from Dataset 3. To get the constant number of 3D surface input points, we use the sampling strategy described in [73]. The main idea of the sampling strategy is to give each point a weight based on its distance to neighbor points. Then we sample points based on the weights until we reach the number of points we want. Details of defining the neighbor points and computing the weight is described in [73].

To evaluate our skeleton extraction method on Dataset 1, we repair surface mesh using screened poisson surface reconstruction method [74] with spherical harmonics to smooth the surface as illustrated in Fig

Fig. 4.15 shows the qualitative comparison of our methods and other state-of-the-art methods [7, 8]. Our method has better visual results. [7] can generate unstructured skeleton points but it lacks topological constraint. It performs well when neuron has tube like structure but it is not good when neuron has more circular shape. Compared with [8], our method can capture more detailed structures which are important for sub-cellular feature extraction, such as branches. For quantitatively evaluation of our method on

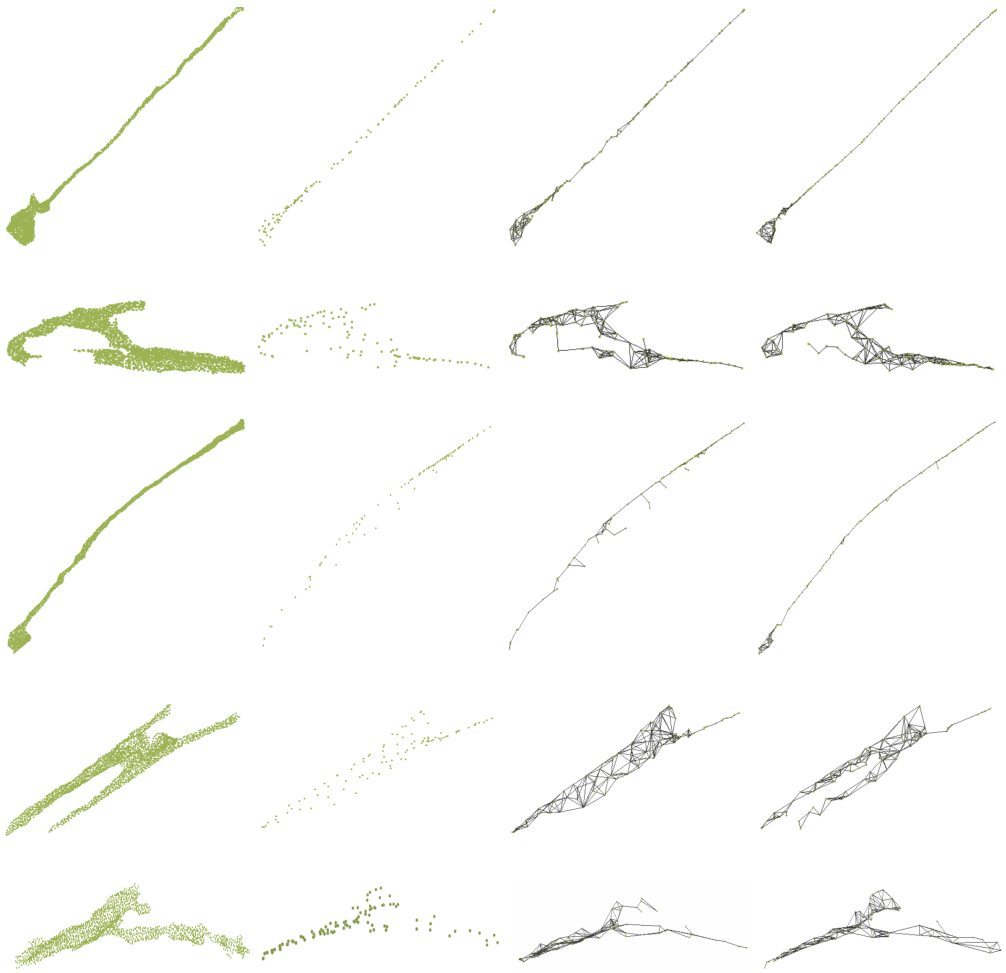

Figure 4.15: The figure shows skeleton extraction results from different methods. From left to right:Input 3D surface points; skeleton points from surface points using DPC [7]; skeleton mesh from surface points using Point2Skeleton [8]; skeleton mesh from our method with surface norm cost function.

Dataset 1, we compute the strictly defined MAT and use the handcrafted method in [8] to remove insignificant spikes to get the simplified MAT. We sample points on the simplified MAT as the ground truth skeleton points. Then we compute Chamfer Distance (CD) and Hausdorff distance (HD) between computed skeleton points and ground truth skeleton points. CD and HD are both distance measurements to measure distance between two sets of points. CD is computed by summing he squared distances between nearest neighbor correspondences of two point clouds. Mathematically, given two sets of point X and Y,

Figure 4.16: Five examples of repaired mesh and their surface point clouds. Top row is the original point clouds and bottom row are repaired meshes.

it is defined as:

$$CD(X,Y) = \sum_{x \in X} \min_{y \in Y} ||x-y||^2 + \sum_{y \in Y} \min_{x \in X} ||x-y||^2 \tag{4.11}$$

HD is the greatest distance of all Euclidean distances from a point in one point cloud to the closest point in the other point cloud. It is defined as:

$$HD(X,Y) = max\{\sup_{x \in X} \inf_{y \in Y} d(x,y), \sup_{y \in Y} \inf_{x \in X} d(x,y)\} \tag{4.12}$$

where $d(x,y) = \sqrt{||x-y||^2}$. We refer to them as CD-skel and HD-skel. To compute CD-skel and HD-skel, we shift and rescale each skeleton so that the skeleton center is located at (0,0,0) and their x,y,z coordinates are all between -1 and 1 for every skeleton. We also measure the difference between the shapes reconstructed from the skeletons and ground truth surface points using CD and HD. We refer them as CD-recon and HD-recon. Similarly, we also shift and rescale the ground truth points so that each neuron is centered at (0,0,0) and each neuron's surface coordinates are between -1 and 1.

Other than those four aforementioned evaluation metrics, we also use the reconstructed neuron volume difference as the evaluation metric considering neuron volume is one of the important property of neuron. We denote it as $vol-pct$. Mathematically, it is defined as $vol-pct = \frac{|vrecon-vg|}{vg}$ where $vrecon$ is the volume of the reconstructed neurons from the skeleton model, and vg is the ground truth volume. Table 4.2 gives the detailed results of different methods. It shows our method has the best performance compared to other methods on Dataset 1 in terms of all evaluation metrics we use.

Table 4.2: Quantitative Comparison with state-of-the-art skeleton model extraction method on Dataset 1

	CD-recon	HD-recon	CD-skel	HD-skel	vol-pct (%)
DPC [7]	0.102	0.298	0.303	0.311	10.1
Point2Skeleton [8]	0.081	0.207	0.155	0.191	8.2
Our method	**0.067**	**0.183**	**0.090**	**0.185**	**5.6**

To test the generality of our method, we apply it on Dataset 6. We use the 6 fold cross validation strategy. We randomly split Dataset 6 into 6 folds. Each time, one fold is used as the test set and the remaining 5 folds are used to train the model. We do this 6 times over 6 different testing data and we get mean and standard deviation of CD-recon, HD-recon, CD-skel, and HD-skel. The same cross validation strategy is used for Point2Skeleton [8] method. DPC [7] is not learning based method, and we just apply it on whole Dataset 6. Table 4.3 provides quantitative evaluation results on Dataset 6. The results show that our deep learning method even with estimated surface norms can accurately encode the original shape information in our skeleton models. Fig. 4.17 shows qualitative results of our skeleton mesh extraction method. The skeleton meshes are shown overlaying on the surface point clouds.

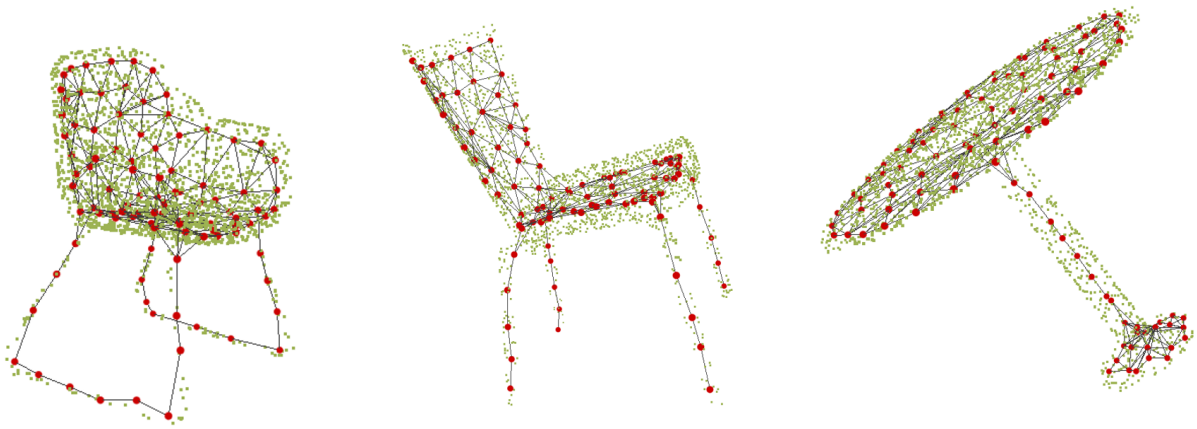

Figure 4.17: Qualitative results on Dataset 6. Blue dots are surface point clouds. Red points with the links represent the skeleton meshes.

Table 4.3: Quantitative Comparison with state-of-the-art skeleton model extraction method on Dataset 6. Number in the bracket is the standard deviation. There is no standard deviation for DPC because it is not based on machine learning.

	CD-recon	HD-recon	CD-skel	HD-skel
DPC [7]	0.063	0.198	0.215	0.281
Point2Skeleton [8]	0.041 (0.008)	0.168 (0.022)	0.073 (0.006)	0.181 (0.029)
Our method	**0.028** (0.007)	**0.149** (0.020)	**0.051** (0.006)	**0.162** (0.030)

4.3.5 Sub-cellular feature extraction from skeleton model

We apply our sub-cellular feature extraction method on Dataset 4 and Dataset 5. We define the length difference percentage (len-pct) and number of branches difference percentage (num-pct) to measure the neuron length error and number of branches error of the computation methods. We compare our sub-cellular feature extraction method with the method proposed in [22]. Table 4.4 gives details of sub-cellular feature computation results. Since only animal 2 of Dataset 4 has ground truth neuron length and number of branches, len-pct and num-pct for Dataset 4 is only for animal 2. Our method provides the better sub-cellular feature extraction results in most cases and the percentage error is no more than 8 percent.

Table 4.4: Sub-cellular feature evaluation results

	len-pct Dataset 4 (%)	num-pct Dataset 4 (%)	len-pct Dataset 5 (%)	num-pct Dataset 5 (%)
NAVIS [22]	7.1	10.5	**3.2**	6.3
Our method	**5.5**	**7.6**	3.8	**5.8**

For Dataset 4, we also analyze the relationships between length and branches of neurons computed from our method. For animal 1 in Dataset 4, we first use our skeleton extraction method to get the skeleton representation of each neuron. Next, we use our sub-cellular feature extraction method to get length of the neuron and number of branches of the neuron. Fig. 4.18 shows the relationships between length and branches of neurons. Blue dots represent neurons from animal 1 and red dots represent neurons from animal 2. Overall, longer neurons tend to have more number of branches. Also animal 1 and animal 2 have similar results in terms of relationships between neuron length and number of branches.

4.3.6 Skeleton Model Comparison

As Fig. 4.19 demonstrates, different types of neurons have different shapes visually. In this chapter, we consider using the skeleton mesh model to quantify those differences. Also we try to classify them based on this shape difference.

We apply our skeleton model comparison method on Dataset 4 for the purpose of analyzing *Ciona* neuron morphology of different neuron types. Given the skeleton model, we embed it into a vector. Then we use K-means++ to cluster vector representations of skeleton graphs. For K-means++, the number of clusters is set to be the same as number of neuron classes. After K-means++, the cluster label is given by using the majority vote of neuron types within the cluster. We use animal 1 neurons to train the K-means++ model and get the cluster (neuron type) centers. Next, we use animal 2 as the test set. For each neuron in the test set, we assign the label based on its closest cluster center. The distance metric we use is the euclidean distance in the vector embedding space. Table 4.5

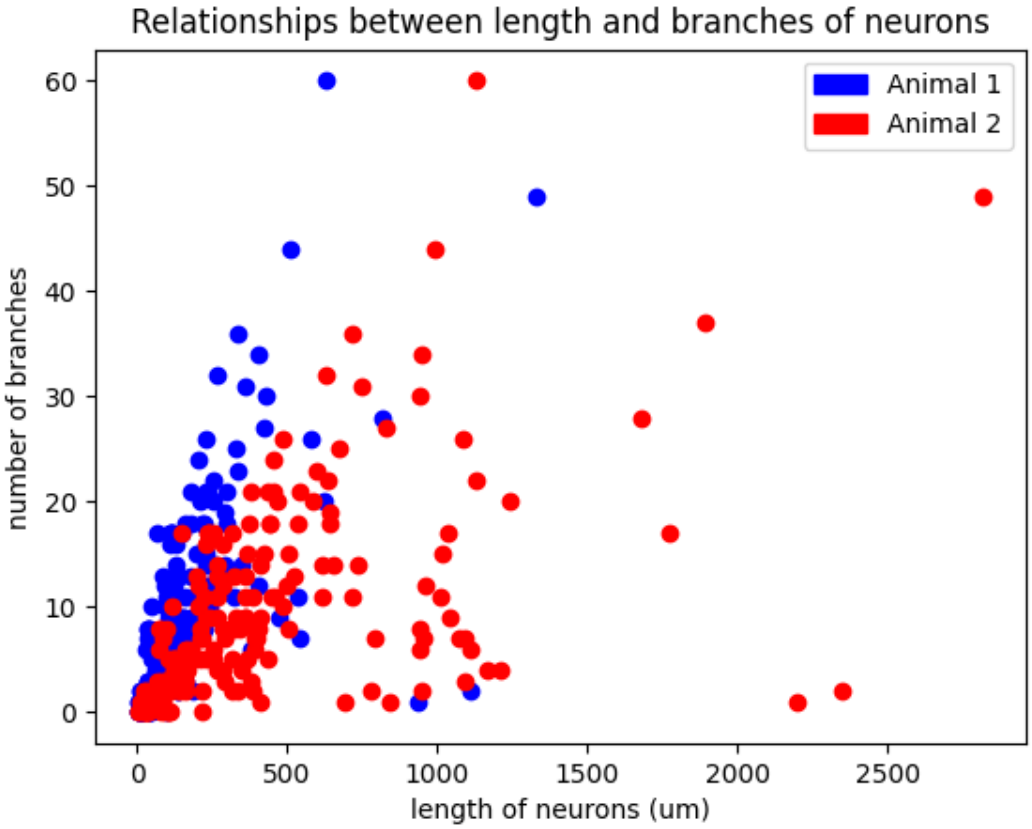

Figure 4.18: Relationships between length and number of branches of neurons using two animals of Dataset 4. Blue dots represent neurons from animal 1 and red dots represent neurons from animal 2.

shows the comparison of clustering (classification) accuracy on both training and test sets using different neuron classification method. The neuron classification methods include graph spectrum method, graph2vec [75], s-rep [67] and our graph level representation method. The graph spectrum method uses the eigen values of the graph's adjacency matrix to form the vector representation. Similar to our method, graph2vec method is another way to convert the skeleton graph to the graph level vector representation. For the s-rep method, it uses the skeleton points' features such as spoke direction, spoke length, and skeleton points' locations to classify neurons. From Table 4.5, we can see

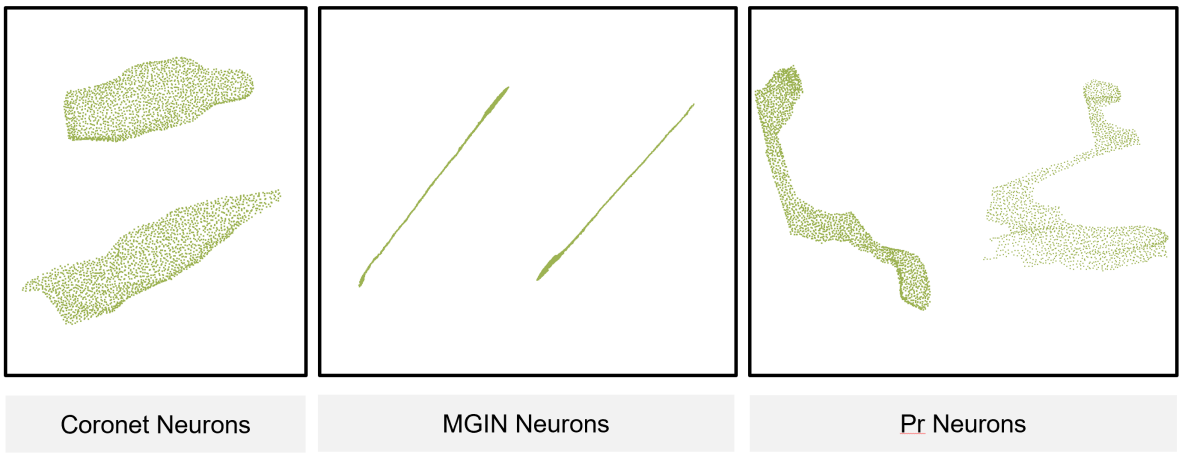

Figure 4.19: Visualization of how different neurons have different shapes. Neurons within the same box are the same type.

that grouping neurons by just using basic skeleton graph spectrum can provide decent classification results on both train and test sets. It shows that neuron types are closely related to its morphology. Also, our method is a better way to represent skeleton graphs in terms of clustering accuracy. Fig. 4.20 shows the confusion matrix on the test set using our classification method. As we see, most cells are classified correctly (on the diagonal).

Table 4.5: Neuron Classification Results

	Graph Spectrum	Graph2vec [75]	S-rep [67]	Our method
Train	0.691	0.767	0.791	**0.893**
Test	0.632	0.718	0.773	**0.871**

Based on previous observations, we do further morphology analysis based on our graph level representation results. After we get the vector representation of each graph, we compute euclidean distance between each pair of vectors. Then we compute the inter class and intra class distance based on pairwise neuron distance as Fig. 4.21 shows. As Fig. 4.21 illustrates, diagonal entries tend to be smaller than other values. It further confirms our conclusion that neuron types are related to neuron morphology. More

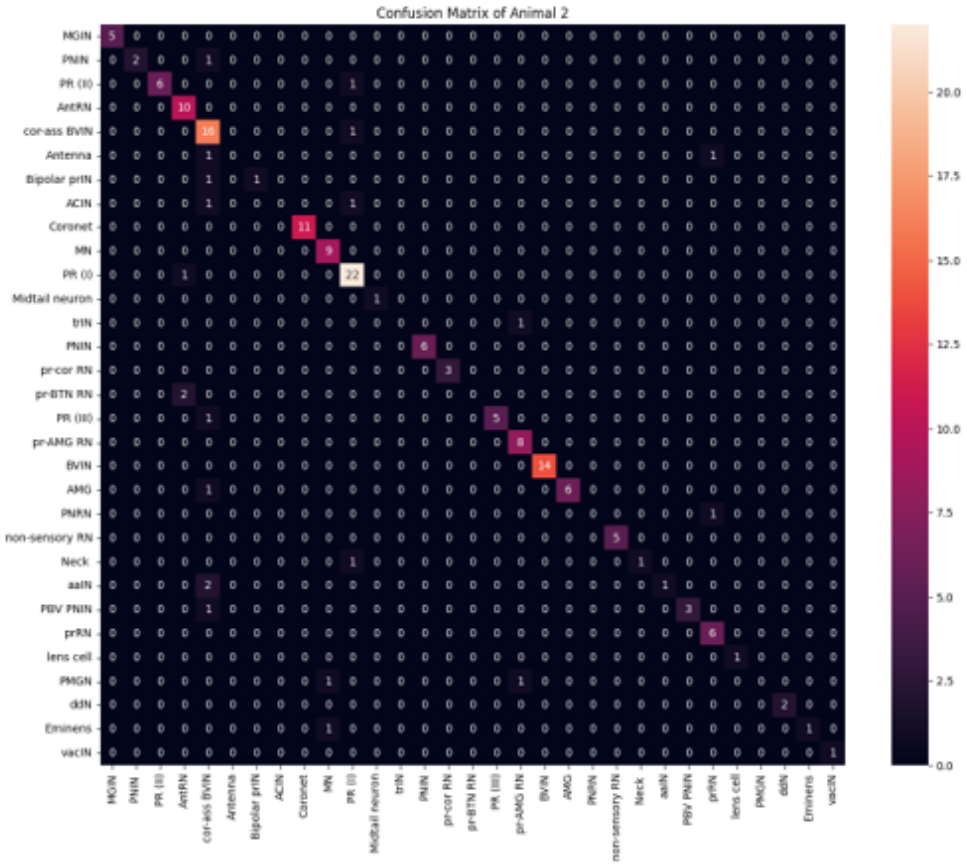

Figure 4.20: Confusion matrix of neuron classification on animal 2 (test set) using our method.

specifically, neurons within a neuron type tend to have smaller morphology distance than neurons between different groups. Also, two animals inter and intra distance look very similar.

Based on this inter and intra class distance, we do hierarchical clustering as Fig. 4.22 shows. The hierarchical clustering results show that BVIN and pr-BTN RN have larger morphology distances from other neuron types. The BVIN neurons are a broad group of intrinsic interneurons located in the brain vesicle of *Ciona*. The main role of this group is to connect the sensory neurons to other groups within the brain vesicle.

The BVIN neurons have partial subclassification based on sensory input [5]. Receiving specific sensory information is an indication of functional role, therefore, the BVIN can be further subdivided into different groups based on the sensory group(s) from which they receive input. Using the sensory input as criteria, the entire group was split up into four groups: prIN if receiving photoreceptor input, antIN if receiving antenna cell input, pr-ant IN if receiving from both, and BVIN if not receiving from either. The pr-BTN RN only have two neurons and their functions are mostly unknown. According to the connectome [5], they receive input from both the photoreceptors and the BTN neurons (neurons involved in processing touch stimuli in the tail), so it's possible they play a role in integrating the two inputs. Any functional differences that may exist between the two are currently unknown, however, the hierarchical clustering suggests that this may be the case.

4.4 Conclusion

In this chapter, we propose a novel neuron morphology analysis pipeline as shown in Fig. 4.23. It mainly includes three parts. First, we propose a robust shapre representation using skeleton mesh. Next, we compute sub-cellular features from the skeleton mesh. Finally, we compare different neuron shapes using skeleton mesh. To the best of knowledge, this is the first time that such an approach is used to represent and classify neuronal shapes. The introduction of the estimated surface norm penalty results in a robust mesh representation that achieves the state-of-the-art performance using well defined metrics. Based on skeleton graph, we formulate sub-cellular feature computation problem as a longest simple path problem that can be easily computed. To compare different neuron morphology, we use a novel unsupervised graph level representation method to get the vector representation of each skeleton graphs. We provide detailed experimental results

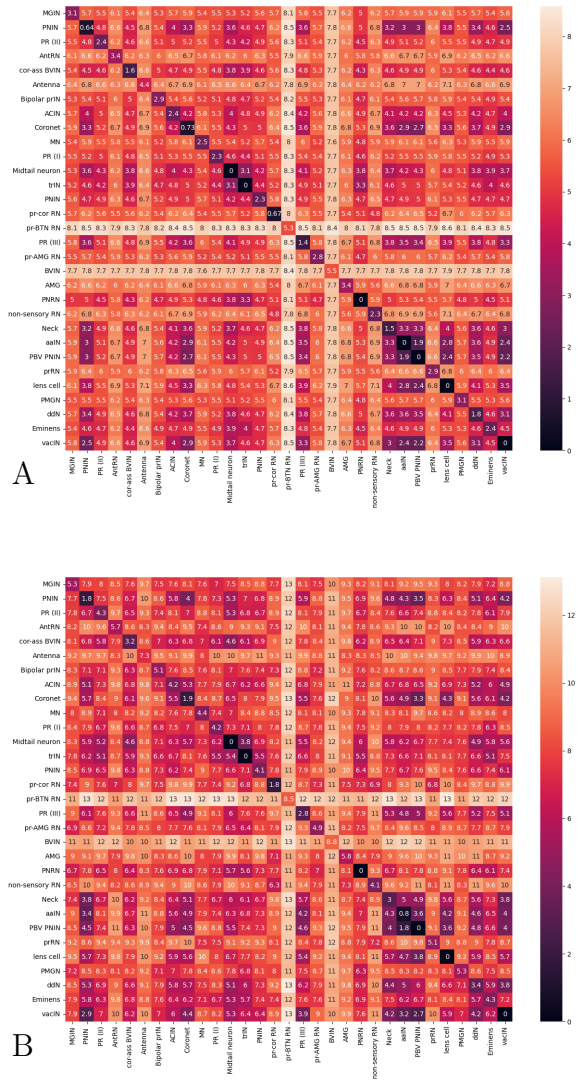

Figure 4.21: Inter and intra class neuron morphology distance on animal 1 (A) and animal 2 (B) . Neuron morphology distance is computed by using euclidean distance between our graph level representation of the skeleton graph.

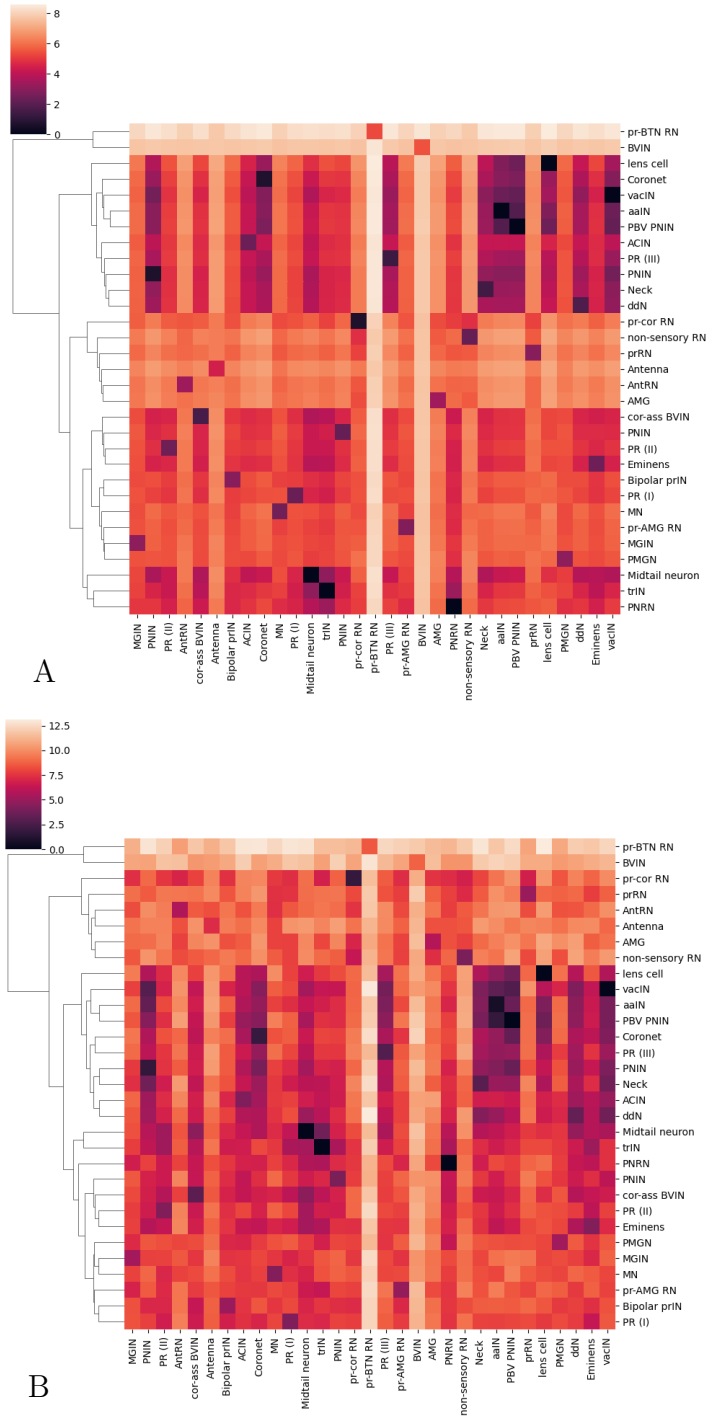

Figure 4.22: Hierarchical clustering of neurons of animal 1 (A) and animal 2 (B).

to demonstrate the effectiveness of our method. Specifically, our analysis of the *Ciona* dataset demonstrates that shape could be used as a significant feature for classifying neuronal types.

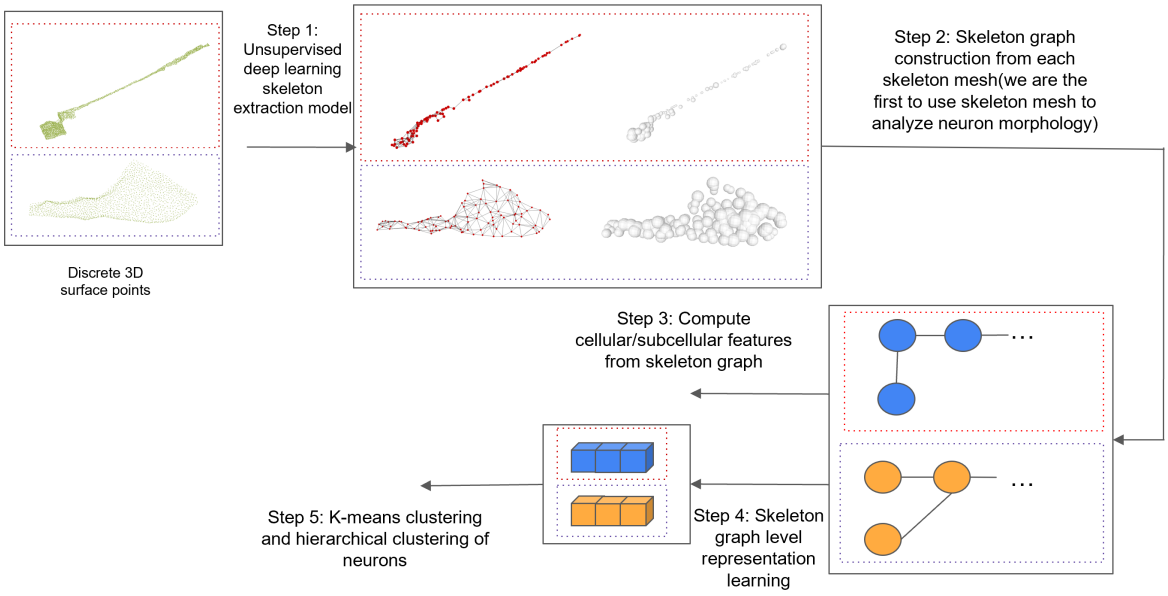

Figure 4.23: Summary of our proposed neuron morphology analysis pipeline

Chapter 5

Conclusions and Future Work

> The past is a place of reference, not a place of residence; the past is a place of learning, not a place of living.
>
> *Roy T. Bennett*

The primary goal of this dissertation is to develop automatic analysis tools for 3D biomedical images. Specifically, we consider two microscopy cell image analysis problems. The first problem is 3D pavement cell growth analysis problem. For this problem, we need to accurately localize and track cell walls and sub-cellar features along the cell walls. The second problem is to analyze the 3D structure of the neuroins, to create a robust 3D shape representation that could be used for semantic classification and relating structure to neuronal function.

To solve the first problem, we propose novel methods for segmentation, tracking, and computing sub-cellular features. We propose a 3D segmentation pipeline for membrane tagged confocal cell images. It includes a rotation equivariance 3D UNet to get the

probability map of cell boundary, 3D watershed for initial segmentation of cells, and conditional random field (CRF) for refinement of cell walls. The novel rotation equivariance 3D UNet has a new 3D rotation equivariance convolutional layers to make it invariant to 3D rotations of input image data. We are also the first to apply CRF refinement on cell segmentation task to refine the boundary. After the segmentation, we build an adjacency graph on the segmentation mask. A simple but effective computational method is proposed to extract sub-cellular features such as three cell wall junctions and anticlinal wall segments. A new efficient but accurate cell tracking method is also developed on the adjacency graph. The proposed method can be used to analyze nuclei tagged data. This tracking method won 3rd place in the 2021 ISBI cell tracking challenge [13].

For solving the neuron morphology analysis problem, we propose a skeleton mesh extraction method from surface point cloud of neurons. Sub-cellular features can then be computed from this skeleton mesh and a skeleton graph embedding method is proposed for neuron classification. It is the first method to apply skeleton mesh for neuron morphology analysis task. We use properties of medial axis transform (MAT) to further improve the skeleton mesh extraction method performance compared to the state-of-the-art method [8]. We are also the first one to propose the sub-cellular feature extraction and the skeleton graph embedding method for neuron classification.

5.1 Future Directions

Time-lapse 3D microscopy cell image analysis

In Chapter 3 we proposed an end-to-end workflow for segmenting and tracking cells in 3D. The cell segmentation is first performed on the individual 3D volumes and then the segmented cells are tracked. Since segmentation is independent of the tracking step, this might result in different number of cells at each time point and post-processing would be

needed to separate the false positive detections from the true ones. A hjoint segmentation-tracking fraework would be more effective. Currently, there are some methods that do joint segmentation and tracking for 2D natural images using Transformer [76] or siamese network [77]. Those methods are good starting points to build a joint segmentation and tracking pipeline for 3D cell images.

Another opportunity in this problem is to extend our 3D time-lapse cell analysis pipeline into multiple scale multiscale analysis pipeline. Biologists sometimes view cell walls at multiple microscopy image scales to analyze how the cell walls grow because some sub-cellular features are best to analyze under certain spatial scale. It is interesting to develop a method to fuse information from multiple scales for automatic analysis.

Neuron Morphology Analysis

For the skeleton mesh extraction method, we need to fix number of points for the input point cloud and output the same number of skeleton points. Sometimes, fewer skeleton points are needed to make the skeleton mesh more compact and abstract. However, sometimes, it is good to have more skeleton points to make the skeleton mesh more similar to the original shape. Therefore, it is helpful to have adaptive skeleton mesh extraction methods that can output various number of skeleton points based on the input shape.

Another opportunity is on the application side of the skeleton mesh. We are the first to introduce the skeleton mesh concept for the neuron morphology analysis. However, skeleton mesh can represent more general shapes. It is interesting to see if this skeleton mesh representation is useful for more general cell shapes such as the pavement cell analysis described previously.

Bibliography

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et. al.*, *Shapenet: An information-rich 3d model repository*, *arXiv preprint arXiv:1512.03012* (2015).

[2] S. A. Belteton, W. Li, M. Yanagisawa, F. A. Hatam, M. I. Quinn, M. K. Szymanski, M. W. Marley, J. A. Turner, and D. B. Szymanski, *Real-time conversion of tissue-scale mechanical forces into an interdigitated growth pattern*, *Nature Plants* **7** (2021), no. 6 826–841.

[3] T.-C. Wu, S. A. Belteton, J. Pack, D. B. Szymanski, and D. M. Umulis, *Lobefinder: A convex hull-based method for quantitative boundary analyses of lobed plant cells*, *Plant Physiology* **171** (2016), no. 4 2331–2342, [http://www.plantphysiol.org/content/171/4/2331.full.pdf].

[4] B. Zhong, K.-K. Ma, and W. Liao, *Scale-space behavior of planar-curve corners*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31** (2008), no. 8 1517–1524.

[5] K. Ryan, Z. Lu, and I. A. Meinertzhagen, *The cns connectome of a tadpole larva of ciona intestinalis (l.) highlights sidedness in the brain of a chordate sibling*, *Elife* **5** (2016) e16962.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, *Advances in neural information processing systems* **30** (2017).

[7] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or, *Deep points consolidation*, *ACM Transactions on Graphics (ToG)* **34** (2015), no. 6 1–13.

[8] C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang, *Point2skeleton: Learning skeletal representations from point clouds*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4277–4286, 2021.

[9] S. A. Belteton, M. G. Sawchuk, B. S. Donohoe, E. Scarpella, and D. B. Szymanski, *Reassessing the roles of pin proteins and anticlinal microtubules during pavement cell morphogenesis*, *Plant Physiology* **176** (2018), no. 1 432–449.

[10] S. H. Jonsson, L. Willis, and Y. Refahi, *Research data supporting cell size and growth regulation in the arabidopsis thaliana apical stem cell niche, [Dataset]* (2017).

[11] J. I. Murray, Z. Bao, T. J. Boyle, M. E. Boeck, B. L. Mericle, T. J. Nicholas, Z. Zhao, M. J. Sandel, and R. H. Waterston, *Automated analysis of embryonic gene expression with cellular resolution in c. elegans*, *Nature methods* **5** (2008), no. 8 703–709.

[12] V. Ulman, M. Maška, K. E. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, *et. al.*, *An objective comparison of cell-tracking algorithms*, *Nature methods* **14** (2017), no. 12 1141–1152.

[13] S. D. Maška M, Ulman V *et. al.*, *A benchmark for comparison of cell tracking algorithms*, *Bioinformatics* **30** (02, 2014) 1609–1617, [https://academic.oup.com/bioinformatics/article-pdf/30/11/1609/17143058/btu080.pdf].

[14] C. U. Lowe, *The relation between cell structure and cell function*, *Pediatrics* **26** (1960), no. 3 454–458.

[15] J. L. Collins, B. van Knippenberg, K. Ding, and A. V. Kofman, *Time-lapse microscopy*, in *Cell Culture* (R. A. Mehanna, ed.), ch. 3. IntechOpen, Rijeka, 2018.

[16] B. W. Graf and S. A. Boppart, *Imaging and analysis of three-dimensional cell culture models*, in *Live cell imaging*, pp. 211–227. Springer, 2010.

[17] R. V. Vőfély, J. Gallagher, G. D. Pisano, M. Bartlett, and S. A. Braybrook, *Of puzzles and pavements: a quantitative exploration of leaf epidermal cell shape*, *New Phytologist* **221** (2019), no. 1 540–552.

[18] B. Möller, Y. Poeschl, R. Plötner, and K. Bürstenbinder, *Pacequant: a tool for high-throughput quantification of pavement cell shape characteristics*, *Plant physiology* **175** (2017), no. 3 998–1017.

[19] M. Halavi, K. A. Hamilton, R. Parekh, and G. A. Ascoli, *Digital reconstructions of neuronal morphology: three decades of research trends*, *Frontiers in neuroscience* **6** (2012) 49.

[20] S. K. Schmitz, J. J. Hjorth, R. M. Joemai, R. Wijntjes, S. Eijgenraam, P. de Bruijn, C. Georgiou, A. P. de Jong, A. van Ooyen, M. Verhage, *et. al.*, *Automated analysis of neuronal morphology, synapse number and synaptic recruitment*, *Journal of neuroscience methods* **195** (2011), no. 2 185–193.

[21] L. Billeci, C. Magliaro, G. Pioggia, and A. Ahluwalia, *Neuronmorphological analysis tool: open-source software for quantitative morphometrics*, *Frontiers in neuroinformatics* **7** (2013) 2.

[22] M. Costa, J. D. Manton, A. D. Ostrovsky, S. Prohaska, and G. S. Jefferis, *Nblast: rapid, sensitive comparison of neuronal structure and construction of neuron family databases*, *Neuron* **91** (2016), no. 2 293–311.

[23] M. Abdellah, J. Hernando, S. Eilemann, S. Lapere, N. Antille, H. Markram, and F. Schürmann, *Neuromorphovis: a collaborative framework for analysis and visualization of neuronal morphology skeletons reconstructed from microscopy stacks*, *Bioinformatics* **34** (2018), no. 13 i574–i582.

[24] S. Li, T. Quan, C. Xu, Q. Huang, H. Kang, Y. Chen, A. Li, L. Fu, Q. Luo, H. Gong, *et. al.*, *Optimization of traced neuron skeleton using lasso-based model*, *Frontiers in neuroanatomy* **13** (2019) 18.

[25] O. Panichev and A. Voloshyna, *U-net based convolutional neural network for skeleton extraction*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[26] L. K. Scheffer, C. S. Xu, M. Januszewski, Z. Lu, S.-y. Takemura, K. J. Hayworth, G. B. Huang, K. Shinomiya, J. Maitlin-Shepard, S. Berg, *et. al.*, *A connectome and analysis of the adult drosophila central brain*, *eLife* **9** (2020) e57443.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[28] G. A. Ascoli, D. E. Donohue, and M. Halavi, *Neuromorpho. org: a central resource for neuronal morphologies*, *Journal of Neuroscience* **27** (2007), no. 35 9247–9251.

[29] T. J. Collins, *Imagej for microscopy*, *Biotechniques* **43** (2007), no. S1 S25–S30.

[30] P. A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. C. Gee, and G. Gerig, *User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability*, *Neuroimage* **31** (2006), no. 3 1116–1128.

[31] K. M. Harris, J. Spacek, M. E. Bell, P. H. Parker, L. F. Lindsey, A. D. Baden, J. T. Vogelstein, and R. Burns, *A resource from 3d electron microscopy of hippocampal neuropil for user training and tool development*, *Scientific data* **2** (2015), no. 1 1–19.

[32] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. J. Douglas, *Trakem2 software for neural circuit reconstruction*, *PloS one* **7** (2012), no. 6 e38011.

[33] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, *et. al.*, *Meshlab: an open-source mesh processing tool.*, in *Eurographics Italian chapter conference*, vol. 2008, pp. 129–136, Salerno, Italy, 2008.

[34] K. Kvilekval, D. Fedorov, B. Obara, A. Singh, and B. Manjunath, *Bisque: a platform for bioimage analysis and management*, *Bioinformatics* **26** (2010), no. 4 544–552.

[35] A. Nwaneshiudu, C. Kuschal, F. H. Sakamoto, R. R. Anderson, K. Schwarzenberger, and R. C. Young, *Introduction to confocal microscopy*, *Journal of Investigative Dermatology* **132** (2012), no. 12 1–5.

[36] X.-G. Zhu, S. P. Long, and D. R. Ort, *Improving photosynthetic efficiency for greater yield*, *Annual review of plant biology* **61** (2010) 235–261.

[37] S. Savaldi-Goldstein, C. Peto, and J. Chory, *The epidermis both drives and restricts plant shoot growth*, *Nature* **446** (2007), no. 7132 199–202.

[38] J. Jiang, P.-Y. Kao, S. A. Belteton, D. B. Szymanski, and B. S. Manjunath, *Accurate 3d cell segmentation using deep features and crf refinement*, in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1555–1559, Sep., 2019.

[39] J. Stegmaier, F. Amat, W. C. Lemon, K. McDole, Y. Wan, G. Teodoro, R. Mikut, and P. J. Keller, *Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos*, *Developmental cell* **36** (2016), no. 2 225–240.

[40] K. R. Mosaliganti, R. R. Noche, F. Xiong, I. A. Swinburne, and S. G. Megason, *Acme: automated cell morphology extractor for comprehensive reconstruction of cell membranes*, *PLoS computational biology* **8** (2012), no. 12 e1002780.

[41] R. Fernandez, P. Das, V. Mirabet, E. Moscardi, J. Traas, J.-L. Verdeil, G. Malandain, and C. Godin, *Imaging plant growth in 4d: robust tissue reconstruction and lineaging at cell resolution*, *Nature methods* **7** (2010), no. 7 547–553.

[42] J. Stegmaier, T. V. Spina, A. X. Falcão, A. Bartschat, R. Mikut, E. Meyerowitz, and A. Cunha, *Cell segmentation in 3d confocal images using supervoxel merge-forests with cnn-based hypothesis selection*, in *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pp. 382–386, IEEE, 2018.

[43] H. Tsuda and K. Hotta, *Cell image segmentation by integrating pix2pixs for each class*, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June, 2019.

[44] M. Majurski, P. Manescu, S. Padi, N. Schaub, N. Hotaling, C. Simon Jr, and P. Bajcsy, *Cell image segmentation using generative adversarial networks, transfer learning, and augmentations*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[45] D. L. Delibaltov, U. Gaur, J. Kim, M. Kourakis, E. Newman-Smith, W. Smith, S. A. Belteton, D. B. Szymanski, and B. Manjunath, *Cellect: cell evolution capturing tool*, *BMC bioinformatics* **17** (2016), no. 1 88.

[46] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, *Cellpose: a generalist algorithm for cellular segmentation*, *Nature methods* **18** (2021), no. 1 100–106.

[47] W. Han, A. M. Cheung, M. J. Yaffe, and A. L. Martel, *Cell segmentation for immunofluorescence multiplexed images using two-stage domain adaptation and weakly labeled data for pre-training*, *Scientific Reports* **12** (2022), no. 1 1–14.

[48] D. E. Hernandez, S. W. Chen, E. E. Hunter, E. B. Steager, and V. Kumar, *Cell tracking with deep learning and the viterbi algorithm*, in *2018 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, pp. 1–6, July, 2018.

[49] Z. Zhou, F. Wang, W. Xi, H. Chen, P. Gao, and C. He, *Joint multi-frame detection and segmentation for multi-cell tracking*, in *Image and Graphics*, (Cham), pp. 435–446, Springer International Publishing, 2019.

[50] Y. Chen, H. Sun, H. Yang, and X. Pan, *Level set method of cell image segmentation based on combinations of edge, region and prior information*, in *Systems and Informatics (ICSAI), 2017 4th International Conference on*, pp. 1245–1249, IEEE, 2017.

[51] M. Chen, *Cell tracking in time-lapse microscopy image sequences*, in *Computer Vision for Microscopy Image Analysis*, pp. 101–129. Elsevier, 2021.

[52] J. Jiang, A. Khan, S. Shailja, S. A. Belteton, M. Goebel, D. B. Szymanski, and B. Manjunath, *Deep learning enabled time-lapse 3d cell analysis*, *arXiv preprint arXiv:2208.07997* (2022).

[53] S. Shailja, J. Jiang, and B. Manjunath, *Semi supervised segmentation and graph-based tracking of 3d nuclei in time-lapse microscopy*, in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 385–389, IEEE, 2021.

[54] B. Chidester, T.-V. Ton, M.-T. Tran, J. Ma, and M. N. Do, *Enhanced rotation-equivariant u-net for nuclear segmentation*, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June, 2019.

[55] Y. Wu and K. He, *Group normalization*, in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[56] P. Soille, *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[57] P. Krähenbühl and V. Koltun, *Efficient inference in fully connected crfs with gaussian edge potentials*, *Advances in neural information processing systems* **24** (2011).

[58] K. E. Magnusson, J. Jaldén, P. M. Gilbert, and H. M. Blau, *Global linking of cell tracks using the Viterbi algorithm*, *IEEE transactions on medical imaging* **34** (2014), no. 4 911–929.

[59] T. Eiter and H. Mannila, *Computing discrete fréchet distance*, .

[60] K. Löffler and T. Scherr, *KIT-Sch-GE*, *Cell Tracking Challenge* (2020).

[61] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä, *Cell tracking via proposal generation and selection*, *ArXiv* **abs/1705.03386** (2017).

[62] M. I. Latypov, A. Khan, C. A. Lang, K. Kvilekval, A. T. Polonsky, M. P. Echlin, I. J. Beyerlein, B. Manjunath, and T. M. Pollock, *Bisque for 3d materials science in the cloud: microstructure–property linkages*, *Integrating Materials and Manufacturing Innovation* **8** (2019), no. 1 52–65.

[63] S.-Y. Ho, C.-Y. Chao, H.-L. Huang, T.-W. Chiu, P. Charoenkwan, and E. Hwang, *Neurphologyj: an automatic neuronal morphology quantification method and its application in pharmacological discovery*, *BMC bioinformatics* **12** (2011), no. 1 1–18.

[64] S. Jiang, Z. Pan, Z. Feng, Y. Guan, M. Ren, Z. Ding, S. Chen, H. Gong, Q. Luo, and A. Li, *Skeleton optimization of neuronal morphology based on three-dimensional shape restrictions*, *BMC bioinformatics* **21** (2020), no. 1 1–16.

[65] P. K. Saha, Y. Xu, H. Duan, A. Heiner, and G. Liang, *Volumetric topological analysis: a novel approach for trabecular bone classification on the continuum between plates and rods*, *IEEE transactions on medical imaging* **29** (2010), no. 11 1821–1838.

[66] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, *Building skeleton models via 3-d medial surface axis thinning algorithms*, *CVGIP: Graphical Models and Image Processing* **56** (1994), no. 6 462–478.

[67] S. M. Pizer, J. Hong, J. Vicory, Z. Liu, J. Marron, H.-y. Choi, J. Damon, S. Jung, B. Paniagua, J. Schulz, *et. al.*, *Object shape representation via skeletal models (s-reps) and statistical analysis*, in *Riemannian Geometric Statistics in Medical Image Analysis*, pp. 233–271. Elsevier, 2020.

[68] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, *Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization*, in *International Conference on Learning Representations*, 2019.

[69] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[70] Q.-Y. Zhou, J. Park, and V. Koltun, *Open3d: A modern library for 3d data processing*, *arXiv preprint arXiv:1801.09847* (2018).

[71] T. N. Kipf and M. Welling, *Variational graph auto-encoders*, *arXiv preprint arXiv:1611.07308* (2016).

[72] P. V. Tran, *Learning to make predictions on graphs with autoencoders*, in *2018 IEEE 5th international conference on data science and advanced analytics (DSAA)*, pp. 237–245, IEEE, 2018.

[73] C. Yuksel, *Sample elimination for generating poisson disk sample sets*, in *Computer Graphics Forum*, vol. 34, pp. 25–32, Wiley Online Library, 2015.

[74] M. Kazhdan and H. Hoppe, *Screened poisson surface reconstruction*, *ACM Transactions on Graphics (ToG)* **32** (2013), no. 3 1–13.

[75] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, *graph2vec: Learning distributed representations of graphs*, *arXiv preprint arXiv:1707.05005* (2017).

[76] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, *Trackformer: Multi-object tracking with transformers*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8844–8854, 2022.

[77] Y. Cui, D. Guo, Y. Shao, Z. Wang, C. Shen, L. Zhang, and S. Chen, *Joint classification and regression for visual tracking with fully convolutional siamese networks*, *International Journal of Computer Vision* **130** (2022), no. 2 550–566.