Reconstruction of 3D Neuronal Structures

A Thesis

Submitted For the Degree of

Master of Science (Engineering)

in the Faculty of Engineering

by

Kanuj Kumar

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

February 2013

To

My parents and my brother

You are the roots on which I have sustained myself more often than you believe.

Acknowledgements

This thesis would not have been possible without the friendship, support and contributions from many people. I would like to acknowledge their support and assistance.

Many of the ideas presented in this thesis come from fruitful conversations and joint work done with my research advisor Dr Vijay Natarajan. It is under his supervision and advice that I could accomplish the work that lead to this thesis. Even during his busiest times, he was so kind to answer my questions thoroughly and efficiently. My heartfelt appreciation and deepest thanks for his patient encouragement and support during the early part of my studies.

I am also thankful to Prof. S. K. Sikdar, for his valuable feedback and for providing the datasets to base my research on. Along the same line I would like to thank his students Nirnath and Kalyan for performing the experiments to generate the datasets.

I would like to thank my lab mates Dilip and Sandeep, as it was their efforts that helped me debug various parts of my code. I learned a lot of things from them in the time they spent doing so. Their help and friendship is gratefully acknowledged.

I would also like to express my sincere gratitude to the chairman, Prof. Y. Narahari for the support and encouragement given to me.

Abstract

The three-dimensional reconstruction of neurons is crucial for accurate neurobiological analysis. However, biologists have to rely on time-consuming manual or semi-manual methods to process the laser-scanned microscopy images into geometrical models that in turn would lead to the creation of libraries of neuronal morphologies. Current techniques for digitizing neuronal morphology in 3D entail manual tracing using commercial packages such as NeuroZoom and NeuroLucida. Such methods introduce systematic inaccuracies depending upon the individual performing the tracing, thus, necessitating the need of fully-automated reconstruction methods for neuronal structures. Despite the recent advancements, the automation of reconstruction process either doesn't exhibit robustness against noise of microscopy images or fail to capture precise dendritic structures. Due to these conflicting requirements, a perfect reconstruction is impossible making some user intervention necessary. In this thesis, we present a framework for reconstruction of neurons that is substantiated through the user validation of output from each process within framework. The adaptive approaches outlined in this thesis were put to use with the goal of enabling automated and flexible outputs to ensure a high quality reconstruction.

Our framework also consists of methods to handle discontinuities in the microscopy image to produce a connected geometrical model and is capable of working on varying contrast and size of data.

Contents

Keywords

Neuron reconstruction. Contour Extraction. Beta Connection.

Chapter 1

Introduction

The nervous system of almost all animals consists of nerve cells, known as neurons. Different information propagation and processing occurs as results of signal processing within and among neurons. The morphology of neurons plays a decisive role in information processing carried out by the entire nervous system. In this context, 3D reconstruction of neuron morphology is of fundamental interest for the analysis and the understanding of neurons functional characteristics. It significantly heightens the ability to process geometrical information about neurons by facilitating visualization of the anatomical relationships of neurons and their patterns of dendritic and axonal contact within nervous tissue. It can also provide anatomical data for construction of electrical and biochemical circuit models of neuron function that can be used in further computer simulations.

Because of the complex morphology of neuronal cells, reconstruction of a neuronal structures using a semiautomatic or manual systems is a time consuming task. The challenges of analyzing optical neuron images typically include low signal-to-noise ratio, ambiguities regarding the branching or crossing of neurites and the linking of fragmented neurite segments. These challenges makes it virtually impossible to develop fully automatic reconstruction techniques for our purpose. The precision of automatic methods is also limited by the quality of the reconstruction algorithm used and usually suffers from various algorithmic constraints. Thus, the goal of decreasing the necessary expense of user interaction often acts contrary to ensure the accuracy and the topological correctness of the models.

Due to the low accuracy and incomplete results of automatic procedures that are now available, geometric reconstructions are commonly done manually with programs such as NeuroLucida (MicroBrightField) [1] or Neurozoom[2]. The accuracy of these manual reconstructions, however is strongly dependent on individual data interpretation to estimate mid-lines and diameters of neuron structures in the images. While a few techniques of varying degrees of automation have been suggested for digitizing neuronal morphology from laser-scanning microscopy images, the limited size of the data sets that can be handled, and the general requirement for extensive manual editing renders these techniques impractical in most cases, especially for reconstructing entire neurons at high resolution. A primary goal of our approach is the development of an automated system for accurate 3D reconstruction of neuron morphology, capable of handling input data independent of the scale of the images. As we place emphasis on our framework to create flexible models, experience of the human anatomist becomes necessary to validate the output of process in the framework. However, in contrast to standard manual reconstruction techniques, we kept the requirement of minimal user assistance to reconstruct the model.

1.1 Problem Definition

We define a reconstruction of a neuron as a process that generates a 3D model and fulfills the following requirements: (i) sufficient accuracy of the reconstructed model can be accomplished, (ii) topological constraints based on the assumption of a tree-like structure must be fulfilled, (iii) the amount of the user's time and effort should be reasonably small, (iv) allows user intervention to correct any unsatisfactory outputs, and (v) should be fast and make efficient use of memory.

[1] Glaser and Glaser, Neuron imaging with NeuroLucida, 1990
[2] Neurome, Inc

1.2 Why the problem is Challenging?

The major challenges are involved in the segmentation of the structure from the microscopy images due to noise imposed by imaging conditions. These challenges include: (i) variable contrast of structure in images due to uneven dye distribution within the cell, (ii) intensity decay along the z-axis due to microscopic resolution, (iii) partial volume effect dependent on point spread function (PSF) of the microscope, and (iv) insufficient imaging resolution due to low signal-to-noise ratio (SNR). These challenges cause segmentation errors, resulting in self-intersecting and fragmented structures. These inferred errors then restrict to maintain the precision and the topological properties for the reconstructed structure.

Most of the existing methods can handle these imperfections by introducing global prior information, such as where a neurite segment starts and terminates, however, manual incorporation of such global information can be very time consuming. Automation is thus a better alternative. However, tools capable of resolving neuronal morphology on both local and global scales, and with sufficient automation, are limited by the skeletonisation methods used, and by quantization errors arising from insufficient imaging resolution.

Most of the prior work on segmentation process (described in chapter 3) rely on filters optimized for images conforming either to the method used to obtain the data or the scale of the data. Thus the demand for the reconstruction tool capable of handling structures independent of the scale is not answered by existing automated methods.

1.3 The Framework

We designed a framework that addresses the problem defined in section 1.1 and relieve the user from manual tracing of the structures in the images, thereby eliminating subjective estimations and imprecision due to the users haste or exhaustion. The framework and the pipeline for reconstruction of neuron structures are shown in Figure 1.1. It requires minimal user interaction for noise removal or to extract different contours components from images. However, the reconstruction process is executed independent of the user's

knowledge. Each process in the framework requires an input parameter along with the data to control the resultant output. The fact that the parameters of the framework can be adapted depending on the represented neuron structure or noise in the image, shows its adaptability to generate an adequate segmentation of structures of different sizes.

In our framework, the task of reconstruction of the neuron structure from image data is

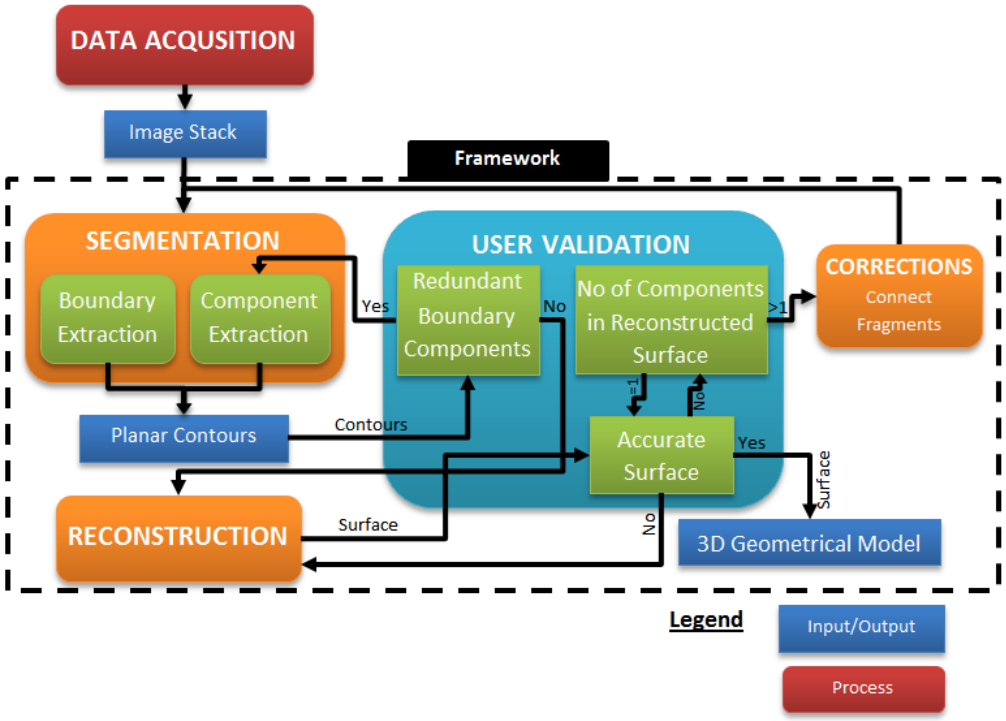


Figure 1.1: The framework for reconstruction

split into two modules. The first processing module is the segmentation, which is used to extract representative boundary of neurite structures from the given image data. Surface is then generated in the reconstruction module from the segmented structures. In our framework, an automated pipeline can also be deduced by initializing the parameters, which can be later refined by performing parameter selection analysis (explained in chapter 4).

Segmentation The segmentation module starts with computation of boundary polygons based on locally adapted thresholds for an image. A connected 3D component is

then formed to filter the noisy structures that employs global connectivity among boundary polygons. The choice of input parameters is assisted via immediate visualization of results from the process.

Reconstruction The goal of the reconstruction module is to find a best surface consistent with the extracted polygons. In our framework, the reconstruction module attempts this construction of triangulation from the boundary polygons. This triangulation is extended further to create correspondence between polygons of adjacent slices. The general principles that guides the volumetric reconstruction algorithm are described in section 2.5.1.

Corrections The goal of the correction module is to produce a connected structure when the reconstruction module is unable to do the same. In order to achieve this, we build a set of minimum-spanning-tree (MST) over the polygons in different connected components and then establish a link between polygons if the corresponding pair is connected by an edge in the MST.

1.4 Thesis Organization

Chapter 2 covers the definitions of the various terms and algorithms. Advantages and limitations of relevant algorithms for neuron reconstruction is discussed in Chapter 3. In addition to describing the overall framework, theory and implementation of each concept is described in detail in Chapter 4. Chapter 5 demonstrates the results and performance of the framework. Chapter 6 concludes the reconstruction framework.

1.5 Our Contributions

Our principal achievements toward a fast and automated system for reconstruction of neuronal structures are

- minimal user time investment by use of automatic methods, while the user can keep control over the algorithm to generate a flexible reconstruction.
- independence from imaging modalities, making it capable of working with varying or discontinuous background intensities, contrast, or resolution in images.
- a novel approach to handle discontinuities in structures often present in microscopy imagery to produce a connected model.
- an accelerated framework capable of processing adjacent pair of images in parallel.

Chapter 2

Basic Concepts

2.1 Neuron Morphology

There are different types and sizes of neurons that are classified by their structural and functional characteristics. Generally, neurons consist of three main parts:

1. *Soma* is the central part of neuron and has a spherical shape with a diameter varying from $4\mu m$ to $100\mu m$.

2. *Axon* is a thin cable-like long extension of the neuron with tens of thousands times the diameter of the soma. Many neurons have only one axon, but this axon may branch out.

3. *Dendrite* is a branched projection of neuron extending for hundreds of micrometres, whose overall shape and structure is referred to as a dendritic tree. towards soma.

2.2 Confocal Laser-scanning Microscopy (CLSM)

Confocal microscopy is an optical imaging technique that can scan a stained specimen with high resolution. The key feature of this imaging technique is its ability to acquire in-focus images from selected depths, a process known as optical sectioning. The basic principle and working of the confocal microscopy imaging technique is explained in [19]. In this technique, back-scattered light is gathered confocally by using a spatial pinhole

in an optically conjugate plane in front of a detector. Through this process most of the photons coming from out of focus planes are filtered and only the light within the focal plane can be detected. After scanning a two-dimensional layer of the specimen, the laser beam focuses one layer deeper to scan the next 2D layer. In this way, the confocal microscope constructs a 3D image stack of the specimen. The accuracy of the acquired image depends on the raster scan, i.e. the spatial steps between different focuses. The shorter the steps are, the bigger the resolution of the recorded image. Each pixel of the image represents a focused point in the scanned specimen. Images acquired through this technique are of sufficient resolution to enable the three-dimensional reconstructions of complex structures such as neurons.

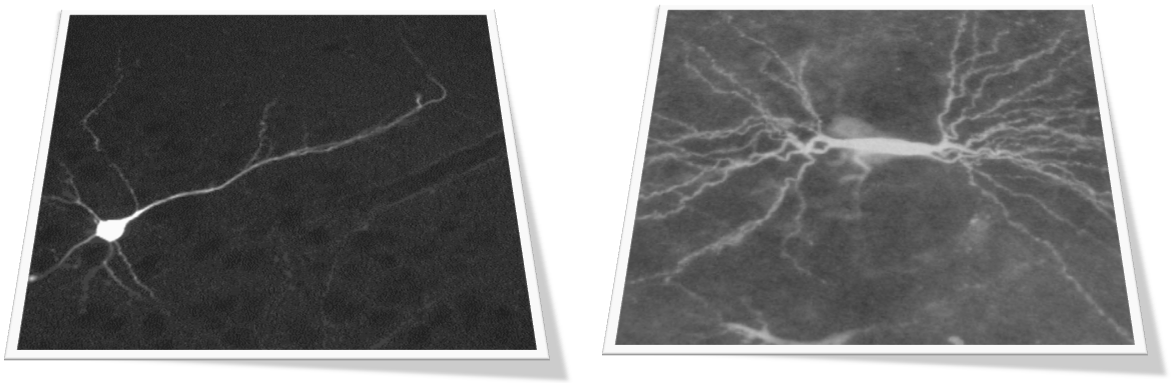
2.3 Dataset

The dataset is composed of stack of parallel 2D images acquired through CLSM or other imaging modalities. Each 2D image is considered as a 2D matrix, where each component of the matrix represents the intensity value of a pixel. In this context, the image stack is a function I(x,y,z) that maps voxel coordinates onto their intensity values. For the remainder of this thesis, the z-axis will refer to the axis parallel to the optical axis, while the x-y plane will refer to the imaging plane.

To demonstrate the performance of the framework, we have acquired following datasets recorded with confocal microscopy where each dataset consists of artifacts or noise inherent from the mechanism of how it was obtained. We are thankful to Prof S K Sikdar (MBU, IISc) and his students for performing the experiments to generate the datasets.

Dataset I [Figure 2.1a]: This dataset shows the subicular pyramidal neuron cells from rat hippocampi. Images were acquired with through CLSM loaded with calcium stained dye. The dataset consists of 47 steps with voxel resolution of 0.65 μm in the x-y axis and 0.5 μm in the z axis. Excitation wavelength was set to 488 nm with index of refraction corresponding to that of the water. The depth of image is 8 bits/pixel with resolution of 512x512 in x-y scale.

Dataset II [Figure 2.1b]: This dataset shows the CA3 pyramidal neuron from the rat hippocampi. Images were acquired with CLSM loaded with Diaminobenzidine dye. The dataset consists of 36 steps with voxel resolution of 1.0 μm in the z axis. In this dataset, dendrites project in all directions from the soma, generally having wider apical segments that taper with distance from the soma. The axon is not visualized in this image.



(a) Dataset I (b) Dataset II

Figure 2.1: CLSM scan of (a) subicular pyramidal neuron (b) CA3 pyramidal neuron

The major challenges posed by these datasets are:

- diffraction effects causing a spread of a point object in the image characterized by the point spread function.
- variable contrast due to uneven dye distribution within the cell causing apparent discontinuity in structures.
- many features of interest (such as thin dendrites) are at the limit of imaging resolution.

2.4 Segmentation

2.4.1 Thresholding

Thresholding is one of the simplest and least computationally intensive technique for image segmentation that considers pixels intensity as the measure to generate a binary image. The pixel value of the resulting binary image either corresponds to a background or foreground (represented structure raw images). The key parameter in the thresholding process is the choice of the threshold value, which could be done manually or through automatic methods. Based on the survey of methods given in [53], the algorithms for thresholding are classified as:

Histogram based methods use the peaks, valleys and curvatures of the histogram to define the threshold. For example, Sezan[47] carries out the peak analysis by convolving the histogram function with a smoothing and differencing kernel and reduces the histogram to a two-lobe function. The threshold is then selected from the histogram lobe. Following the similar pattern, Carlotto[11] and Olivo[39] carry out the multi-scale analysis of the probability function of histogram. The threshold is defined as the valley point the first peak in the smoothed histogram. This threshold position is successively refined over the scales starting from the coarsest resolution.

Clustering based methods cluster the gray-level intensities as background and foreground or alternately model them as a mixture of two Gaussians. Riddler[43] advanced one of the first iterative schemes based on two-class Gaussian mixture models. At every iteration a new threshold is established using the average of the foreground and background class means. The iterations terminate when the changes between successive thresholds become sufficiently small. Otsu[40] suggested minimizing of the weighted sum of within-class variances of the foreground and background pixels to establish an optimum threshold. This method gives satisfactory results when the numbers of pixels in each class are close to each other.

Entropy based methods exploit the entropy of the distribution of the gray levels in a images. Kapur et. al.[29] consider the image foreground and background as two different signal sources, and choose an optimal threshold when the sum of the two class entropies reaches its maximum.

Object attribute based methods search for a measure of similarity (such as fuzzy shape similarity, edge coincidence) between the gray-level intensity and the binarized images. Leung and Lam[32] define the thresholding problem as the change in the uncertainty of an observation on specification of the foreground and background classes. This reduces the class uncertainty of a pixel, and the information gain is measured by a predefined function. The optimum threshold is then established as generating a segmentation map that, in turn, minimizes the average residual uncertainty about which class a pixel belongs to after the segmented image has been observed.

Spatial methods use higher-order probability distribution or correlation between pixels neighbourhood. Pal[41] suggest the use of concurrences probability of the gray values for threshold selection, realizing that two images with identical histograms can have different n^{th} order entropies due to their spatial structure.

Local methods adapt the threshold value on each pixel to the local image characteristics such as range, variance, or surface-fitting parameters of the pixel neighborhood. White and Rohrer[56] compare the gray value of the pixel with the average of the gray values in some predefined neighborhood of the pixel. If the pixel is significantly darker than the average, it is denoted as foreground; otherwise, it is classified as background.

2.4.2 Morphological Operators for Binary Images

The theory of different morphological operators explained here, has been adapted from [23]. In binary morphology, an image is viewed as a subset of an Euclidean space $\mathbb{R}^d$ or the integer grid $\mathbb{Z}^d$, for some dimension d.

Structuring element The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple "robe" is called structuring element and is itself a binary image (i.e. a subset of the space or grid). For example, let $E = \mathbb{R}^2$; then the structuring element B is an open disk of radius r, centered at the origin. Let E be a Euclidean space or an integer grid, and A a binary image in E.

Erosion The erosion of the binary image A by the structuring element B is defined by: $A \ominus B = \{z \in E | B_z \subseteq A\}$, where B_z is the translation of B by the vector z, i.e., $B_z = \{b + z | b \in B\}, \forall z \in E$.
When the structuring element B center is located on the origin of E, then the erosion of A by B can be understood as the locus of points reached by the center of B when B moves inside A. For example, the erosion of a square of side 10, centered at the origin, by a disc of radius 2 is a square of side 6 centered at the origin (Figure 2.2). The erosion of A by B is also given by the expression: $A \ominus B = \bigcap_{b \in B} A_{-b}$.

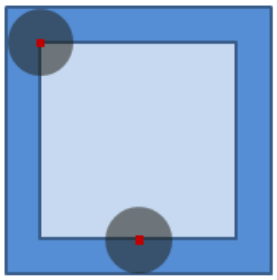


Figure 2.2: Erosion of the dark-blue square by a disk, resulting in the light-blue square

Dilation The dilation of the binary image A by the structuring element B is defined by: $A \oplus B = \bigcup_{b \in B} A_b$ Since the dilation is commutative, it can also be given by: $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$ If structuring element B has a center on the origin, as before, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. In the above example, the dilation of the square of side 10 by the disk of radius 2 is a square of side 14, with rounded corners, centered at the origin. The radius of the rounded corners is 2 (Figure 2.3). The dilation can also be obtained by: $A \oplus B = \{z \in E | (B^s)_z \cap A \neq \emptyset\}$, where $B^s = \{x \in E | -x \in B\}$, i.e. symmetric of B.

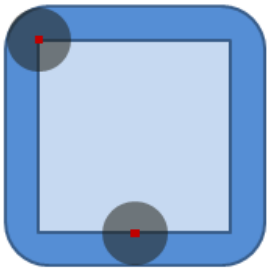


Figure 2.3: Dilation of the dark-blue square by a disk, resulting in the light-blue square with rounded corners

Opening The opening of A by B is obtained by the erosion of A by B, followed by dilation of the resulting image by B: $A \circ B = (A \ominus B) \oplus B$. It is also given by : $A \circ B = \subseteq_{B_x} A \bigcup B_x$, which means that it is the locus of translations of the structuring element B inside the image A. In the above example, the opening is a square of side 10 with rounded corners, where the corner radius is 2 (Figure 2.4).

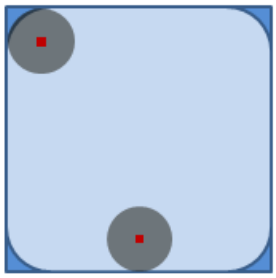


Figure 2.4: Opening of the dark-blue square by a disk, resulting in the light-blue square with round corners

Closing The closing of A by B is obtained by the dilation of A by B, followed by erosion of the resulting structure by B: $A \bullet B = (A \oplus B) \ominus B$. It can also be obtained by: $A \bullet B = (A^c \circ B^s)^c$, where X^c denotes the complement of X relative to E, i.e $X^c = \{x \in E | x \notin X\}$. It means that the closing is the complement of the locus of translations of the symmetric of the structuring element outside the image A. In the above example, closing of the two squares by a disk, results in the union of the dark-blue shape and the light-blue areas as shown in Figure 2.5.

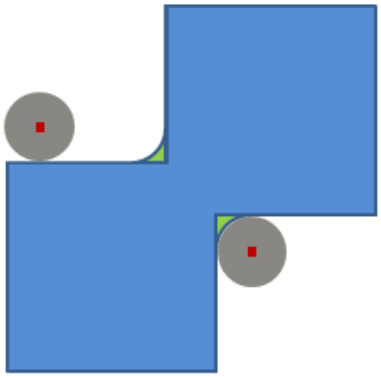


Figure 2.5: Closing of the two squares by a disk, resulting in the union of two squares

Closing and opening operators together serves as a basic workhorse of morphological noise removal in binary images. Opening removes small objects from the foreground (usually taken as the dark pixels of an image) and places them in the background, while closing removes small holes in the foreground, changing small islands of background into foreground.

2.4.3 Neighbourhood Templates

Moore neighborhood It comprises of eight cells surrounding a central cell on a two-dimensional square lattice. It is similar to the notion of 8-connected pixels as shown in Figure 2.6.

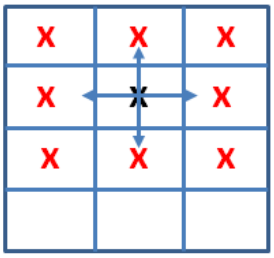


Figure 2.6: 8-connective (Moore neighborhood)

Von Neumann neighborhood It comprises of four cells orthogonally surrounding a central cells on a two-dimensional square lattice. It is similar to the notion of 4-connected pixels as shown in Figure 2.7

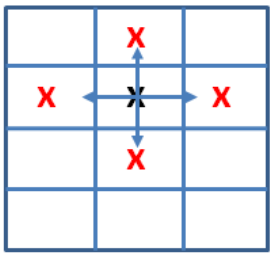


Figure 2.7: 4-connective (Von Neumann neighborhood)

2.4.4 Moore-Neighbour Tracing Algorithm

The boundary pixels of the structures, obtained after segmentation, in itself do not define the link among them. Thus to convert the image pixels into numerical data contours we need to have an algorithm that can do just so. The idea behind the use of Moore neighborhood tracing algorithm is to find the link among pixels of the given boundary structure. The Moore's algorithm is defined as follows: The Moore's algorithm

---

Algorithm 1
Input A square(pixel) tessellation, T(image) containing a connected component P of boundary pixels.
Output A sequence B $(b_1, b_2, ..., b_k)$ of boundary pixels i.e. the contour
Definitions M(a) = Moore neighborhood of pixel a, p = the current boundary pixel, c = current pixel under consideration, b = backtrack of c (i.e. neighbor pixel of p that was previously tested)

```
B ← ∅
for i_y := 1 to height do                                   ▷ From bottom to top
    for i_x := 1 to width do                          ▷ From left to right scan the cells of T
        if (i_x, i_y) is white then                     ▷ until a white pixel, s, of P is found
            s = (i_x, i_y)
            Insert s in B.
            p = s
            b = (i_{x-1}, i_y)
            c = next clockwise pixel (from b) in M(p).
            while c do ≠ equal s do
                if c is white then                                              ▷ backtrack
                    Insert c in B                                  ▷ move the current pixel c
                    b = p                          ▷ to the pixel from which p was entered
                    p = c
                    c = next clockwise pixel (from b) in M(p)
                else                                                            ▷ advance
                    b = c
                    c = next clockwise pixel (from b) in M(p).
                end if
            end while
        end if
    end for
end for
```

---

terminates after visiting the start pixel for the second time.

2.5 Reconstruction

2.5.1 Overview of the Problem

The problem of reconstructing the surface of a solid object from a series of parallel planar sections (referred hereinafter simply as slices) is to find a surface consistent with the observed polygons (or contours) in slices where each section consists of a set of closed polygons that define the boundary of the material of interest to be modeled. This kind of reconstruction should handle three intrinsic subproblems as defined by Meyers et. al.[34] and Bajaj et. al.[5] namely, the Correspondence, the Tiling, and the Branching problem.

Correspondence Problem

The correspondence problem arises when an object is represented by more than one contour in any of the sections of a data set. When that is the case, it is necessary to determine which of the contours in section are to be connected to contours of the adjacent sections. For example, we can think of an object as being composed of a number of tubes joined at branches where the number of contours representing an object in a section change (see Figure 2.8). A solution to the correspondence problem must

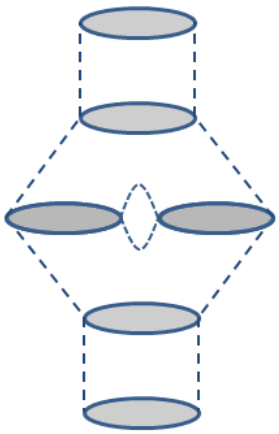


Figure 2.8: Represent the different solution for correspondence between contours in (a)

determine how best to connect the contours of a data set in light of these possibilities.

Figure 2.9 shows how correspondence between contours can be handled in many ways.

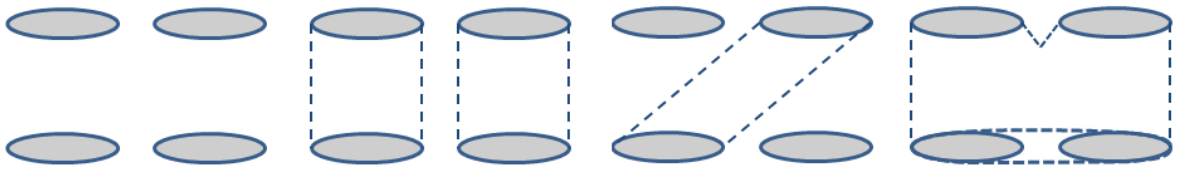


Figure 2.9: Represent the different solution for correspondence between contours in (a)

Tiling Problem

A solution of the tiling problem for a pair of planar polygonal contours is a piecewise planar surface having the contours as boundary curves. Since the surfaces of real objects do not self-intersect, a condition that the triangles in the surface do not intersect except along their edges can be added to obtain the solution to tiling problem. Figure 2.10 shows an example of the tiling problem and its solution. The tiling problem is easy to

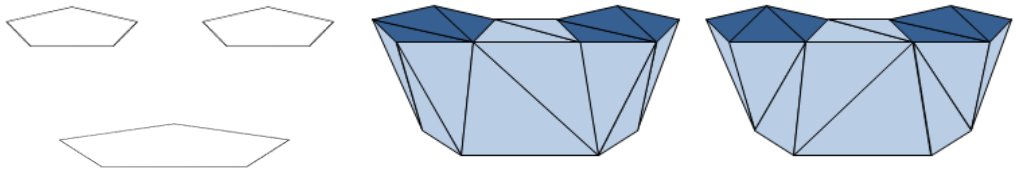


Figure 2.10: Different solutions to the tiling problem for contours in (a)

solve if the contours have similar shape, size and orientation. If they do not, as is often the case in real-world data, computing an acceptable tiling is more difficult. Part of the problem is that the mesh may contain unusually large triangles.

Branching Problem

Branching problem is related with handling the saddle points that may appear in the models. At a branch, a single tube can split into several tubes or some number of tubes may merge and then split again into a different number of tubes (see Figure 2.8). Figure 2.11 shows some cases that an algorithm for automatic branch construction should handle.

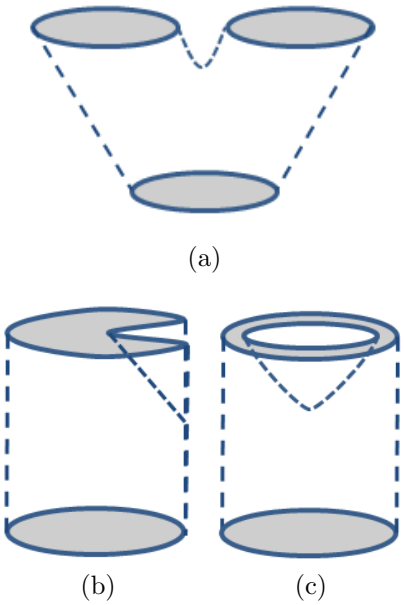


Figure 2.11: Different scenarios to be handled by branching problem

2.5.2 Classification of Reconstruction Approaches

The problem of reconstructing 3D models from 2D contours has been studied over many years. In order to contextualize the methodology proposed in our work, we show how alternative methods take advantage or lack in generating the volumetric reconstructions.

Optimization Methods Optimal approaches employ graph theory and dynamic programming strategies to build a 3D model. For example, in the work by Keppel[26], he reduced the problem of constructing a triangulated approximation to the surface between

a pair of contours to a search problem on a toroidal graph. He used maximization of enclosed volume as a metric for the search problem. The main difficulty with this method is that it required convex contours in the cross sections, thereby increasing computation time for breaking down non-convex into convex contours. Sloan and Painter[52] describe some improvements to the graph search used in the optimizing algorithm that result in a constant factor speedup of the algorithm. The Levins[30] method builds a set of intermediate contours by interpolating the adjacent contours in the z-direction with cubic B-splines, in order to solve the branching problem. It is based on calculating the distance field for each point of each section, where contours can be regarded as iso-curves with an iso-value of zero. The main limitation of this method is the large number of triangles obtained in the surface. Meyers[35] used the polygonal form of skeleton called medial axis to obtain information about the relationships of vicinity among the regions where ramifications occur. The possible types of connections among skeleton loops helped to classify the ramifications, however, the method does not work correctly in the cases of ramifications from many-to-many contours. In addition, the projections on a same plane of the contours related with the ramification can intercept each other.

Composite Contour Methods In this approach, contours are modified in sections for constructing a tiling when faced with branching objects. The most commonly used approach is to form composite contours from the post-branch contours. The section of the post-branch contours is treated as if it were two sections, one matching the section of the pre-branch contours and the other matching the section following the post-branch contours. Christiansen and Sederberg[13] describe a method which forms composite contours by adding a vertex between adjacent contours to model the saddle surface implied by the contours. It connects contours in neighboring slices by mapping their bounding box onto a unit square using the shortest edge to make a connection. In complex boundary regions, they resort to user interaction to guide a solution. Shantz[48] proposed a similar solution to the branching problem and went on to address issues that arise when the shapes of contours on neighboring sections differ significantly. His

strategy for composite contour construction used $n-1$ minimum-length links to form a composite from n contours, and resorts to user intervention to produce an acceptable result. He also addresses the difficulty of tiling pairs of contours with significant shape differences, by using the medial axis to improve the tiling in such cases. Ekoule et. al.[17] describe a method for constructing tiling of branching objects that involves formation of a composite contour that is then used at a level intermediate between the pre and post-branch contours. They form composites by using the maximum area polygon formed from the centroid of the contours to guide composite formation. Finally, a planar triangulation is computed for the region interior to the composite but exterior to each of the contours from which the composite was formed. A deficiency of this method is that it fails to consider the shape of the pre-branch contour when choosing the locations at which it joins the post-branch contours to form the composite.

Generalized Cylinder Methods In this approach, a generalized cylinder is used to solve the correspondence problem. A generalized cylinder is a solid represented by an axis curve in three space and a cross-section or sweeping function (that may be function of the location along the axis curve). Branching objects are described using collections of generalized cylinders. The efficiency of generalized cylinder methods depends to a large extent on how well the objects of interest can be modeled by generalized cylinders. If the objects of interest are not smooth and elongated, the generalized cylinder approach may not work well. Even if the objects to be reconstructed are well suited to the generalized cylinder approach, the success of generalized cylinder methods depends on the quality of the rules used for constructing cylinders from the input contours.

Soroka[51] proposed a method that involves assembling contours into elliptical cylinders, and then assembling them into objects. If many of the contours of an object are not elliptical, the method would be unable to incorporate them into elliptical cylinders and would be likely to fail. Bresler et. al.[6] present another way of using generalized cylinders to determine correspondence. They define a likelihood function over a generalized cylinder, based on the data collection method and a model of the objects to be reconstructed.

Graph Based Methods Graph-based methods usually compute a minimum spanning tree based on contour shape and position, or construct a graph based on Morse Theory to represent contour connectivity for solving the correspondence problem. The main limitation of the MST based algorithm is its inability to solve the correspondence problem correctly for general graph topologies. Skinner[34] compute the minimum spanning tree (MST) of the contour graph to solve the branching problem by assigning costs to the edges of the graph. Shinagawa, Kunii and Kergosien[50] describe a method using Morse Theory to construct a Reeb graph describing the connectivity of the contours in a data set. Construction of the Reeb graph for a set of contours requires that the user specify the topological genus of the object, and the number of connected components and genus of each in the case of multiple objects. Giertsen, Halvorsen and Flood[22] describe a structure very similar to the Reeb graph, but construct it manually during the process of digitization of data from electron micrographs. Their solution to the correspondence problem is essentially manual interaction, but uses the Reeb graph to organize the data. The algorithm proceeds by making the highest priority connections in regions where the number of contours in each section does not change, and then adds connections in order of decreasing priority subject to the a priori knowledge of the number of connected components and the topological genus. The quality of the results produced by this method depends on the accuracy of the knowledge about topological genus and number of connected components. For large spacing between sections, the authors state that a complicated data set would require much human intervention. In the work by Jones and Chen[25], they define a signed distance field to create an implicit function that, jointly with Marching Cubes[31], is used to approximate the boundary surface of the original object. The implicit function defines the correspondence automatically thus making it difficult to choose the different connections.

Volume based Methods Volume reconstruction methods form a mesh over the cross sections and then sculpt away parts of the mesh to make it agree with the data.
The approach that inspired many other works in the area was given by Boissonnat[3]. He

used the projection of Voronoi skeletons from the contours in consecutive pairs of slices to generate a graph. From this graph, a 3D Delaunay triangulation is constructed which undergoes a tetrahedron elimination process to generate the volumetric model. Region correspondence is defined both by the 3D Delaunay triangulation and the tetrahedron elimination process. However, his method does not describe a solution to the branching problem. Geiger[21] improved Boissonnats approach by handling the graph produced by the projection of the 2D Voronoi diagram in order to handle complex branches and dissimilar contours. Cheng and Dey[12] use Boissonnats theoretical results to reconstruct surfaces without generating the 3D Delaunay triangulation. Later, Bajaj et. al.[5] used the Voronoi skeletons generated from projections to generate the triangulation, where correspondence among regions is implicitly defined by the surface properties. A difficulty common to all such approaches is the geometric problems involved in computing projections and intersections. Also in the follow up of Boissonnat's mathematical framework, Nonato et. al.[37] explicitly derived a proximity measure from the 3D Delaunay triangulation to determine the correspondence. His approach also guarantees that reconstructed models are 3D manifolds.

Contour stitching Methods Most of the research on contour-based surface reconstruction has focused on methods for connecting (stitching) the vertices of neighboring contours into a mesh. Edelsbrunner and Mucke[16] presented a family of shapes (α-shapes) that can be defined by a point set and generated by a Delaunay triangulation based algorithm. However, the strategy adopted in (α-shapes does not ensure that the resulting model is a PL-manifold. Bernardini et. al.[7] described an algorithm for creating surfaces from unstructured points by connecting three points that solely lie within a sphere of user-specified radius to form a triangle. The sphere is pivoted around one of the triangles edges until it comes in contact with another point. Amenta et. al.[1] developed a surface reconstruction algorithm based on Delaunay triangulation and 3D Voronoi diagrams. Given sufficient sampling of the underlying surface the reconstruction is guaranteed to be topologically correct. Later, Amenta et. al.[2] performed surface

reconstruction by first calculating a piecewise-linear approximation of the underlying surfaces medial axis transform (MAT). The surface is then constructed by applying an inverse transform to the MAT. Dey et. al.[14] extended this work with the Cocone algorithm, which uses complemented cones of the Voronoi diagram to provide additional guarantees about the reconstructed surface. The Cocone algorithm itself has been extended to produce watertight surfaces and to handle noisy input data.

2.5.3 Voronoi Diagrams and Delaunay Triangulation

In this section, we briefly describe some definitions and properties of Voronoi diagram and Delaunay triangulations as described in [37].

Let $A = \{x_1, \ldots, x_n\}$ be a set of points in general position in R^m, i.e., there is no affine subspace of R^m containing A and there is no sphere S^{m-1} through a subset of A with $m + k$, $k > 1$ points. The Voronoi diagram for A is a decomposition of R^m into m-dimensional convex cells $V_1, \ldots, V_n$ with the following properties:

- Each V_i contains a single point x_i of A.
- Given $x \in R^m, x \in V_i$ if and only if $d(x, x_i) \leq d(x, x_j)$, for every $i \neq j$, where $d(x, x_i)$ is the Euclidean distance between x and x_i.

It has been shown earlier[4] that the intersection of k Voronoi cells, $2 \leq k \leq m + 1$, is either empty or is an $(m - k + 1)$-dimensional cell contained in the diagram and a triangulation can be obtained from the Voronoi diagram by associating each of its p-dimensional cells with an $(m - p)$-simplex. The triangulation so obtained is called Delaunay triangulation, and it maintains a duality relationship with the Voronoi diagram. From the duality relationship and the general position of the points it follows that each 0-dimensional cell of the diagram is the center of a sphere circumscribing an m-simplex and that this sphere does not contain in its interior any other points of A. This triangulation in used in computing the mesh for the given cross sections (section 4.3.3). These can be defined more formally as:

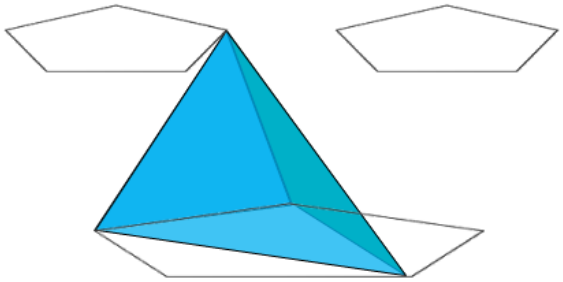
Empty sphere. The sphere S is called A-empty, if the open ball bounded by S does not include any point of A.

Delaunay simplex A simplex with vertices in A is called Delaunay, if there exists a A-empty sphere passing through all its vertices.

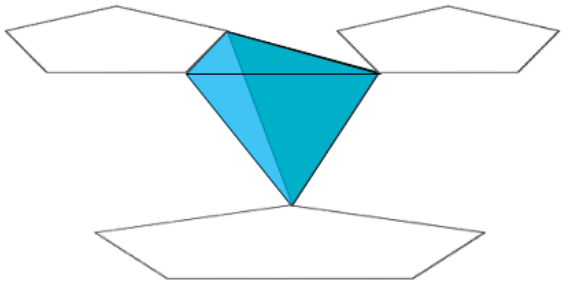
For example, considering the 2D case, each vertex in the Voronoi diagram (given by the intersection of three Voronoi cells) is associated with a triangle (2-simplex); each edge is associated with an edge of the triangulation (1-simplex); and each cell V_i is associated with a vertex x_i of the triangulation (0-simplex).

Tetrahedra Classification

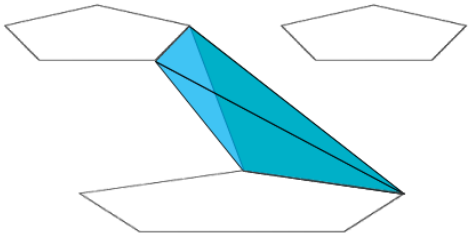
Let C_1 and C_2 be two sets of contours bounding regions contained in adjacent planar sections P_1 and P_2 and DT the 3D Delaunay Triangulation of the vertices in $C1 \cup C2$. Based on contour orientation, such that the interiors of the regions are always on their left-hand side, an edge of DT contained in P_1 or P_2 can be classified as either internal or external to a region while edges on the contours are labeled as contour edges. The tetrahedra of DT which have a face in $P_1(P_2)$ and an opposite vertex in $P_2(P_1)$ are classified as type 1 tetrahedra, or if they have an edge in each of the planes P_1 P_2, as type 2 tetrahedra. Based on the classification of its edges, a tetrahedron in DT is classified as:
Internal A tetrahedron with at least one edge internal to a region r is said to be an internal tetrahedron of r.



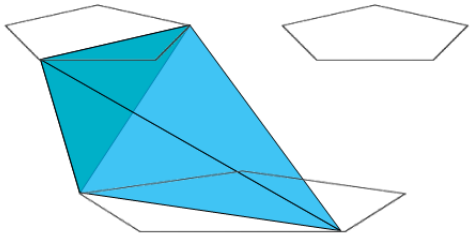
External A tetrahedron with at least one external edge and no internal ones is said to be an external tetrahedron.



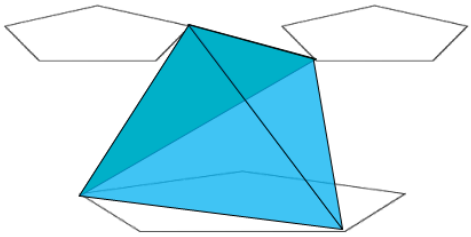
Redundant A type 2 tetrahedron with two contour edges is called a redundant tetrahedron.



Reverse Internal A type 2 tetrahedron whose edges in P_1 and P_2 are internal ones.



Reverse External A type 2 tetrahedron whose edges in P_1 and P_2 are one internal and the other external.



Voronoi Skeletons

Let VD_i be the 2D Voronoi diagram of the vertices of contours $C_i (i = 1, 2)$ in adjacent planes $P_i (i = 1, 2)$, respectively. The Voronoi skeleton of a region $r \in P_i$ is defined as the subset of the VD_i edges that are dual to the DT internal edges with points in the contours of C_i that bound r. Figure 2.12 shows a set of regions and their corresponding Voronoi skeletons.

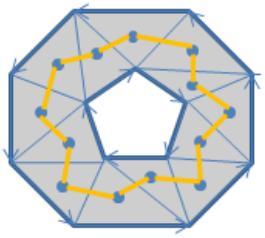


Figure 2.12: 2D Delaunay triangulation and the corresponding Voronoi Skeleton(orange)

Strong Overlapping Regions

Two regions $r_1 \in P_1$ and $r_2 \in P_2$ are defined as strong overlapping regions if the orthogonal projection of the Voronoi skeleton of r_1 onto P_2 intersects the Voronoi skeleton of r_2. The following two propositions (proven by Nonato et. al.[38]) shows that the strong overlapping regions can be directly defined from the three-dimensional Delaunay triangulations.

Proposition 1. Two regions $r1$ and $r2$ are strong overlapping regions if and only if the three-dimensional Delaunay triangulation of the points in $r1 \cup r2$ has a reverse tetrahedron.

Proposition 2. If two regions $r1$ and $r2$ are not strong overlapping regions, then either the internal tetrahedra of $r1$ have no intersection with $r2$ or all internal tetrahedra of $r1$ contain a vertex or a contour edge of $r2$.

Both of properties 1 and 2 completely describe how the three-dimensional Delaunay triangulation connects the contours in adjacent planar sections. It can be derived from the above propositions that the presence of reverse tetrahedra gives a measure of proximity between two regions $r1$ and $r2$.

2.5.4 Beta Connection

The β-connection algorithm described in this section is particularly adequate to the reconstruction of objects with high degree of branching as in neuronal structures. Based on the 3D Delaunay Triangulation, it can produce a family of models for a given set of cross sections. Model generation is governed by an integral parameter, denoted β, whose value controls the degree of correspondence amongst regions in neighboring slices. Increasing the value of β results in stronger region connection. The resulting models are guaranteed to be piecewise linear(PL) manifolds, a highly desirable property for numerical simulations. The algorithm also satisfies the re-sampling criterion, i.e., intersection of the resulting model with the original cutting planes produces the original set of planar cross-sections. We now start with a description of a graph structure from which beta-components are derived.

The DT Associated Graph The graph is computed in order to define the components which establishes the connection among the contours. The graph is defines as follows: Let C_1 and C_2 be two sets of contours representing the boundaries of regions R_1 and R_2 contained in adjacent planar sections P_1 and P_2, respectively, and DT the 3D Delaunay triangulation of vertices in $C_1 \cup C_2$. From DT, a graph $\mathcal{G}$ can be constructed that has two types of nodes. Each region $r \in R1 \cup R2$ and all its internal tetrahedra define a region node in $\mathcal{G}$, and each external or redundant tetrahedron of DT define an external node in $\mathcal{G}$. Edges linking pairs of nodes in $\mathcal{G}$ are obtained as follows:

1. If two regions r_1 and r_2 have internal tetrahedra with a common face, or if there is a reverse tetrahedron with internal edges in r_1 and r_2, then the region nodes representing r_1 and r_2 are connected by an edge in $\mathcal{G}$.

2. If an external or redundant tetrahedron t shares a face with an internal tetrahedron of a region r, then the region node r and the external node t are connected by an edge in $\mathcal{G}$.

3. Two external nodes are connected if and only if the tetrahedra they represent share a face,that is, external and redundant tetrahedra sharing a common face are connected by an edge in $\mathcal{G}$.

Figure 2.13 illustrates the graph $\mathcal{G}$ generated from DT of contours in Figure 2.10a. The graph $\mathcal{G}$ is a connected graph as all regions nodes of $\mathcal{G}$ contain tetrahedra that either share a face to another tetrahedra or contain face in another region node. A **path** of

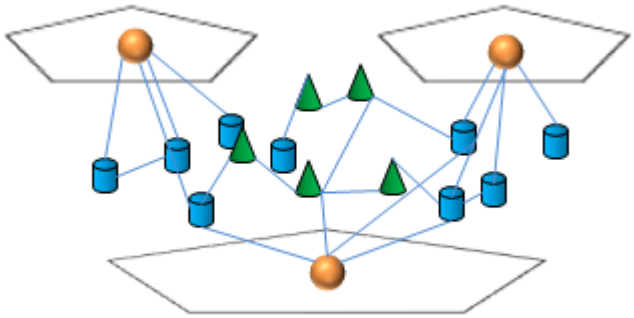


Figure 2.13: The DT graph

length n connecting two region nodes a and b in $\mathcal{G}$ is a sequence of nodes $\{\tau_1, \ldots, \tau_{n+1}\}$, such that $\tau_1 = a, \tau_{n+1} = b$ and (τ_i, τ_{i+1}), $i = 1, \ldots, n$, is an edge of $\mathcal{G}$. The **distance** between two region nodes a and b, denoted $d_{\mathcal{G}}(a, b)$, is the length of the shortest path between a and b.

Beta-Components Two region nodes a and b are said to be β-connected, denoted $a \overset{\beta}{\approx} b$, if there is a sequence of region nodes $\{\sigma_1, \ldots, \sigma_k\}$ in $\mathcal{G}$, where $\sigma_1 = a$ and $\sigma_k = b$, such that $d_{\mathcal{G}}(\sigma_i, \sigma_{i+1}) \leq \beta$, β is a natural number. It has been shown in [37] that β-forms an equivalence relation which guarantees that each value of β defines equivalence classes constituted by those region nodes of $\mathcal{G}$ positioned at a distance smaller than or equal to β from one another. As each region node represents a region contained in a plane $P_i (i = 1, 2)$, β also defines equivalence classes for the original regions, thus allowing correspondence to be specified through these equivalence classes. Varying the

value of β produces different equivalence classes, in-turn enabling multiple choices of correspondence amongst the regions. An important property of β-connection proven in [37] is that the strong overlapping regions (which share reverse tetrahedra) are the first to be connected for any $\beta > 1$. Also, no correspondence amongst regions is established for $\beta = 0$.

Component Disconnection As described above, each equivalence class originates a connected component in the reconstructed model. To disconnect components according to the equivalence classes specified by β all external and redundant tetrahedra from the Delaunay triangulation are removed, whose vertices are not in the same component. In some situations, even after tetrahedron elimination, components may remain connected through vertices or contour edges as illustrated in Figure 2.14. In this case, vertices (and edges) are displaced to completely disconnect components, as shown in Figure 2.14c. A special case when $\beta = 0$, when each region generates an independent connected

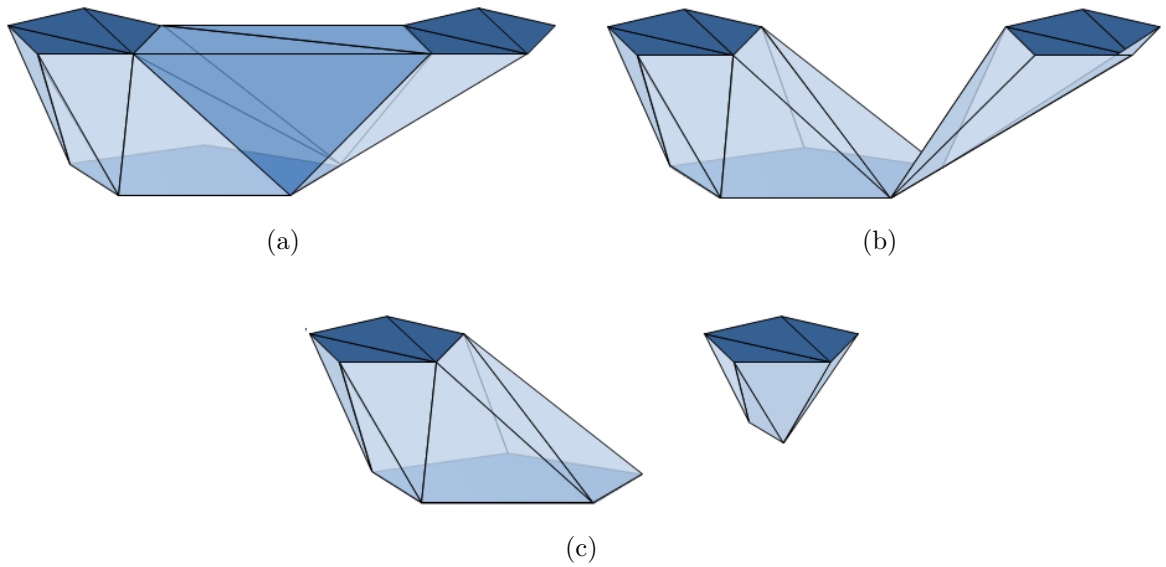
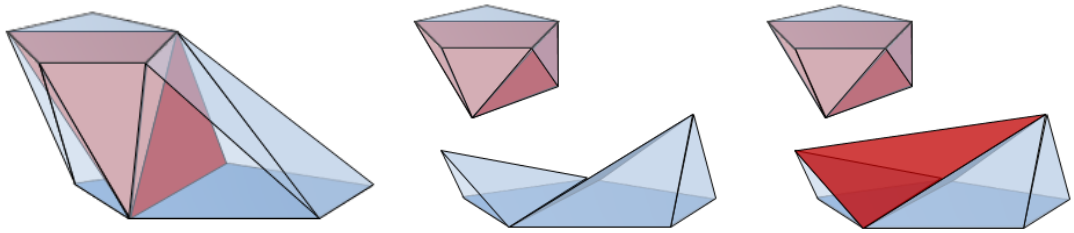


Figure 2.14: (a) 3D Delaunay triangulation (b) elimination of external tetrahedron (c) translation of vertices to avoid edge singularity on a contour edge

component. In this case, the presence of reverse tetrahedra in strong overlapping regions

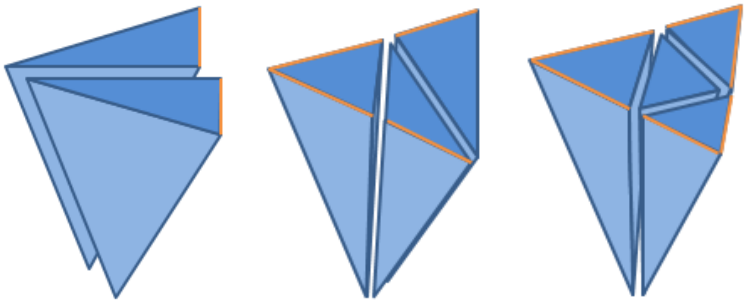
does not allow disconnection to be properly executed. The problem is that if vertices are translated to disconnect strong overlapping regions, reverse tetrahedra internal to both regions will remain in one of them and generate wedges in the internal edges of the other region (Figure 2.15b). A wedge characterizes a singularity, i.e., the boundary of the union of the tetrahedra around the wedge is not homomorphic either to a sphere or to a half-sphere. Singularities must be avoided in order to ensure that the reconstructed object is a PL-manifold. Thus, to disconnect strong overlapping regions it is necessary to duplicate the reverse tetrahedra and then translate vertices and edges, as shown in Figure 2.15c)



(a) (b) (c)

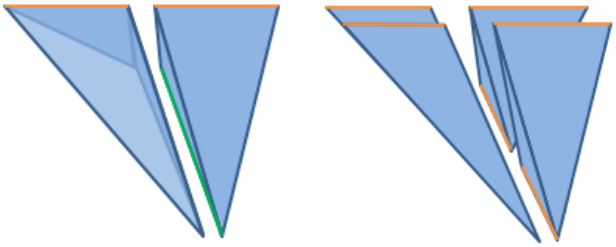
Figure 2.15: (a) Reverse tetrahedron (b) Singularity produced by translation of vertices (c) Duplication of reverse tetrahedron

Tetrahedron Subdivision To ensure that the connected components obtained from the disconnection process satisfy the re-sampling condition, β-connection algorithm also eliminates the external tetrahedra within components. However, as in the disconnection case, the removal of these may introduce singularities. The authors have solved this problem with the tetrahedron subdivision process(TSP). In this process, external tetrahedra whose elimination introduces singularities are subdivided (as shown in Figure 2.16 and 2.17), where new vertices inserted in each external edge are translated to an intermediate position between consecutive slices. The external edges of the tetrahedron are marked in orange. This subdivision process guarantees the manifold condition(i.e. free of singularity). Results of subdivision process applied to a reverse tetrahedron are shown in Figure 2.18.



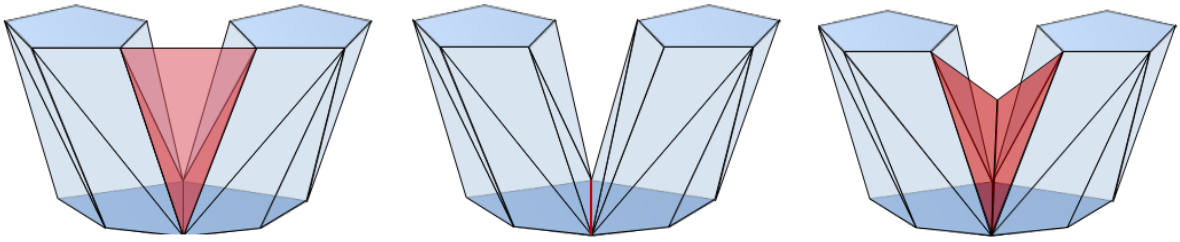
(a) (b) (c)

Figure 2.16: Subdivision of type 1 tetrahedra with (a) one external edge, (b) two external edges, (c)three external edges



(a) (b)

Figure 2.17: Subdivision of type 2 tetrahedra with (a)one external edge (b)two external edges



(a) (b) (c)

Figure 2.18: (a) reverse external tetrahedron(marked in red) (b) singularity created along the edge(marked in red) by removal of reverse external tetrahedron (c) subdivision and vertex translation of the reverse external tetrahedron

The Algorithm The summarized version of $\beta-$connection algorithm is described below:

Algorithm 2
Input: a set of contours C bounding the regions contained in two adjacent planar sections P_1 and P_2.
Output: a solid O' representing the structure between P_1 and P_2

Delaunay triangulation Compute the 3D Delaunay Triangulation of vertices in C and classify the resulting tetrahedra.

$\beta-$-components Create the Delaunay Triangulation associated graph and compute $\beta-$-components from the equivalence classes defined by β.

Connected components disconnection Remove external and redundant tetrahedra connecting the different β-components and if necessary, translate vertices and duplicate reverse tetrahedra.

Tetrahedra subdivision For each β-component:

- Remove external and redundant tetrahedra whose elimination does not introduce singularities, otherwise, subdivide the external tetrahedra and remove the corresponding redundant tetrahedra.
- For each subdivided tetrahedra, translate the new vertices to an appropriated position between slices.

2.6 Tools and Libraries

2.6.1 CGAL

CGAL is an open-source library written in C++ that provides algorithms that are commonly used in computational geometry. It offers data structures and algorithms for computing triangulations (2D constrained triangulations and Delaunay triangulations in 2D and 3D), boolean operations on polygons, mesh generation (3D surface and volume mesh generation) and simplification. These structures and predicates are regrouped in form of CGAL Kernels. It further offers interfaces to third party software such as the GUI libraries Qt, and the Boost Graph Library. In our work, we have used this library to compute the constrained 2D and regular 3D Delaunay triangulation for set of samples points in 3D space.

2.6.2 LibTIFF

LibTIFF is an open-source library, written by Sam Leffler, for reading and writing Tagged Image File Format (abbreviated TIFF) files . This library also provides a small collection of tools for doing simple manipulations of TIFF images. We have used the library to read the images acquired with microscopy, which are usually stored in TIFF format.

2.6.3 Paraview

Paraview is a multi-platform, open-source software, developed by Kitware Inc. that allows visualization of large data sets. It uses the Visualization Toolkit as the data processing and rendering engine and has a user interface written using the Qt cross-platform application framework. To speed up the visualization of the reconstructed model, our framework reconstructs the model in the format that can be easily incorporated into the Paraview. We also credit the software for providing an intuitive interface to visualize multiple models at the same time, the feature which we have utilized extensively while debugging our reconstruction module.

Chapter 3

Related Work

Several algorithms have been proposed for three-dimensional reconstruction of neurons from fluorescence laser-scanning microscopy data. The importance of the problem has led to quite a few attempts at developing semi-automated or automated methods which can be broadly divided in following categories:

Centerline-extraction based methods In this approach, neuron morphology is extracted from a centerline model(or skeleton) and the reconstruction is generated assuming a cylindrical model(a tubular-like shape of dendrites). Using this approach has the advantage of computing accurate topological structure, but issues such as of irregularities of dendrites surfaces are ignored while extraction of centerline. Also, this approach is computationally intensive since most of the operations are done on each voxel. Moreover, in contrast to boundary-extraction based methods, it involves both local and global characterization while modeling neurites.

Boundary-extraction based methods In this approach, voxels representing the boundary of structure are extracted using algorithms such as watershed or thresholding. These voxels are then chained together either by vectoral (directional) tracing or using dynamic programming to search for a minimum cost path. The main problem with this approach is the ability to produce continuity of edges, which then requires further postprocessing to link the broken edges. The linking algorithms may introduce unnecessary ambiguity and incorrect links of noisy data. Recent work[15] have adapted the approach to include global information to resolve these ambiguities.

Most of the modern methods have adapted themselves by using the combination of the above approaches to decrescent the disadvantages of each. In this section, we outline existing methods for neuron reconstruction and discuss their advantages and disadvantages. Considering the incomplete results of the automatic procedures, neurobiologists still use the commercially available semi-manual processing techniques. The accuracy of these however is strongly dependent on individual interpretation to estimate mid-lines and diameters of dendrites. One of the commercially available software tools is NeuroLucida (by MicroBrightField, Inc) which performs the manual tracing of boundaries of the neurite structure where user specifies the terminal and seed points. Unfortunately, the manual tracing of complex dendritic trees in large datasets becomes overly tedious. Therefore, the automatic reconstruction procedure are better for fast visualization and extraction of the rough topology.

Amir[45], proposed a fully automatic model-based approach for the automatic reconstruction of neurons based on the use of a tubular shape as the model assumption. The model is constructed by composing the partially recognized parts of neurons structures through successive processing steps. Firstly, he used the support vector machine(SVM) algorithm to recognize the voxels of the image which represent the neurons structure. The resulting representative voxels were then grouped together with the aid of a mean shift clustering algorithm. The clustered segments of the neurites are then connected together to compute the skeleton of the neuron. Finally, a registration algorithm is used to fit the reconstructed model onto the given image. Using his approach, the resulting structures have more accurate boundaries of the segmented structure, but, due to the noisy feature space of SVM, the voxels representing thin neurites are not classified from the background voxels. Even after extending the feature space, by including the voxels gray values and gradients, the classifier is unable to accurately distinguish the noisy artifacts. Thus, the resulting reconstructed model consists of many connected spherical components. The author has presented results for reconstruction of real neuron using a single SVM classifier trained using a toy data set.

Schmitt et. al.[46, 18] use a semi-automatic method that employs computation of skeleton as well as boundaries of the structures for extraction of morphology. They first use a segmentation algorithm based on geodesic active contours to compute the cross-sections(or boundaries). A skeleton is then computed based on centerline where user initializes the branching points of skeleton to represent the structural description of the neuron(the topology and the dendritic lengths and diameters). Finally, the surface is built using generalized cylinders that fits a surface to the structure using the geodesic active contour model. A three-dimensional distance map of voxel's distance to the nearest cylinder of the skeleton reconstruction is computed which is subsequently deformed to best fit the image intensity distribution, loosening the restriction to cylindrical cross-sections. The surface is filtered by a smoothness constraint that fits data by minimizing an energy-functional.

Zhao et. al.[58] compute a geometrical model using a 3D cylinder filter to formulate the shape of a neurite fiber. From the model, they derive a tracing algorithm based on deformable templates. Once the set of fragmentary trace or neurite fibers have been obtained for an image, they are assembled to form a neuronal tree structure. They define a neurite graph consisting of individual fibers and assign the geodesic distance as the cost of edges. The graph is then filtered to derive tree structure. The method takes the advantage of dealing with noise while tracing the neurite segments itself. The template parameters can also be varied to adapt to different neurite sizes. If many branches are present in a small region, such as the end of an axonal projection, the fitting template may jump from one branch tip to another or fail to fit on a short segment between two branch points resulting in topological errors. Also the methods fails to trace complete segments when there are discontinuities in neurites.

In [44], Rodriguez et. al. developed a semiautomated cell extraction algorithm based on a two-pass thresholding procedure. Prior to cell extraction, threshold selection is performed on the deconvolved images by interactively adjusting the threshold using a 2D contouring algorithm until the contour fits exactly over the projected outline of the

spines. In the first pass, the extraction routine marks all connected pixels from an initial seed inside the cell. A threshold that just overestimates the extent of the cell is then selected, so that the spines maintain connectivity to the dendrites. A second pass resurfaces the resulting dataset, at the original, minimal threshold value. which isolates the cell from the background, so that the dataset is able to captures spines structures. Finally, a polygonal surface is extracted from the volumetric data using a variation of the Marching Cubes algorithm, which generates a piecewise linear isosurface. Variation of standard Marching cubes is done to avoid ambiguous saddle cases, which can lead to holes in the surface. For surfacing highly curved objects such as the spiny dendrites, this method produces goods result at low computational cost.

Fua et. al.[20] describe a method for tracing of dendrites pattern by computing an optimal tree using a modified MST procedure, similar to He et. al.[24], which combines a EM-based local estimate of the probability of voxel belonging to a neuron filament with the global tree(or skeleton) properties of the complete structure. They formulate the problem as Bayesian inference and avoid having to de-noise the image by keeping a probabilistic semantic in the result of the segmentation step. The problem is decomposed as derivation of inference measure from actual location of the dendritic tree and its voxels visibility in the images. The authors have presented results only for 2D images and quote it extension for 3D cases. The problem with the approach involves high computational cost, especially to handle the list of all possible voxels.

Santamaria et. al.[49] use a similar approach of [45] to classify voxels assuming the tubular shape of the dendrite structures using SVM. The initial dataset is filtered to remove noise using frame-based de-noising on the optical images. They present a novel approach for extraction of the centreline model, for which a tubular measure is derived by from the eigen-values and by learning the relation between structural information (the eigenvalues) and the actual tubular object of interest. The eigenvalues are used to train the model by assigning the labels to those eigenvalues that belong to the dendrite. From the centerline model, branching points and dendrite segments are constructed by

tracing back the paths from the terminal points to the root, to represent the set of paths as a single connected tree structure. A surface is then computed by assigning the classified voxels to level sets and computing the dendrite lengths and diameters using the distance transform. A preliminary filtering of voxels belonging to soma and pipette is also done by using a K-means algorithm that utilizes a mask(ellipse) that encloses both the soma and pipette. The authors have presented good results regarding the extraction of topological structures.

Dima et. al.[15] present an approach for neuron segmentation using a 3-D wavelet transform to perform a multiscale validation of dendrite boundaries. This segmentation is used to construct a skeleton complete with an estimate of local axial directions and their variances, which are then used to locate bifurcations. Initial edge detection is done by filtering the image with low-pass filters of different size followed by an application of the gradient operator. The low-pass filtering reduces noise, but it also changes the location of edges due to a blurring effect. The local segmentation has the advantage of detecting the weakest and thinnest neuronal structures present in the images, however, is unable to distinguish the background noise. To avoid this ambiguity on local neighbourhood, they introduced a new measure called "across-scales validation" to distinguish the boundary edges from the noise. The measure is dependent on spatial distribution of boundary edges in the adjacent neighbourhoods and contains high values for edges found on adjacent neighborhood. The resulting boundary edges are then traversed across their gradient direction to segment the voxels belonging to object. Since gradient directions may not be accurate and edge points may be missing due to noise, false pairing situations also appear in the resulting dataset, which are corrected using the seed fill algorithm. The skeleton and its branching points are computed from the segmentation, which constructs at each voxel the list of directional vectors(or rays) which intersect each other. Their results show gaps in the skeleton, for which they plan to add a manual correction option. They also extract the branching points and points of high curvature (bends of dendrites), to detect the variance in axial directions of voxels, which enables detection of incorrect boundary edges. They show that across-scales validation of edges

and the almost blind elimination of noisy edges makes their method their independent of variations in contrast, and the object sizes.

Hamilton et. al.[24] present an algorithm that operates by performing connectivity testing over voxel neighborhoods to extract a graph representation of the structures. Voxels are obtained using the combination of thresholding, skeletonisation, and thinning process. The resulting graph is then analyzed for width of connected nodes to filter the voxels that belong to the soma and the artifacts. Assuming the tree structure of the neurite, graph is then filtered to prevent the cycles and loops by computing the MST. Resulting fragments are connected through the semi-interactive method where user specifies the region of error. They identify all of the critical points within this user-specified region and define the radius for connection among fragments at each critical point. Finally critical points are connected via assigned cost. The surface is computed using the vacant-neighbouring scheme over voxels. Although, the resulting structure consists of accurate topology, it is unable to represent any intermediary branching between voxels and the method is computationally expensive since it operates on each voxel for every process in the algorithm.

Al-Kofahi et. al.[27] describe a semi-automatic reconstruction scheme which uses an adaptive exploratory search at the voxel intensity level. They compute the skeleton on the data using directional kernels, guided by a generalized 3-D cylinder model. A seed point and a direction is initially chosen by the user from which the tracing algorithm begins. The procedure terminates upon reaching the end of traced segment. The soma is detected using a combination of thresholding and connected component operations on the pixels of raw images. Since, their method is able to work directly with unprocessed confocal images, without any preprocessing, it is computationally faster than other algorithms. Adaptive templates are used to trace over discontinuities in the dendrites which combines the two or more partial traces to form a complete trace based on the local intensity information and the local orientations of the partial traces. However, to ensure a complete trace, the selection of redundant points becomes mandatory, making the task

user-dependent and complex.

Peng et. al.[42] developed a novel semi-automatic graph algorithm, called the all-path pruning (APP), to trace the 3D structure of a neuron. An APP initially produces an initial over-reconstruction(graph) by tracing the optimal geodesic shortest path from the seed location (specified by the user) to every possible destination voxel/pixel location in the image, to avoid potential mistracing of some parts of a neuron. They simplify the reconstruction by pruning the redundant structural elements, using a maximal covering minimal-redundant (MCMR) subgraph searching algorithm (modification of Djikstras algorithm). The algorithm consists of three procedures which are run successively. First procedure called "dark leaf pruning" chooses a threshold which defines the lowest visible voxel intensity to iteratively remove all leaf nodes whose respective voxel intensity is below that threshold. Second procedure called "covered leaf pruning" defines a radius-adjustable sphere (called structural component(SC)) centered at a node, and then enlarges the radius gradually until 0.1% of the image voxels within this sphere are darker than the global threshold (average voxel intensity of the entire image). Using the volume projection of nodes, nodes that are covered by other reconstruction nodes are marked redundant and are pruned. After covered-leaf node pruning, redundant inter-nodes (that connect leaf nodes to branching nodes or the root) are removed using an iterative process which checks for volume projection of parent and child nodes and removes the parent if it covers its child. The resultant graph consists of shortest path map from leaf nodes with a tree like structure. The algorithm has is s efficient since it uses the topology of the initial reconstruction to constrain the search space which reduces a combinatorial search to linear search. It is also able to implicitly handle dealing with the broken or punctuated neuron structures, or low SNR images.

Meijering et. al.[33] adapted a semi-automatic interactive technique called live-wire segmentation paradigm for the tracing of elongated structures in images. A Gaussian kernel and a steerable directional filter is used to compute the intensity and orientation of structures (called ridges), given by the eigenvector corresponding to the eigenvalue

with smallest magnitude. The ridges are the elongated structures which shows the large variation in the intensity values along the direction. They also compute a function for translating local ridge intensities and orientations into costs maps. The adjacent ridge pixels are then linked using the live-wire segmentation paradigm, where user initially specifies the a starting point, followed by a graph-searching algorithm to find the shortest paths from that point to all other points in the image according to a predefined cost function. This procedure is repeated again where user moves the cursor to the next point along the path of interest and the new path is computed again. Although, the technique is computationally inexpensive and exploits the expertise of the user in solving ambiguities, it could become cumbersome for a user to handle complex morphologies of dendrites.

Urban et. al.[55] proposed a skeletonisation algorithm to produce the segmentation of the structures in the images. The high frequency noise is removed initially using a 3-D Parseval frame based filter. Following denoising, thresholding is employed to binarize the denoised data. The authors explicitly determine the voxels that belong to the pipette and the soma employing the information that these two corresponds to brightest features (voxels) in the images. To extract the medial axis of the segmented neuron, they employ a distance transform(DT)-based skeletonisation method that guarantees to be tree-structured, which is then used to cull false dendrites from the neuron. A measure of saliency related to how much a branch lies outside the skeletal reconstruction is also computed. An approximating spline is then fit to each path in the tree of paths representing the medial axes of the salient branches in the binarized neuron data. Finally, a cylinder-tree representation of the neuron is computed that using dendrite widths along each spline. The framework proposed by the authors is automated and fast in terms of computation, but is unable to handle apparent dendrite gaps. Also, it may be possible that short dendrites, which merely originate from surface noise, belong to legitimate branches, which the framework does not account for.

Zhang et. al.[59] proposed an automated 3D centerline extraction algorithm to extract

morphology of the neuronal structures. their approach combines 3D Dynamic Programming (DP) technique and marker controlled watershed algorithm to solve the problem overlapping dendrites. The approach consists of tracking and updating along the navigation directions of multiple axons via computation of spanning tree to find the path that gives the shortest geodesic distance between two points.

Uehara et. al.[54] described a neuron reconstruction algorithm based on a wave propagation methodology. Using gradient eigen analysis, the algorithm generates a field indicating the probability of each voxel belonging to a cylindrical structure (assumed to be dendritic). A digital wave is then propagated through this field to provide dendrite paths. However, the multi-scale gradient analysis used here is prohibitively expensive for large volumes.

Chapter 4

Methodology and Implementation of Algorithms

This chapter describes the detailed formulation of our neuron reconstruction process done using the adaptation of the algorithms described in chapter 2.

4.1 Image Enhancement and Preliminary Filtering

Modern microscopy equipment typically includes a PC-class machine for instrument control and data analysis. Given the volume of raw data and the task at hand, it makes sense to discard as much data as possible in the earlier stage. Preliminary filtering includes noise reduction and enhancement of images to reduce the storage space and tune the focus/brightness/contrast of the data. For example, in confocal microscopy, both excitation light and fluorescence light are increasingly scattered with imaging depth, causing a reduction in signal-to-noise(SNR) ratio. Low SNR makes it difficult to resolve the thin neural structures in the images. To resolve this, non-linear diffusion filters can be used to reduce the scatter while retaining sharp edges. Point spread function (PSF) imposed on the data by the optics of the microscope can also be computed in this stage to deconvolve the image using a standard deconvolution method.

Although not necessary for our framework, these methods have proven effective to produce more precise segmentation of structures from the images.

4.2 Segmentation

Segmentation as described earlier is essentially a process of classifying voxels of structure from the background. Often automatic segmentation methods fail to preserve the details of fine structures due to the poor signal-to-noise ratio or imperfections in sample preparation of microscopy images. The design of fully automated techniques remains a difficult problem, and it seems likely that some form of user interaction will always be required to resolve ambiguities. Accepting this fact and attempting to meet the requirements of problem defined below, we have developed an interactive segmentation technique that exploits the expertise of the user in extracting the structure, but greatly simplifies the task by providing the visual and automatic tools. Based on the problem definition given in section 4.3.2, the goals of the segmentation module are: (i) a reliable segmentation of structures, independent of image contrast and scale of input data, (ii) removal of artifacts from the images, and (iii) fast computation time.

4.2.1 Algorithm

Given an image stack of a real neuron, the following algorithm extracts the neuron structure from the images and represent in the form needed by the reconstruction module.

Step 1 [**Thresholding**] Binarize the image into foreground and background pixels.

Step 2 [**Visual Manipulation**] Remove artifacts such as pipette etc in the images.

Step 3 [**Morphological Operations**] Remove artifacts such as small holes or objects in the segmented image caused by presence of air bubbles or dust while preparing the sample.

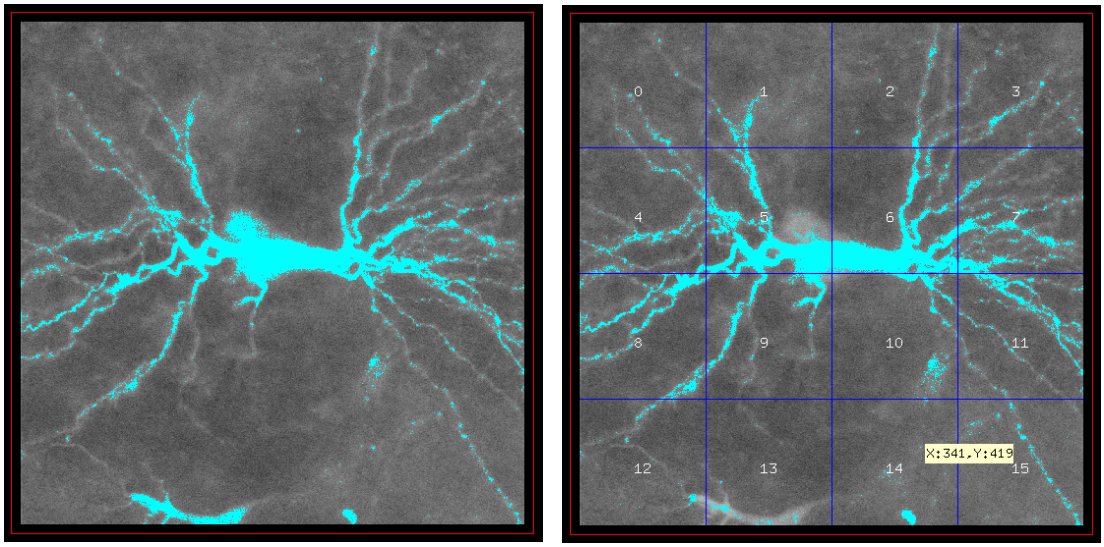
Step 4 [**Contour Extraction**] Convert the boundary pixels of segmented structure into contour data structure.

Step 5 [**Connected Components Calculation**] Eliminate spatially distant noisy structures by computing the largest connected component among the extracted contour set.

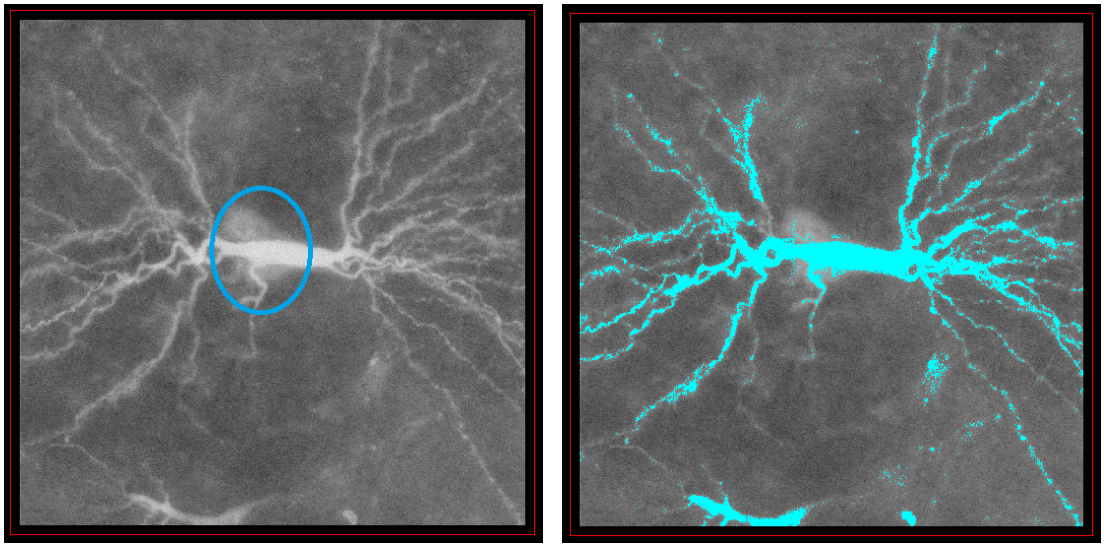
Since we use local neighbourhood for the image of input data in steps 1-4, we were able to execute the corresponding steps in parallel for each image. However, to achieve noise robustness we perform the connected component analysis on global scale of the input data.

4.2.2 Thresholding

As described earlier in section 2.4.1, it is needed to decide an optimum threshold value at which the maximum amount of information about the object of interest is revealed and the minimum amount of information is lost, however, presence of different kind of noise makes the automation of optimum threshold selection a difficult process. In our work, we experimented with one or two methods from the approaches defined in section 2.4.1, where none could guarantee the accurate segmentation of the objects. Thus, we designed visual tools to produce a reliable segmentation and employ advantage of fast computation of thresholding technique. To ensure that all the possible signals are captured, we first produce the initial over-complete segmentation of the neuron with the average intensity value of the entire image as the threshold and provide the user with the overlay of source and threshold image for manual verification of the initial segmentation (Figure 4.1a). User can easily manipulate the segmented image in realtime by varying the threshold value to extract more information. User is also given the choice to specify local threshold for a resizeable bounding region (Figure 4.1b) in the image. This way, user can even extract small structures which may not be covered with a global threshold for entire image. Additionally, user can also cull objects or artifacts, within the specified region (Figure 4.1c and 4.1d).



(a) (b)



(c) (d)

Figure 4.1: (a) Extraction of structure with overlay of source image (b) Grid Overlay for local threshold selection (c) Artifact (represented by circle) in the image (d) correction of artifact with grid tool

4.2.3 Morphological Operations

In the raw image, it may be possible that the pixels interior to the structure have intensity closer to the background or is surrounded by more number of background intensity pixels, which will lead to presence of small holes or objects in the binary data, respectively (as shown in Figure 4.2a and Figure 4.2b). To remove these artifacts, we employ morphological operations such as openings and closing described in section 2.4.2. In essence, we check the neighborhood of the pixel for its label, whether it belongs to foreground or background where we use both 8-connective and 4-connective neighborhood templates while performing morphological operations. The choice of template and the number of operations are tunable by the user after visual verification. The results of different operations are shown in Figure 4.2.

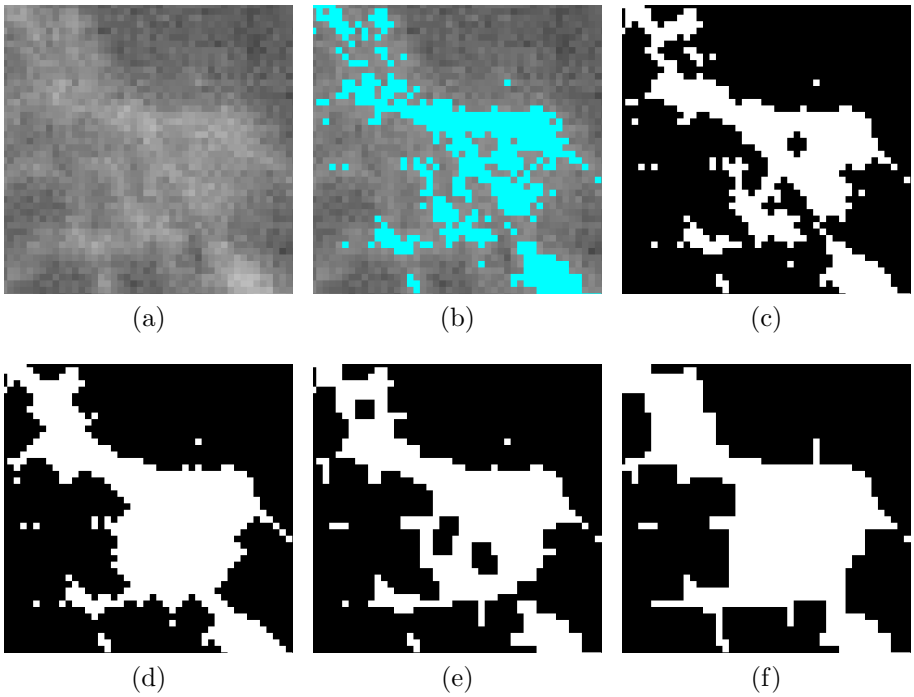
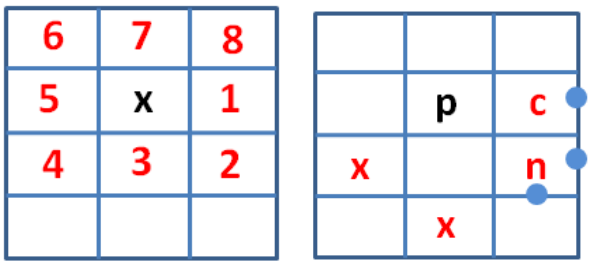


Figure 4.2: (a) Raw data (b) Binary Data, (c) One morphological operation of opening followed by closing operator using 4-connective template (d) Two morphological operations of opening followed by closing operator using 4-connective template (e) One morphological operation of opening followed by closing operator using 8-connective template (f) Two morphological operations of opening followed by closing operator using 8-connective template

4.2.4 Boundary Extraction and Contour Tracing

Boundary Extraction In the segmented image, we can easily discard interior pixels of structures, since they can be represented by the outer and inner boundary pixels alone. To extract these boundary pixels, we check the neighborhood of each pixel for its label. If the 4-connective neighbourhood template of a pixel p contains the background pixel, then we mark p as the boundary pixel.

Contour Tracing Based on the requirements of our reconstruction module (defined in sec 4.3.3), we need a data structure that represents the links in boundary pixels of the extracted structures. We define these link in contours, where each contour is a internal or external respective to whether its chains the inner or outer boundary pixels. To establish the link in boundary pixels, we designed an algorithm which is loosely an adaptation of the Moore-Neighbour tracing algorithm (section 2.4.4). Since, the algorithm in itself was not sufficient to fulfill the requirements needed by our reconstruction module, we modified the algorithm to produce non-intersecting contours. This makes the process more efficient by avoiding the calculations inherent from division of intersecting contours to non-intersecting contours. To trace a contour, we search each row from the top-left

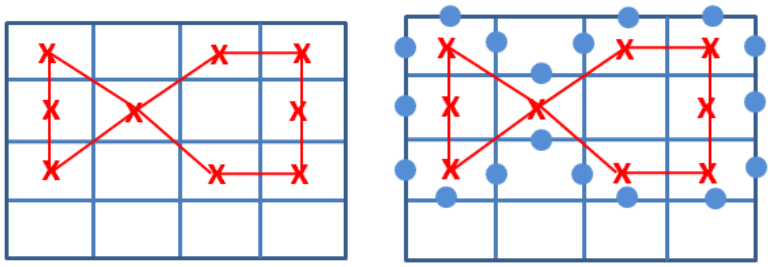


(a) (b)

Figure 4.3: (a) Neighbourhood tracing order (b) Path template (marked by circles) for the given p,c,n ordering of pixels

corner of the image to find an unscanned boundary pixel s. Starting at the pixel s, we follow its neighborhood in clockwise order checking for its label as shown in the Figure 4.3a. Let p, c and n be the boundary pixels discovered in order of neighbourhood

traversal. Instead of marking them as the boundary pixels, we add the pixels from the predefined path templates for all possible positions of p, c and n in the neighborhood. Figure 4.3b show a path template (represented by circles) for one of the configurations of p, c and n. Each dot in the path template is computed as mid point of corresponding pixel. These path templates are defined to ensure that boundary pixels are single pixel wide. We also handle a few corner cases where we explicitly insert additional pixels to these templates to maintain this property. We continue to add these path templates based on the configurations of p, c and n discovered until c reaches the starting pixel s, which marks the end of a contour. The main scanning algorithm terminates until all boundary pixels are processed. This way we ensure all boundaries in the segmented image are detected and traced only once. Since we consider all possible configurations of p, c and n, we are able to trace both convex and non-convex boundaries. However, internal boundaries are traced by the same algorithm after reversing the order of neighborhood traversal. (Figure 4.4) shows a external boundary where a boundary pixel is inserted twice using Moore-Neighbor Tracing algorithm, and how by using path templates we prevent creation of intersecting contours. The contours from the tracing algorithm are



(a) (b)

Figure 4.4: (a) intersecting contour(represented by cross) resulting from Moore-Neighbor Tracing Algorithm (b) contour(represented by circles) from our path-template based algorithm

stored as unstructured polygons in a VTK data file. Figure 4.5 shows the application of the algorithm applied to trace the boundaries of Dataset I.

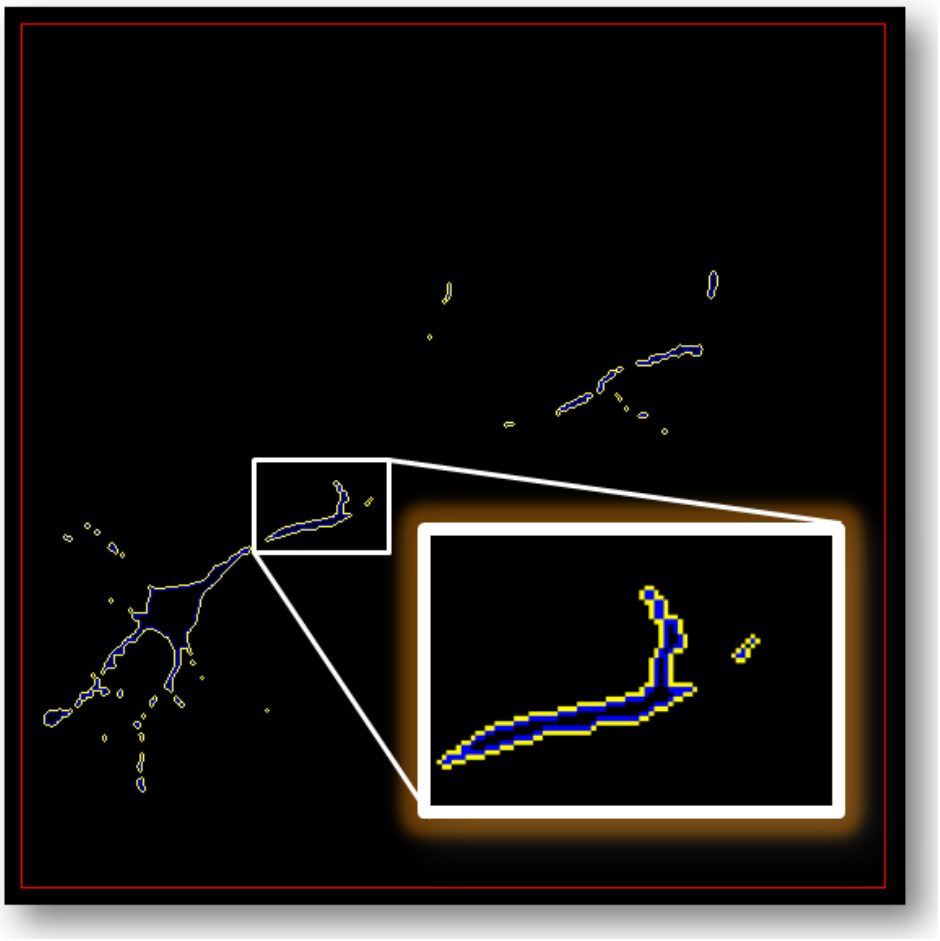


Figure 4.5: Contour tracing of boundary image in Dataset I

4.2.5 Connected Components Analysis And Extraction

In order to remove the spatially distant contours (present due to poor segmentation while removing artifacts), we employ connected component analysis on the contour stack. This steps results in significant and much needed data reduction, thereby improving the performance of the reconstruction module. To identify and eliminate these contours, we build an undirected graph $\mathcal{G} = (V, E)$ on the set of contours. Edges of the graph have an associated weight, equal to the geodesic distance between its nodes. The geodesic distance between two contours C_1 and C_2 is defined as $dis_g(C_i, C_j) = min(dis(v_p, v_q) \; \forall \; v_p \in C_i, v_q \in C_j)$, where $dis(v_p, v_q)$ is equal to the Euclidean distance between vertex v_p and v_q. A pair of nodes C_i and C_j ($C_i \in$ slice S_a and $C_j \in$ slice S_b), have an edge between them iff $a = b$ or S_a, S_b are adjacent slices and $dis_g(C_1, C_2) < threshold, t_0$. We then perform breadth-first search (BFS) on the graph $\mathcal{G}$ to determine the connected components. Finally, we group the contours of extracted components into a new 3D stack of slices. Since we perform the connected component analysis on the set of contours than voxels we are able to save considerable computational time to filter these noise-caused structures. Usually, the selection of the largest component is sufficient to represent the

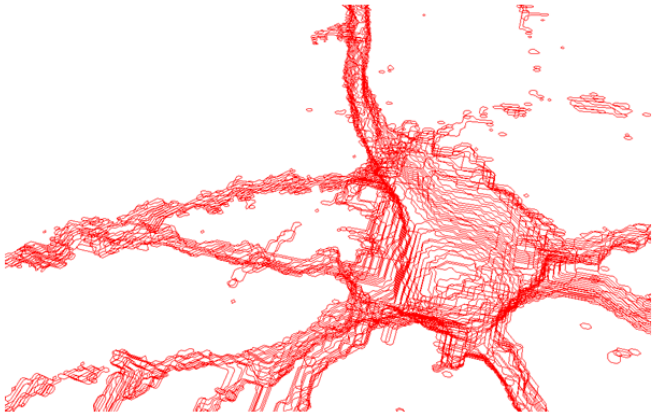


Figure 4.6: 3D Visualization of components contours

structure, but the choice is given to the user to enable selection of multiple components from the results. A projection of the aligned stack of contours is then presented to the

user where user can visually validate the extracted contours set (Figure 4.6). If necessary, user can vary the threshold in the viewer and visualize the result in realtime. The choice of threshold t_0 is simplified via analysis of components over all possible values of thresholds. We collect the number of components and the length of the largest component information for analysis. Threshold t_0 can be chosen from the values where two graphs stabilize. Figure 4.7 shows the component analysis plot for Dataset I where the optimal range for t_0 is 20-24.

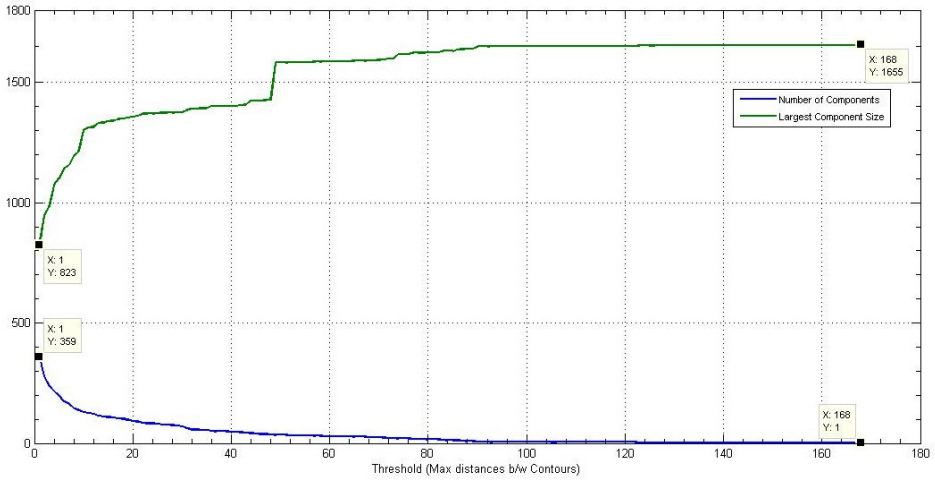


Figure 4.7: Plot of number of connected components and size of largest connected component of the component graph. This plot is used to choose an appropriate threshold for the component extraction

4.2.6 Conclusion

We conclude that our segmentation technique completely fulfills the required criteria (see section 4.2) as it enables fast visualization and extraction of the rough topology of the neuron structures. Also, it is independent of the data being used and can be employed on data obtained from any imaging modality.

4.3 Reconstruction

The reconstruction method present in this section is an adaptation of β-connection algorithm (section 2.5.4), modified to correctly handle the branching problem for reconstruction of neuronal structures. The applicability of the algorithm is illustrated in section 4.3.1. The algorithm relies on the construction of the Delaunay Triangulation(DT) from the contours contained in parallel cross sections. The DT of the contours set automatically generates an volumetric triangulation connecting the contours.

4.3.1 Why Beta Connection

We chose β-connection algorithm as the reconstruction method as it offers several advantages over other algorithms presented in section 2.5.2. It makes no prior assumption about the input and operates on any kind and number of contours without any requirement of user interaction during the process. It is also capable of generating multiple alternatives when establishing region correspondence and intrinsically handles tiling and branching problem. Many of the algorithms mentioned in section 2.5.2 are unable to handle the correspondence problem in a flexible manner, without limiting the number of contours in the slices, their geometries, or their containment hierarchies. In addition, the β-connection algorithm can handle instances of multiple branching without leaving gaps between the contours. The algorithm also deals with the topological singularities that may appear during the process of reconstruction so that the reconstructed object is a manifold. More importantly, all the operations in the reconstruction process are combinatorial enabling a more robust and efficient implementation.

4.3.2 Decomposition of the reconstruction problem and related definitions

Section A section A_i consists of a set of simple polygons $C_j, j = 1, \ldots, n$, some possible lying inside others, representing the intersection of a plane P_i with the object $\mathcal{O}$. To state this reconstruction problem formally, we introduce the following definitions in terms of implementation.

DEFINITION 4.1. *Given the sections $A_1, \ldots, A_k$ as the input, our goal is to reconstruct an approximation $\mathcal{O}'$ of the original object $\mathcal{O}$. We further decompose the problem into reconstruction of partial objects $o_i, i = 1, \ldots, k-1$, where o_i is the reconstruction between adjacent sections A_i and A_{i+1} and satisfies the following conditions*

Conformity Condition *All edges of contours in A_i, A_{i+1} are contained in the o_i.*

Manifold Condition *o is a bounded compact set whose boundary is locally everywhere a topological disk.*

The conformity condition ensures that the reconstructed object boundary doesn't intersect the contours, while the manifold condition forbids connections between sections along zero or one dimensional features. Since the reconstructed objects $o_i, i = 1, \ldots, k-1$ will satisfy the conformity condition, they can be glued together in the end and their union will constitute the overall reconstructed object $\mathcal{O}'$. i.e. $\mathcal{O}' = \bigcup o$.

Simplices. A simplex σ^p of dimension p in Euclidean space $\mathbb{R}^m$, $0 \leq p \leq m$ is the closed convex hull of $p+1$ points $v_0, \ldots, v_p, v_i \in \mathbb{R}^m$, in general position, i.e., when the vectors $v_1 - v_0, v_2 - v_0, \ldots, v_p - v_0$ are linearly independent. For example, 0-simplex is called a vertex, a 1-simplex is called an edge, 2-simplex is called a triangle, and 3-simplex is called a tetrahedron. The points $v_0, \ldots, v_p$ are called the vertices of σ. A face of σ is defined by any j-simplex $0 \leq j \leq p$ spanned by a subset S of $v_0, \ldots, v_p$ (i.e. convex hull of S). The faces of σ different from σ itself, are called the proper faces of σ; and their union is called the boundary of σ denoted by $\partial\sigma$. If σ is a face of a simplex τ then τ is said to be incident to σ and τ bounds σ.

Simplicial Complex. A simplicial complex $\mathcal{K}$ is a finite collection of simplices satisfying the following conditions:

- If $\sigma \in \mathcal{K}$ then all faces of σ belong to $\mathcal{K}$
- If $\sigma, \tau \in \mathcal{K}$ then either $\sigma \cap \rho \neq \emptyset$ or $\sigma \cup \tau$ is a common face of σ and τ

Regularized Simplicial Complex. A regularized simplicial complex $\mathcal{K}$ is a three-dimensional simplicial complex such that any p-simplex, in $\mathcal{K}$, $p = 0, 1, 2$ is contained in at least one 3-simplex of $\mathcal{K}$.

Star and Link. The star of a simplex $\sigma \in \mathcal{K}$ is the union of all simplices in $\mathcal{K}$ having σ as a face, denoted by $st(\sigma, \mathcal{K})$ The link of σ is the union of all simplices of $\mathcal{K}$ lying in $st(\sigma, \mathcal{K})$ that are disjoint from σ. For example, if k is a vertex in a triangulation $\mathcal{T}$ in $\mathbb{R}^3$, the star of k is the union of all tetrahedra containing k and the link of k is the union of the faces of these tetrahedra that do not contain k. Figure 4.8 shows the star (marked in blue and orange) and the link(marked in blue) of an edge.

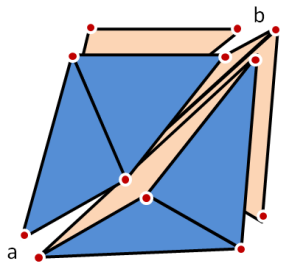


Figure 4.8: star and link of edge ab

Manifolds. A simplicial complex $\mathcal{K}$ is a p-manifold if the star of a vertex in $\mathcal{K}$ is homeomorphic to open set of Euclidean space $\mathbb{R}^p$. In particular, if $\mathcal{K}$ is a manifold, then every $(p-1)$ simplex in $\mathcal{K}$ is bounding either one or two p-simplex.

Boundary. The boundary of p-manifold $\mathcal{K}$ is the sub-complex $S \subset K$ constituted by all $p-1$-simplex that are incident to only one $p$simplex. The simplices that are not on the boundary are called interior simplices. For ex. A 2-simplex t (triangle) of $\mathcal{K}$ is an

interior triangle if t is shared by two tetrahedra of $\mathcal{K}$, otherwise, t is a boundary triangle. The vertices and edges contained in the boundary triangles are called boundary vertices and boundary edges of K, respectively. Figure 4.9 shows an example of interior and boundary simplex of a 3-manifold.

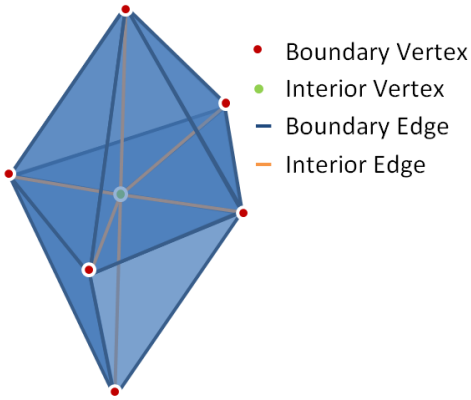


Figure 4.9: Interior and Boundary simplex of a 3-manifold

Singularity A simplex $\sigma \in \mathcal{K}$ has a singularity at a point p if its link is not homeomorphic to a sphere or to a hemi-sphere. Figure 4.10 shows a 2D example with singularity at point p.

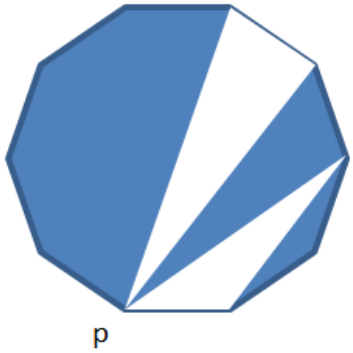


Figure 4.10: 2-manifold with singularity at point p

4.3.3 Algorithm

In this section, we describe the concepts regarding the implementation of the β-connection algorithm and the modifications done to handle the branching problem in a different manner. To summarize the β-connection algorithm, a mesh of tetrahedra is constructed between slices, with contour points as vertices. New vertices are then added to ensure that the contours are part of the mesh. Finally, some tetrahedra are eliminated to make the volume consistent with the contours. The reconstructed surface is then extracted as surface of the remaining volume.

Arrangement of Input data The input data consists of a sequence of cross-sections, where each cross-section consists of a hierarchy of contours (closed simple polygons with non-intersecting boundaries). A parent (or external) contour fully encloses all its children, and no other parent contour encloses a parent. Each section is also marked by its height along the z axis; thus, every vertex can be specified by its three coordinates. Based on the requirements of the algorithm, following arrangement on the input sections are mandatory:

- The planes of sections should be parallel and aligned in z-direction in 3D space.
- The boundaries of the contours contained in the sections should be non-intersecting(i.e. simple contours). This conditions helps to define the boundary of the reconstructed object, necessity of which is described later in section4.3.3.
- The contours should be oriented so that interior or exterior of regions are clearly defined. We orient the contours consistently, so that for each contour, when viewed from above, the structure lies to its right. Consequently, all external contours are oriented in a clockwise direction, and all internal contours are oriented in a counterclockwise direction.

If the containment hierarchy of the contours is omitted from the segmentation, it can be re-computed in the reconstruction module. The construction of the hierarchy and of the contour orientations is easily performed using a standard line-sweep procedure in each

section. Henceforth, we focus reconstruction of the partial object o i.e. a reconstruction formed by considering the space between pair of adjacent planes at a time.

The Initial Delaunay Triangulation

The first phase of the reconstruction module generates a 3-dimensional Delaunay triangulation that contains all edges of the contours on the two consecutive slicesM_i and M_{i+1}. Boissionat[3] proved following properties that proved to be helpful in relation to computation of 3D Delaunay mesh from planar cross sections:

Property 1 If the points in A are positioned on two consecutive parallel planar sections P_1 and P_2, the intersection of their 3D Delaunay triangulation with $P_i(i = 1, 2)$ is the 2D Delaunay triangulation of the points in P_i. This property helps to define coherence between sections (i, i-1) and (i, i+1).

Property 2 If the points in P_1 and P_2 are polygon vertices, polygon edges that are not contained in the Delaunay triangulation can be subdivided into new edges, until they are contained in the triangulation. This property necessitates the requirement 2 of input data, so that edges of contours in sections can belong to the Delaunay triangulation of its vertices.

In order to fulfill the conformity condition following the Property 2, vertices are added to the contour edges until all the contour edges are contained in the resulting triangulation denoted by T. To obtain such triangulation in an efficient manner, the following algorithm has been adopted:

1. Compute the 2-dimensional Delaunay triangulation D of the vertices in each section.

2. Mark the edges of the contours that are not contained on D.

3. Subdivide all marked edges, inserting new vertices on the contours.

4. Repeat these steps until the triangulation contains all contour edges.

5. Obtain a new 3D Delaunay triangulation that includes those new vertices obtained in step 4.

We now consider the 3D Delaunay triangulation T, the union of the contour-vertices and the new added points. After the 3D Delaunay triangulation has been built, star and link of simplices are the only necessary operations to be done on the resulting solid mesh. With the data structure (explained in section 4.5) we adopted, these topological tests can be executed in linear time. Once we obtain the Delaunay triangulation, we classify the edges as specified in the β-connection algorithm, depending on the relative position of the edge with the corresponding sections, with the addition of "in-section" edges and "ground" edges. "in-section" edges are the ones that have one vertex of tetrahedron in one section and other in the adjacent section, while "ground" edges are the ones that lies entirely in one section.

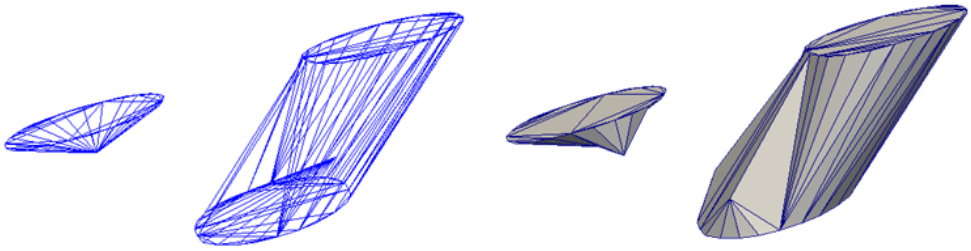
Reconstruction - Adaptation of β-Connection Algorithm

The basis of the β Connection Algorithm was initially introduced by Boissonnat[3] which proved that the regions that are geometrically well positioned can be found through topological tests on the DT. Later, Nonato et. al[37], showed that the distance measure among regions can be intrinsically derived from DT, responsible for establishing the connection among them. The distance parameter was defined as a positive integer parameter, called β. Varying β thus allowed the construction of different models for a given set, as evident in Figure 4.11.

Component Disconnection As described in the section 2.5.4, it is necessary to disconnect the components completely by translation of vertices to intermediary level between sections, after removal of external and redundant tetrahedra. However, instead of computing the centroid of the contour as the new intermediary voxel as demonstrated by results in the paper[37], we compute the centroid of the tetrahedra whose vertices are being translated as the new intermediary voxel. Figure 4.12 shows how this way we can avoid intersections in the reconstructed surface.

(a) (b) (c)

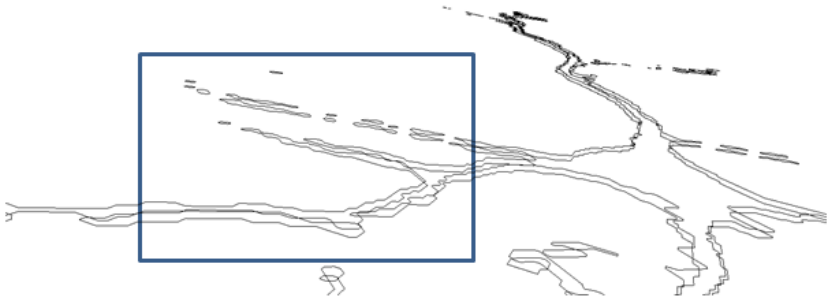
Figure 4.11: Models generated for β = (a) 0 (b) 1 (c) 3



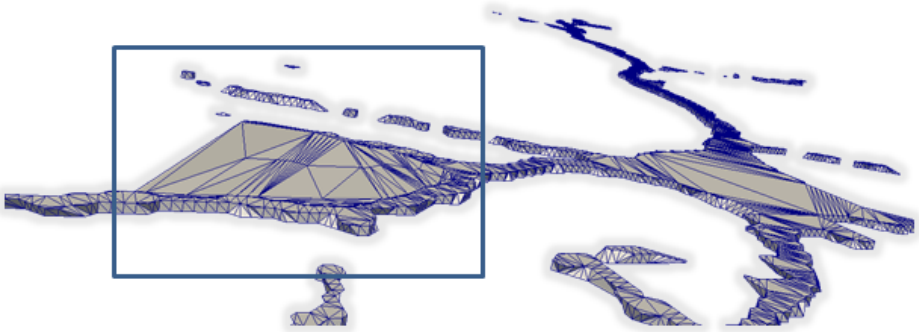
(a) (b)

Figure 4.12: (a) Intersection created in the reconstructed surface with centroid of the contour as the intermediary voxel (b) Reconstructed surface using centroid of the neighbouring tetrahedra as the intermediary voxel

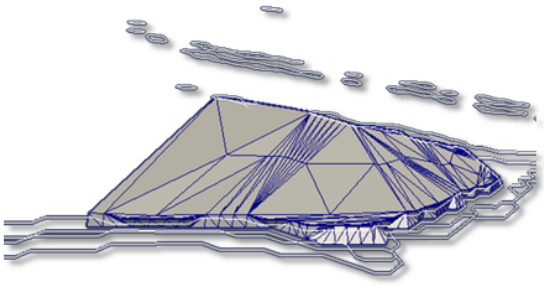
Tetrahedron Subdivision The β-connection algorithm in itself reconstructs the objects that satisfies the conformity and manifold condition. The algorithm subdivides the external tetrahedra and translates the vertices created on the subdivision of these tetrahedra to an intermediary position between the slices, generating branches at that intermediary level. The objective of this subdivision process is twofold (1) to guarantee that the intersection of the object with the corresponding planes is exactly the original set of contours and (2) the reconstructed object should be a manifold with boundary. The results of the algorithm for a pair of adjacent sections of Dataset I are shown in Figure 4.13. However with the subdivision process, the β-connection algorithm generates branches in inner regions of adjacent sections as shown Figure 4.13c, where it is necessary to avoid this kind of branching in the reconstruction of neuron structures. Following the results from the Figure 4.13c, we handle the branching problem by completely avoiding the connections in external regions and still produce a complex which satisfies the manifold condition. The underlying tetrahedron disconnection method, which jointly with the β-connection correspondence strategy, improves the usability of our volumetric reconstruction technique.



(a)



(b)



(c)

Figure 4.13: (a) A pair of adjacent sections from Dataset I (b) Reconstruction of surface between sections in (a) (c) Close-up view of reconstructed surface which shows branching of contours in adjacent sections resulting from the β-connection algorithm

Disconnection Algorithm The algorithm of tetrahedron elimination within β-components is listed below:

1. Eliminate the external tetrahedra which do not generate singularities and corresponding redundant tetrahedra.
2. Subdivide the edges and disconnect the external and redundant tetrahedra whose elimination generate singularities.

The disconnection of the tetrahedron in the external regions is based on following criteria:

Type 1 tetrahedra. For a type 1 tetrahedron with two external edges, we divide along the external and in-section edges and translate the vertices along those edges. For disconnection along edges, we chose the orientation specified in the Figure 4.15a where *cb*, *dc* and *bd* are the "ground" edges. When the type 1 tetrahedra contains three external edges, we follow the division criteria as in type 2 (Figure 4.15b) where *cb*, *dc* and *bd* are the "ground" edges.

Type 2 tetrahedra. For a type 2 tetrahedron, we subdivide along all edges and translate the vertices along those edges. For disconnection, we chose the orientation specified in Figure 4.15b where *da*, *cb* are external edges.

Redundant tetrahedra. A redundant tetrahedron is subdivided using the criteria of type 1 tetrahedra with two external edges, with the possible orientations defined in Figure 4.16. Our disconnection algorithm is different from the β-connection subdivision pro-

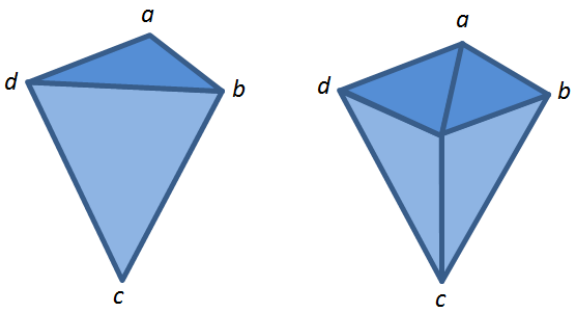
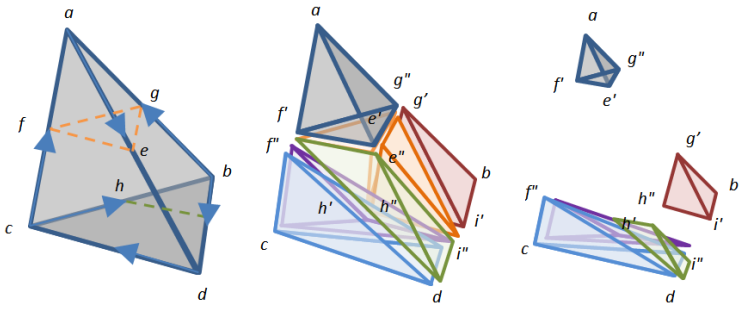
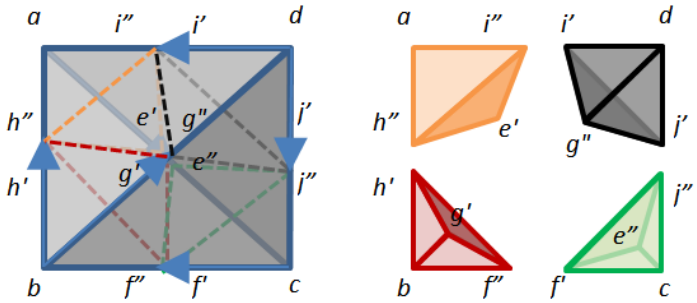


Figure 4.14: Subdivision of type 1 tetrahedra with one external edge bd

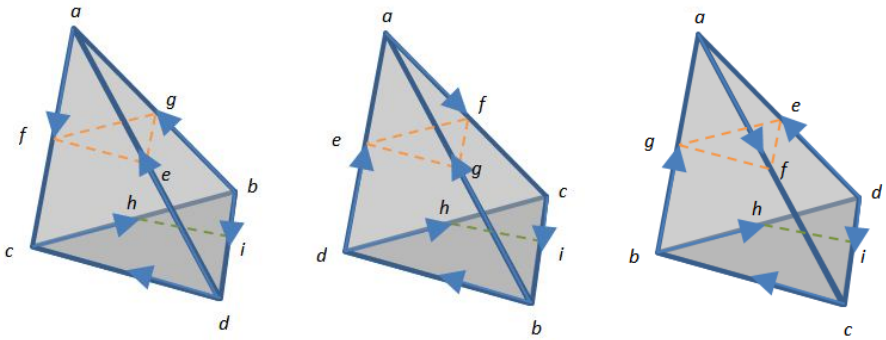


(a)



(b)

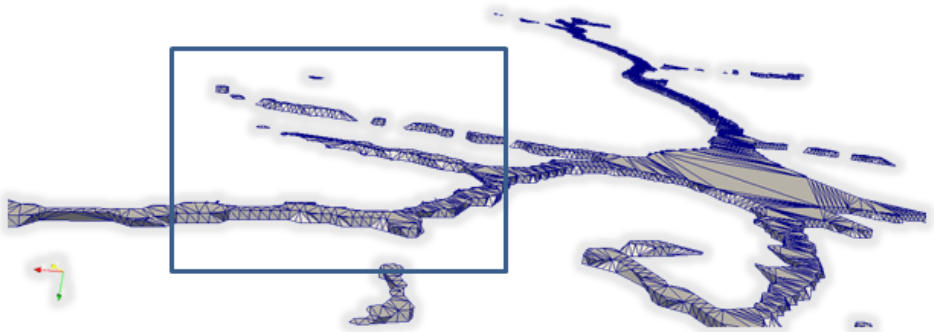
Figure 4.15: Disconnection of (a) Type 1 tetrahedron with two external edges (cb and bd) (b) Type 2 tetrahedron with two external edges(ad and bc)



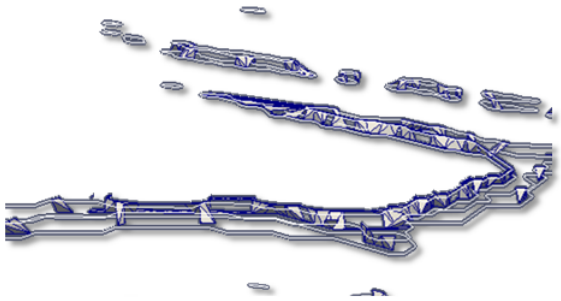
(a) (b) (c)

Figure 4.16: possible orientations of redundant tetrahedra for disconnection with external edge (a) dc (b) bd and (c) cb

cess in how it disconnects both external and redundant tetrahedra while β-connection retains both of them and subdivide only external ones. The disconnection algorithm solves the branching problem in a very satisfactory way and also confirms to manifold condition. Since the disconnection process respects the contours (i.e. the subdivided tetrahedron always contain its contour edges), the neighbourhood around the edge is always maintained, guaranteeing the manifold condition. In order to enforce the re-sampling condition, we can also translate the new points inserted in the subdivision to an intermediate position between the parallel planes, however this may not be necessary in terms of reconstruction of neuron structures. Figure 4.17 illustrates the results of disconnection algorithm applied to same pair of sections of Dataset I.



(a)



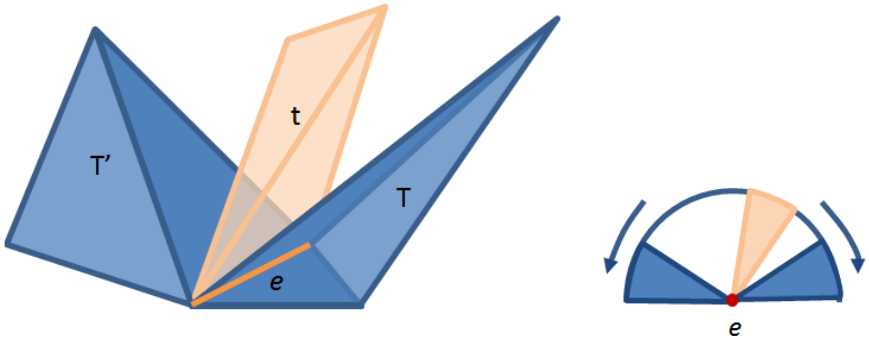
(b)

Figure 4.17: (a) Reconstruction of surface of sections in 4.13a (b) Close-up view of branching in contours from our disconnection method

Handling Singularities In this section, we demonstrate how we deal with singularities that appear in the reconstruction process.

DEFINITION 4.2. ***Face-to-Face connectivity*** *Two tetrahedra T and T', are said to be face-to-face connected in a complex $\mathcal{K}$, if there exists a sequence of tetrahedra $T = T_0, \ldots, T_k = T'$ such that for all $i, T_i \in \mathcal{K}$, and (T_{i-1}, T_i) share a face.*

Based on the above definition we can now define the edge e as singular if any tetrahedron in star of e is not face-to-face connected in $\mathcal{K}$ to at least one tetrahedron incident to e. Let T and T' be two tetrahedra incident to an edge e. A pivot from T to T'around e is a sequence of face-to-face connected tetrahedra incident to e joining T to T'. A tetrahedron t is disconnected if and only if elimination of t does not change the number of face-to-face connected components around e. We compute these components as follows. We start pivoting at t around its ground edge e in the $star(e)$ until we find t or a tetrahedron incident to e that does not share its face with another tetrahedron of $star(e)$. In the latter case, we start pivoting again in the opposite direction to determine the next component. Tetrahedron t incident to e is disconnected if it can be visited twice with different directions or if it does not shares both of its faces containing e. In both the cases, number of face-to-face connected components remains same after its elimination.

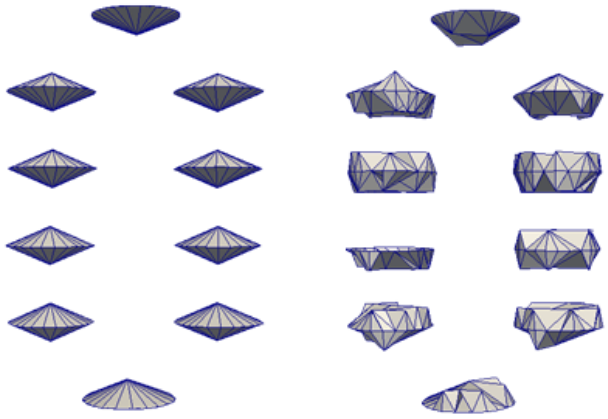


(a) (b)

Figure 4.18: (a) Star of edge e in tetrahedron t (b) 2D view of star of e

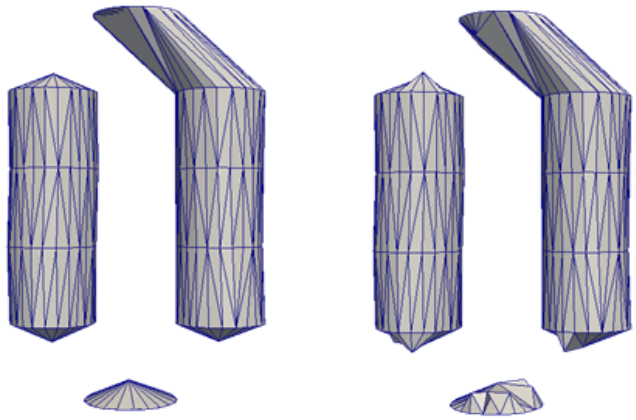
Comparison with Beta connection Figures 4.19 and 4.20 show side-by-side comparison of our disconnection method with the β-connection subdivision algorithm in the reconstructed mesh, using torus as the data model with different values of β. Left side shows the results from β-connection subdivision algorithm, while right side shows results from our disconnection method.

(a)

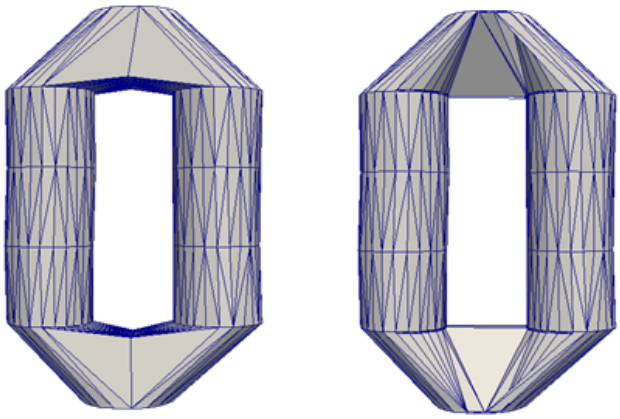


(b)

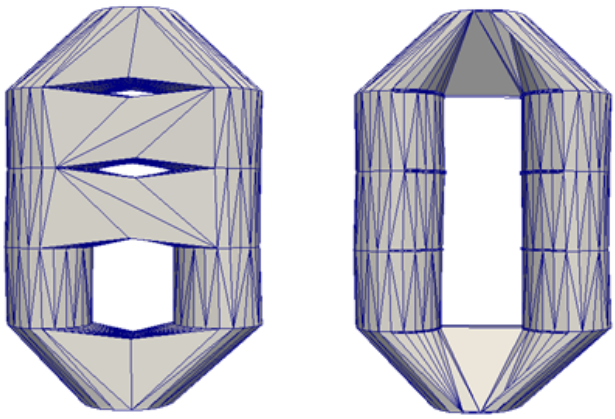
Figure 4.19: (a) Contours of torus model (b) Reconstruction with $\beta = 0$



(a)



(b)



(c)

Figure 4.20: Reconstruction with β = (a) 1 (b) 3 (c) 5

4.4 Corrections

It is possible that the resulting model consists of multiple components, if the segmentation process produces a set of spatially distant contours as a result from insufficient dye penetration in microscopy images, and the reconstruction module is unable to establish a connection among them with the specified value of β parameter. We can solve this problem by varying the β parameter to establish a higher correspondence in regions of disconnected component, but since the value remains same for an inter-slice region, it might create unwanted connections in another regions. We rectified this problem by computing the minimum spanning tree(MST) of the components, which finds relevant connections between the contour nodes. The process is carried out on the cross-sectional data.

We first build a graph $\mathcal{G}_1$ among the contours in cross-sections, with its edges formed by relation $dis_g(C_j, C_k) > 1 \ \forall \ C_j, C_k \in$ slices S_i, S_{i+1}. Next, we compute the components by performing a BFS on graph $\mathcal{G}_1$. These components (shown in Figure 4.21) are then used as nodes for constructing the graph $\mathcal{G}_c$. The edges of graph $\mathcal{G}_c$ are assigned a label $lab(Co_j, Co_k) = min(\ dis_g(C_m, C_n) \ \forall \ C_m \in Co_j \cap S_i, \ C_n \in Co_k \cap S_i; \quad i = 1, \ldots, k)$, where Co_j and Co_k are the nodes of graph $\mathcal{G}_c$. Next, we filter the edges of graph $\mathcal{G}_c$ by the relation $lab(Co_j, Co_k) < threshold, t_1$. The selection of threshold is again done through component analysis plot as shown in Figure 4.22. Finally, we compute the MST on the filtered component graph $\mathcal{G}_c$ using Kruskal's algorithm[28]. The resultant output of the algorithm is a tree or a forest, determined by connectedness of the filtered component graph $\mathcal{G}_c$. To establish the correspondence among the contours of obtained forest(or tree), we rasterize the segmented images along the shortest path connecting the two contours using Bressenham Line Drawing Algorithm[8]. The images are then refetched to the pipeline to generate a new reconstruction. The above mentioned process increases the proximity factors of distant regions(or contours), which then enables the correspondence among them in the reconstruction module with the same β value.

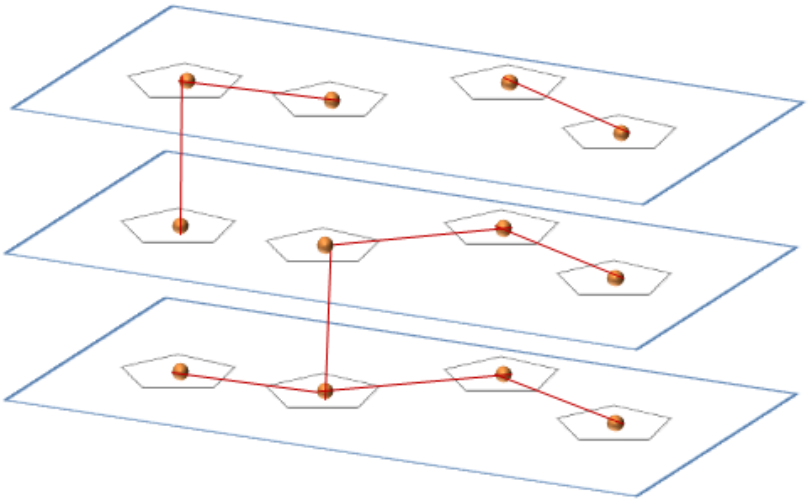


Figure 4.21: Components of Contour Graph $\mathcal{G}_1$

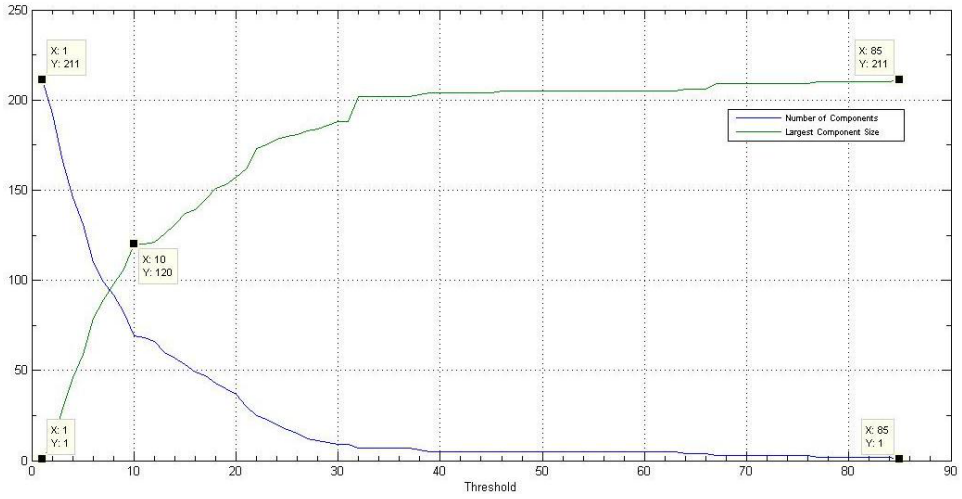


Figure 4.22: Plot of number of connected components and size of largest connected component of the component graph. This plot is used to choose an appropriate threshold for the establishing connections

4.5 Data Structure

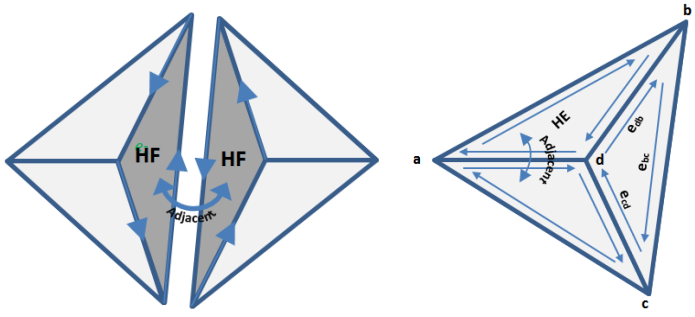
One of the challenges of volume reconstruction is the definition of data structure to support fast object manipulation and representation. In our implementation, we use half-edge data structure (adapted from [36]) to represent tetrahedral meshes and to efficiently perform the topological operations. It optimizes the memory consumption / execution time ratio for different topological operations in reconstruction process. However, instead of using the data structure in its entirety, we have adapted the same to our needs, where we have substituted singularity and boundary computation properties with the algorithm to maintain the efficiency. Given a regularized simplicial complex $\mathcal{K}$, a HE data structure is organized in terms of following entities:

Solid represents each connected component of the complex $\mathcal{K}$.

Cells represents the list of tetrahedra of the solid.

Vertices represents the vertices of the solid.

Half-Faces represents the semi-faces of cells. It is a either a pointer to semi-face sandwiched between two adjacent cells (share a common face) or the boundary face of the cell. The adjacency relationships among half-faces are also stored where each half-face knows the adjacent half-face in the neighbor cell, as shown in Figure 4.23a. The diagram in Figure 4.23b shows a small section of a half-edge representation of a triangle mesh where each half-face consists of circular lists of its half-edges to define its orientation. This half-edges can either be oriented clockwise or counter-clockwise around the face based on the orientation of overall mesh.



(a) (b)

Figure 4.23

Half-Edges represent the half-edges of cells. A half edge is a pointer to the oriented semi-edge of the half-face represented by its initial vertex as shown in Figure 4.24. With this notation, next and previous half edges of e_{db} are e_{bc} e_{cd}, respectively. The adjacency relationships among half-edges e_1, e_2, e_3, e_3 is defined as follows:

e_1, e_2 and e_3, e_4 are face adjacent.

e_2, e_4 are cell adjacent

The half-edges are employed to give access to the adjacency relationships in the boundary surface. A pair of half-edges are directed have opposite directions.

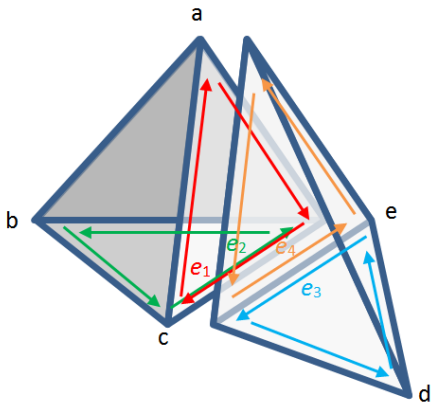


Figure 4.24

Star Edge: A star of edge consists of the all tetrahedron incident on that edge. It is represented as a linked list containing the pointer to each of the tetrahedron in its star.

These properties makes the half-edge data structure an appropriate choice to represent manifold surfaces (i.e. every edge is bordered by exactly two faces). Figure 4.25 shows the hierarchical relationship among the nodes of HE. The data structure was implemented using object-oriented concepts where the child node contains the pointer to its parent, thereby providing constant time access to its parent.

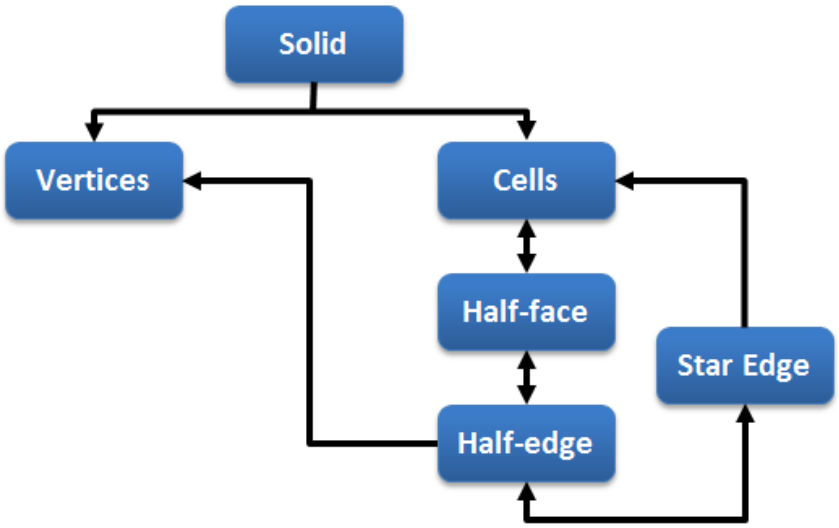


Figure 4.25: Arrangement of nodes in data structure

Computational Complexity HE data structure described in above section provide different tradeoffs between memory usage and time complexity of basic operations such as identifying a simplex, accessing its edges, and computing its stars. For instance when querying star of the edge, the operation will be linear in the number of tetrahedra incident to the edges, but constant time per-tetrahedron.

Considering the case of tetrahedral removal in a regularized simplicial complex $\mathcal{K}$ represented by the HE data structure, the time to update the neighborhood of a given topological element is proportional to the number of elements in that neighborhood, due to the adjacency and incidence relations stored in HE data structure.

4.6 Complexity analysis

We have measured the complexity of the algorithms in framework as a function of number of vertices (or pixels) in the section (or image). In the segmentation module, contour extraction runs on the subset of the pixels, while thresholding, morphological operators and boundary extraction algorithms run on the entirety of the image, giving the complexity bound as $O(p)$ in time and space, where p is the number of pixels of image. Table 4.1 show the complexity of each algorithm in segmentation module. As seen in the table, it is the component extraction algorithm that bounds the complexity of segmentation module, as it involves processing of each vertex of contours to compute geodesic distances between the nodes of contour graph. The complexity of computation of the geodesic distances algorithm is $O(zn^2)$, where n is the number of vertices of contour in a pair of adjacent sections and z represents the number of sections. The complexity of building the contour graph and extracting the largest component is $O(zc)$, where c is the number of contours in a section.

The reconstruction module is more efficient in terms of implementation and has its complexity bounded operations that run in linear time proportional to the neighbourhood of simplex. The complexity of the available incremental algorithm for 3D Delaunay triangulation is $O(zn^2)$, while reconstruction process is bounded by $O(zNT)$, where N is size of neighborhood of tetrahedron, and T is the number of tetrahedra in simplex. Computation of the contour nesting hierarchy in each section and orienting the contours in the correct directions takes $O(n \log n)$ time, where n is the number of vertices in a pair of adjacent sections. The space complexity of the module is $O(t)$, where t is the number of triangles in the pair of adjacent sections.

The correction module is also bounded by computation of geodesic distance between the nodes of contour graph as in the component extraction algorithm, which gives its complexity as $O(zn^2)$ where n is the number of vertices of contour in a pair of adjacent sections and z represents the number of sections.

Table 4.1: Complexity of Segmentation module

		Complexity
Binary Image Extraction		$O(zp)$
Morphological Operations		$O(zp)$
Boundary Extraction		$O(zp)$
Contour Extraction		$O(zp)$
Component Extraction	Computing geodesic distances	$O(zn^2)$
	Component Graph	$O(zc + E)$
	Component Extraction	$O(zc)$
Total		$O(zn^2)$

Table 4.2: Complexity of Corrections module

	Complexity
Computing geodesic distances	$O(zn^2)$
Building Connections Graph	$O(zc)$
Building Component Graph	$O(zc + E)$
Computing MST	$O(E \lg C)$
Rasterized Operations	$O(zp)$
Boundary Extraction	$O(zp)$
Component Extraction	$O(zn^2)$
Total	$O(zn^2)$

Table 4.3: Complexity of Reconstruction module

	Complexity
Computing 3D Delaunay Triangulation	$O(zn^2)$
Reconstruction of Surface	$O(zNT)$
Building overall mesh	$O(zn)$
Total	$O(zn^2)$

Table 4.4: Legend

z	number of slices (images)
p	number of pixels in a slice
n	number of vertices in a pair of slices
c	number of contours in a pair of slices
E	number of edges in graph used in operation
C	number of components in connections graph
N	size of neighborhood of tetrahedron
T	number of tetrahedrons in DT

Chapter 5

Results

In this chapter, we present the results obtained with the our framework on the given dataset. To facilitate its use, we have worked the algorithms into a full-fledged software tool. Each module of the framework was implemented using an object-oriented approach making it easy to establish the correspondence in various data structures and adapting the code to rapid changes and testing demands. The interchangeability provided by abstract and derived classes allowed easy representations of the input data, and support for multiple output data formats with minimal source code changes.

The complete framework, data structure and methods to access its information were implemented in C++ with Visual Studio as IDE. The software was compiled for Microsoft Windows 7 using Microsoft VC++ compiler. Because of our choice for the specific programming language and processing environment as described earlier, the tool is, in principle, computer-platform independent. By optimizing the implementation of the algorithms we were able to achieve acceptable processing speeds, details of which are described later in the section.

For visualization purposes, we developed our own 3D viewer using VTK (Visualization Toolkit) and OpenGL library, that smoothens the reconstructed mesh using a laplacian filter. We have tested our results on HP workstation xw6600, equipped with an Intel Xeon quad-core processor, 8 gigabytes of memory, and a Nvidia GeForce 8800 graphics card with 1 gigabytes of video memory.

We now begin with the reconstruction of Dataset I along the flow of pipeline in the framework and show how each module performs in the framework.

Segmentation We first analyse the raw images of the dataset for noise which helps to decide the threshold to use for the segmentation process. Figure 5.1 shows the non background (inherited noise and structure) pixels in the raw image of the Dataset I. Based on the noise present, we set an initial estimate of threshold, g_0=22. We further manipulate the regions in images to extract the small structures using visual tools.

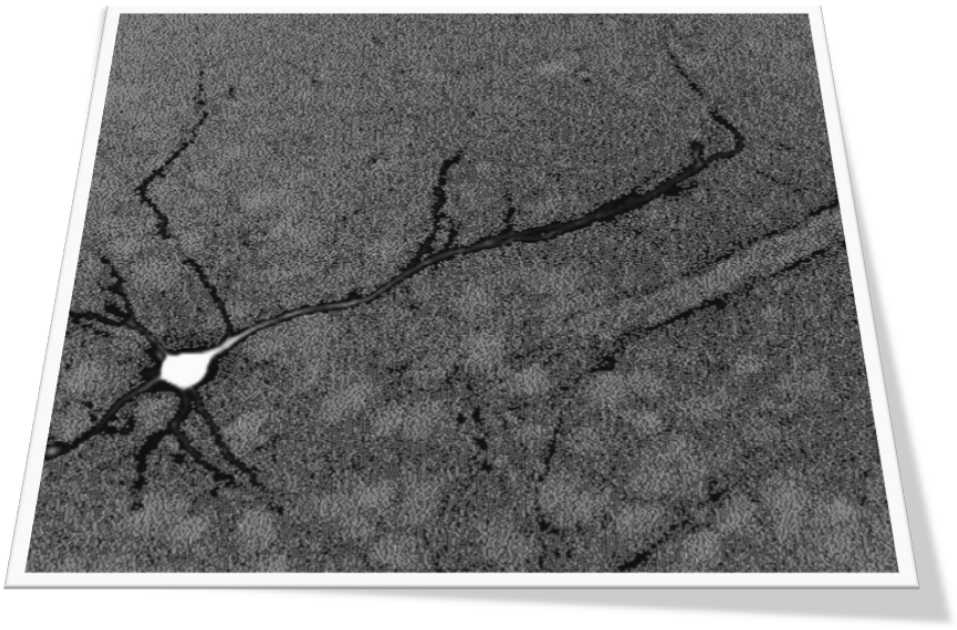
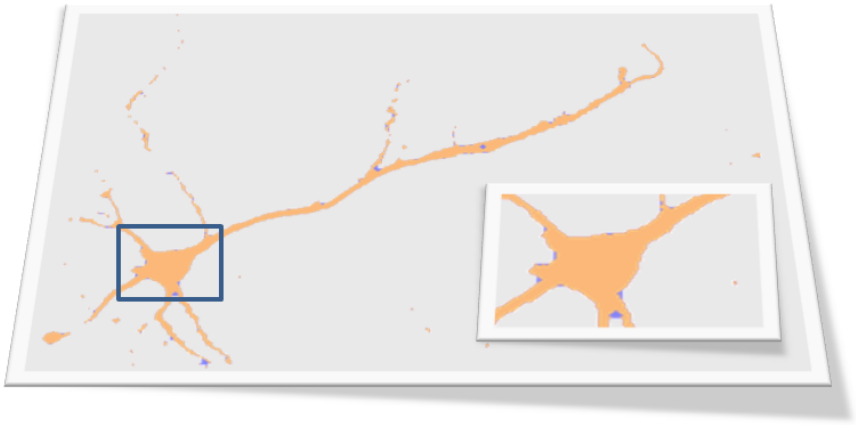
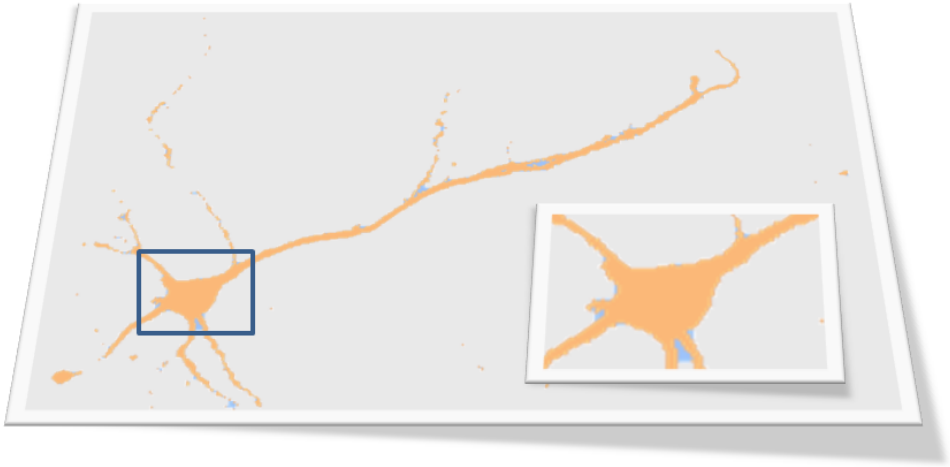


Figure 5.1: Noise represented by grey pixels present in the microscopy images

Morphological Operations Once we sufficiently extract the information in thresholding process, morphological operations are applied on the images to patch holes and vacant neighbourhood pixels in the images. The results of morphological operations with a 4-connective template and 8-connective template are shown in Figure 5.2. We used two operations of openings followed by two operations of closing operator in our result. Based on the visual analysis, we chose 8-connective template for further processing in the framework.



(a)



(b)

Figure 5.2: Newly added pixels (marked in blue) with morphological operations applied to segmented image with (a) 4-connective template (b) 8-connective template

Contour Extraction and Component Analysis Next, we trace out the boundaries in the images to form the contours stack. To remove distant sparse contours inherited from the noise, we use the component analysis algorithm. Figure 5.4a shows the contours selected (marked in blue) for filtration with the optimal threshold value selected from the component analysis plot (Figure 5.3). From the plot, we can see that two graphs (largest component size and number of components) stabilize over the range 20-24. We selected 22 as the threshold based on visual analysis of the contour stack. This contour filtration step has proved to be very effective in reducing the size of data, which is critical towards increasing the computational efficiency of our reconstruction module. The reconstructed mesh for the extracted contours is shown in Figure 5.4b with components of mesh from noisy contours marked in blue color.

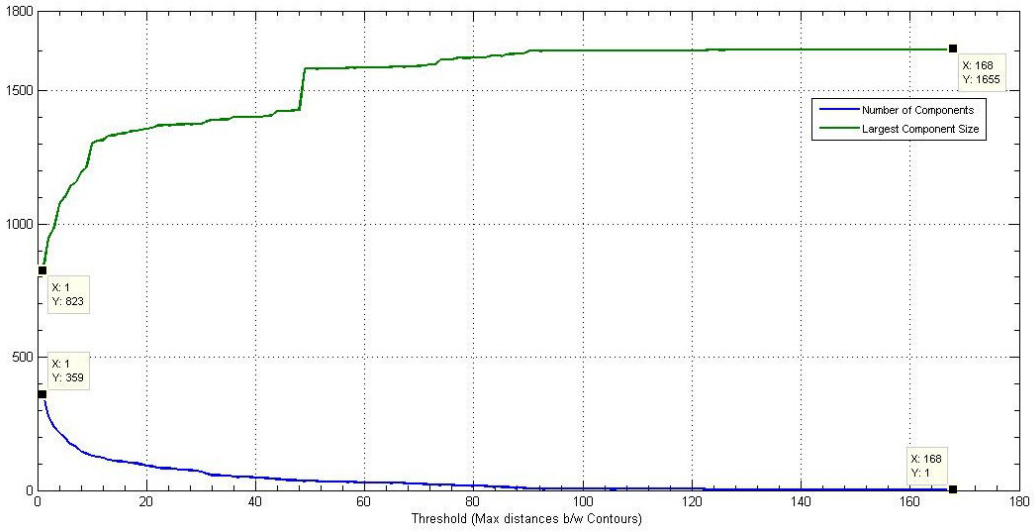
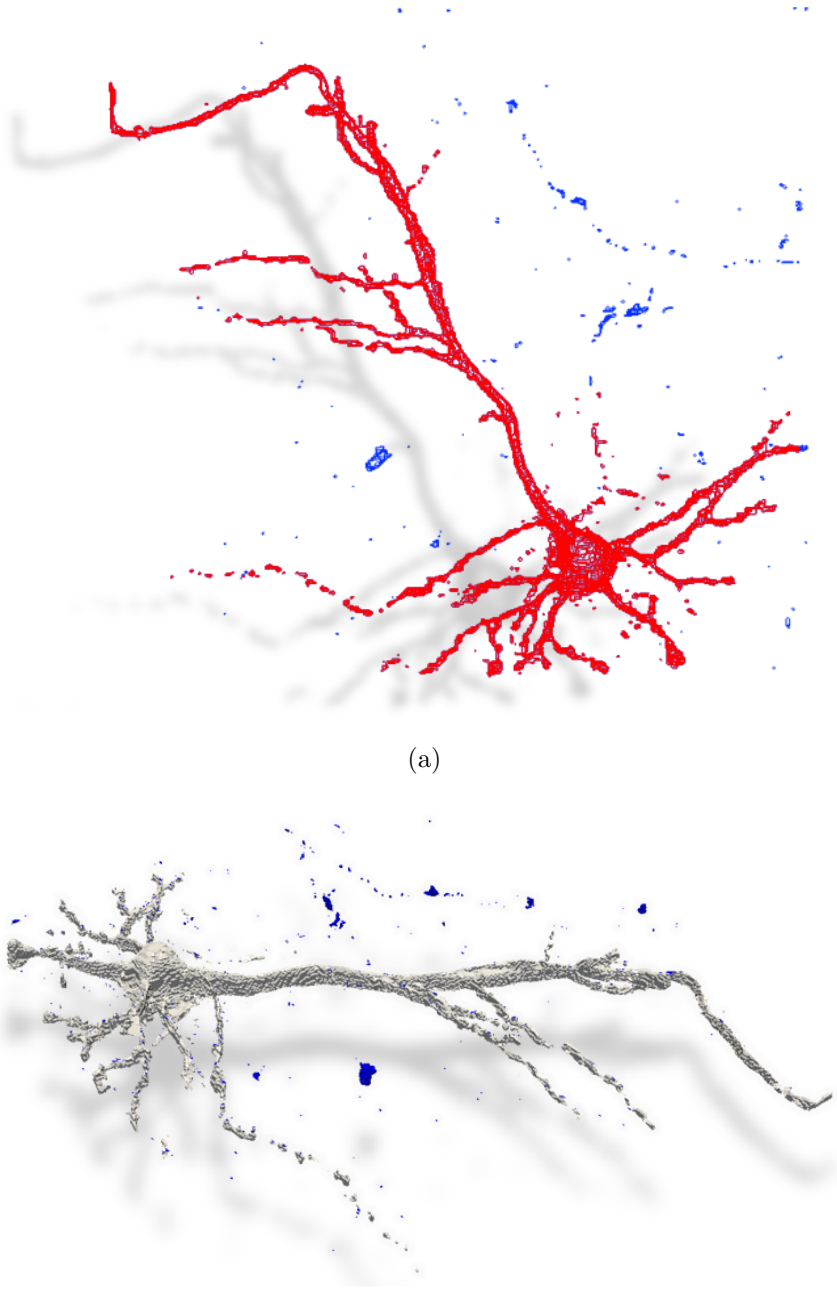


Figure 5.3: Component Analysis graph for the Dataset I



(a)

(b)

Figure 5.4: (a) Contours(marked in blue) selected with the threshold, $t_0 = 22$ in component extraction (b) Reconstructed mesh with components from selected contours in (a) marked in blue color

Corrections As observed in Figure 5.4b, the reconstructed mesh still has disconnected components in the dendritic branches, thus creating the necessity of the correction module. Figure 5.6 shows how the connections are linked in the contour stack and correspondingly in the reconstructed mesh from those contour stack for threshold $t_1 = 10$. The threshold selection is again done through component analysis plot (Figure 5.5) on the component graph of extracted contour stack. As seen in the plot, the optimal range of threshold t_1 is 10-13. Figure 5.7a shows the final reconstructed mesh where it is extended correctly as a connected component in the dendritic branches with newly created connections marked in blue color.

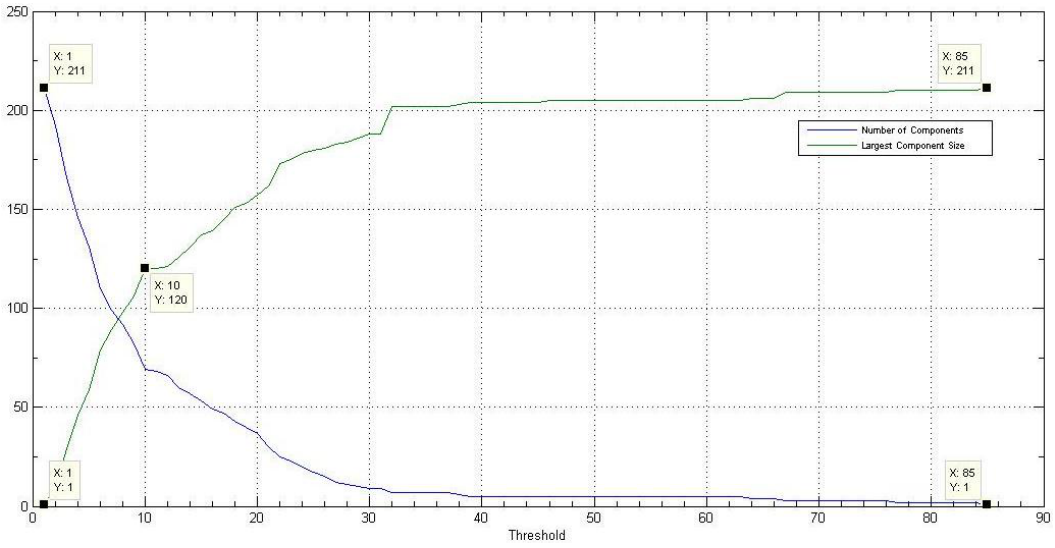
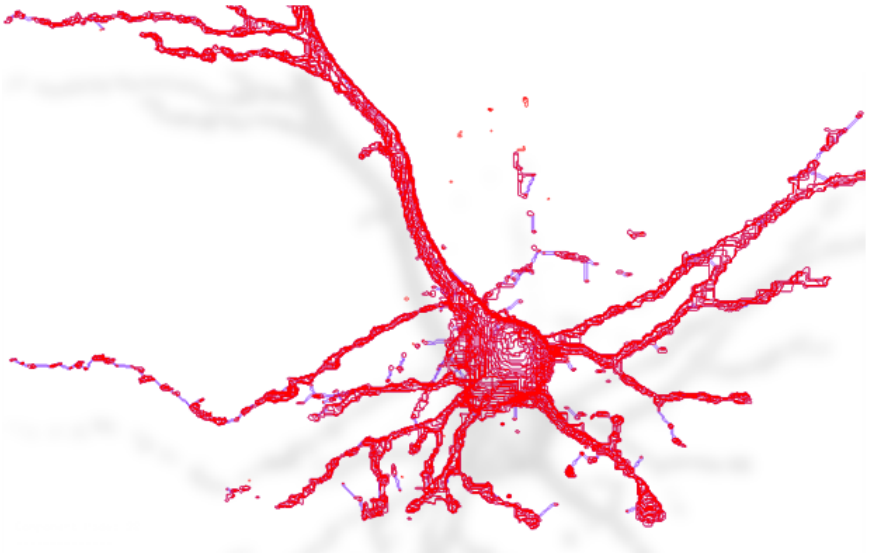
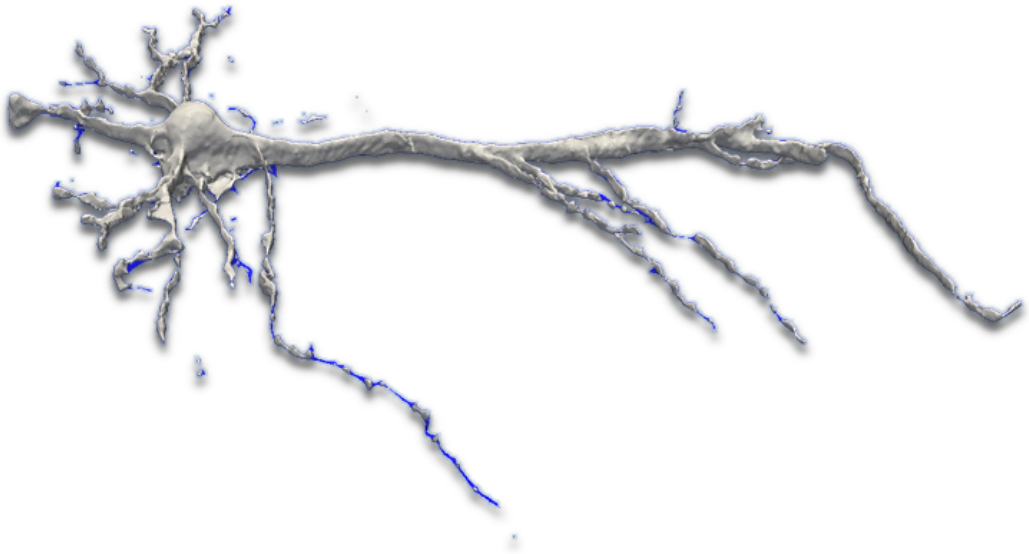


Figure 5.5: Plot of component analysis in corrections module



(a)



(b)

Figure 5.6: (a) Connections established (marked in blue) in contours stack (b) Reconstructed mesh with newly created connections marked in blue

Reconstruction Figure 5.7b shows how variation of β parameter affects the connection in the reconstructed mesh with extra correspondences generated marked in blue color as compared to correspondence from $\beta = 1$. In our result, we have used consistent value of β-parameter across the contour stack, but it is possible to vary the values of β across different pairs of sections to obtain different results.

(a)

(b)

Figure 5.7: (a) Reconstructed Mesh for $\beta = 1$ (b) Reconstructed Mesh with $\beta = 3$

Performance of the framework The computation time for the above dataset are presented in Table 5.1 an 5.2 with parallel execution done on four cores. The total time

Table 5.1: Performance of Segmentation module (in sec)

	Linear	Parallel
Binary Image Extraction	3.760	2.153
Morphological Operations	3.682	2.715
Boundary Extraction	3.120	1.809
Contour Tracing	3.900	1.982
Computing geodesic distances	213.378	
Component Graph	3.81	
Component Extraction	1.82	
Total	230.35	227.667

Table 5.2: Performance of Reconstruction module (in sec)

	Linear	Parallel
Computing 3D Delaunay Triangulation	23.83	6.83
Reconstruction of Surface	771.72	244.62
Building overall mesh	0.967	
Total	796.51	252.417

Table 5.3: Performance of Corrections module (in sec)

	Linear
Computing geodesic distances	54.366
Building Connections Graph	1.856
Building Component Graph	0.07
Computing MST	0.05
Rasterized Operations	1.378
Boundary Extraction	4.81
Component Extraction	81.463
Total	143.99

taken by our framework for the reconstruction of Dataset I is 624.07 sec $\approx$ 10 minutes. On comparison with commercial softwares such as NeuroLucida, it takes an average time of 4-6 hours to trace the boundaries in the dataset and generate the surface of the given dataset. This performance gain shows the significance of our framework in terms of reconstruction of neuronal structures.

Error Analysis To evaluate the reconstruction from our framework, we reconstructed a virtual neuron model and compared the results of reconstruction with the original mesh. We used the 3d modelling software "Autodesk 3ds Max" to design a virtual neuron model. The model contains a soma, axon and flow of dendrite in 3d space (as shown in Figure 5.8). Following the design, we computed the cross sections (Figure 5.9a) of model at intervals in z scale, where we used normalized value for computing interval length to maintain the aspect ratio of model. Further to introduce the partial effect of adjacent images in the focused image, we took average of intensities of pixels in adjacent sections to form a new image. To achieve the partial volume effect and intensity decay,

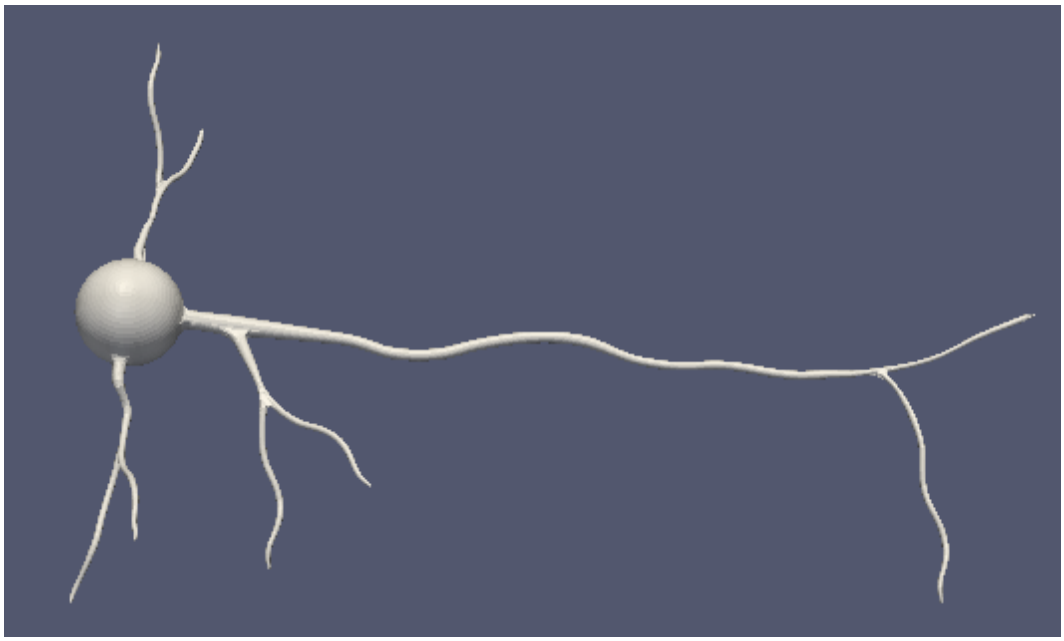
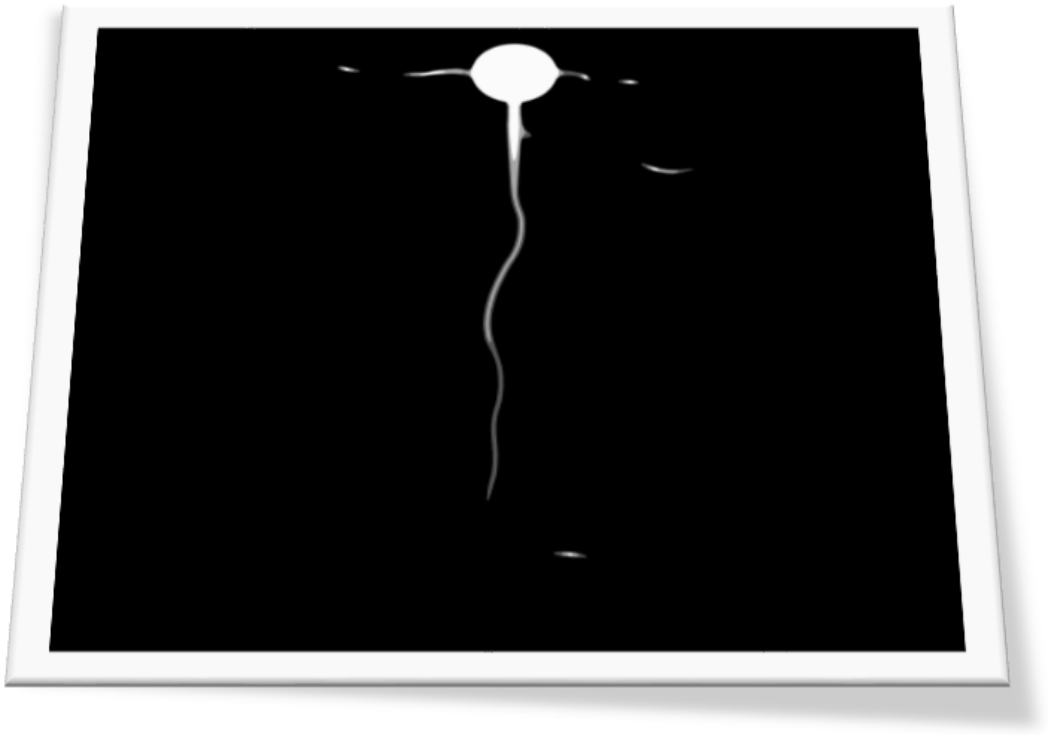
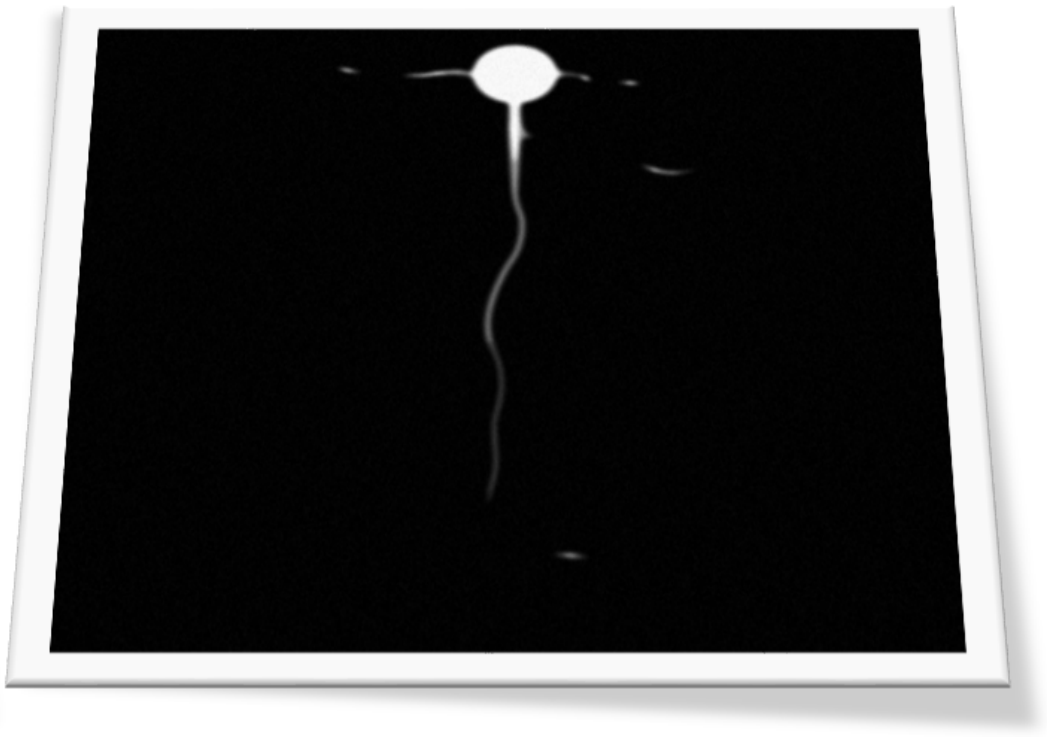


Figure 5.8: Virtual Neuron Model

we applied the proprietary "Lens blur" effect in Adobe Photoshop to the images (Figure 5.9b). This process helped us achieve realistic challenges often present in the images due to uneven dye distribution and point spread of microscope. Finally, we fed these images to our framework and obtained the reconstructed mesh (Figure 5.10). For error analysis, we used a proprietary tool called Metro[10] which numerically compares two triangular meshes by evaluating the difference between the two meshes on the basis of the approximation error. It uses a mean distance as the measure of approximation error



(a)



(b)

Figure 5.9: (a) cross sections of the virtual neuron model (b) with Lens Blur effect

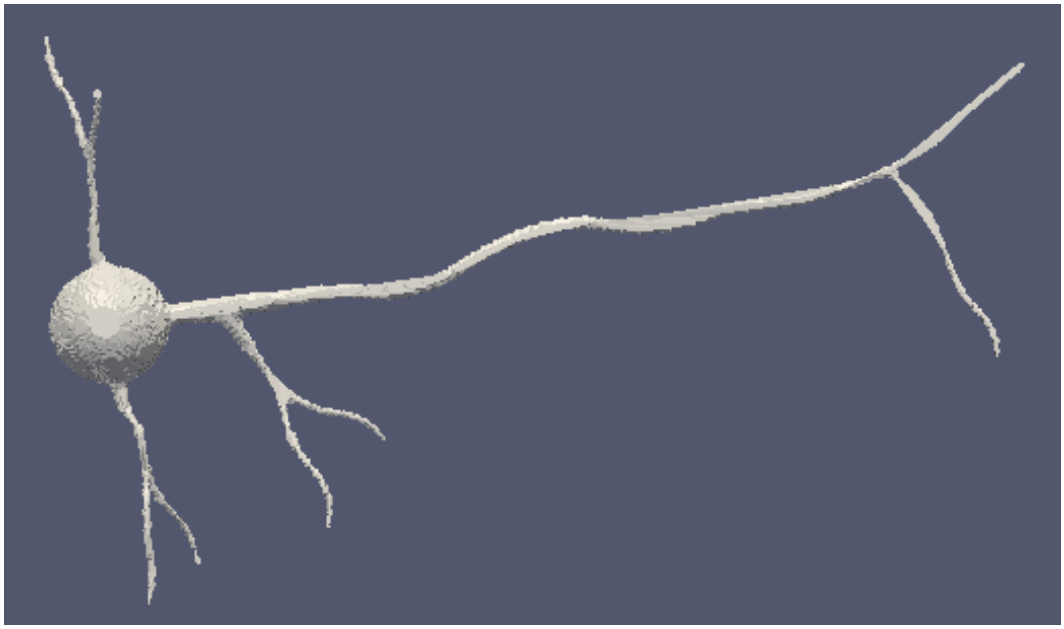


Figure 5.10: Reconstruction of Virtual Neuron Model

which is defined as

$$E_m = \frac{1}{|S_1|} \int_{S_1} e(p, S_2) ds$$

where S_1, S_2 are the meshes being compared and $e(p, S_2)$ is the distance of point p in S_1 to nearest point in S_2. Metro adopts an approximate approach based on surface sampling and the computation of point to surface distances. The surface of the first mesh is sampled, and for each elementary surface point, distance to the second mesh is computed. Surface sampling in Metro is achieved by scanning of triangular faces with a user-selected sampling resolution.

The output from the Metro tool consists of numerical data on input meshes characteristics (mesh volume, feature edges total length, diagonal of the bounding box); and the mean and maximum distances between meshes. For our use, we have used mean and RMS distance as the two measures to evaluate our reconstruction. We used both reconstructed mesh (backward) and original mesh(forward) as the sampled mesh and evaluated the results. We computed percentage error on the non-sampled mesh as value of mean distance with respect to diagonal of the bounding volume of sampled mesh. With the given results in Table 5.4, we can conclude that the reconstruction accuracy of the framework is optimal for the given purpose.

Table 5.4: Error Analysis Results from Metro tool

	Mean	RMS	Percentage Error
Forward Distance	2.41	3.20	0.4
Backward Distance	2.55	3.08	0.4

Chapter 6

Conclusion and Future Work

The success of our approach to the reconstruction problem depends critically on the user's expertise to derive the solution, that is judged by its quality and accuracy of the reconstructed 3D model. Although, for reasons mentioned in the chapter 1, it seems impossible to eliminate all user interaction and it remains a challenge for us to automate the framework further and investigate its potential for fully automated neuron reconstruction. Our approach to treat the disconnected fragments problem in the reconstruction of neuron structures offers flexibility in the choice of degree of connections to be made. Moreover, the fact that the resulting mesh is free of singularities enables the resulting models to be used in simulations without any need for postprocessing. We feel that results from our framework were more than adequate for the given dataset and can be applied for functional visualization of other neuron structures. From a neuroscientists standpoint, several characteristics are important, including the centroid of a soma, its volume, its surface area, pattern in the dendritic connectivity and topology of such structures, which can be easily derived from our reconstructed mesh. We would also like to mention the comments we have received from Prof. S.K. Sikdar who is a neuroscientist working within Indian Institute of Science, Bangalore. "I wish to mention that the method has been found to be useful in the preliminary analysis of dendritic shaft and spines of a single neuron filled with a dye, where the 2-photon confocal microscopy was used to reconstruct the images. The preliminary studies confirm the usefulness of the

method and I am confident that this will prove to be useful for detailed neuronal reconstructions in the future. I consider this work to be very challenging and it is probably the first time that such detailed attempts to computationally reconstruct the image of a neuron has been attempted in the country, independently.".

The problem still left to work out is the detection of crossovers i.e when two neuron structures cross each other very closely in the axial z-dimension and cause a overlap of signal in the images. These overlaps are difficult to detect in raw images without algorithms that require a lot of computational overhead. Further work in our plans is to derive the correspondence parameter (β) for each region or contour from the component analysis graph, instead of using unified β parameter across pair of adjacent slices.

From the above discussion, we conclude that our semiautomatic neurite reconstruction technique yields a significant improvement over fully manual tracing(or reconstruction) methods or other approaches by the conjunction of an accelerated and parallel implementation of processes, and corrigible results due to parameter initialized algorithms.

References

[1] N. Amenta, M. Bern, "Surface reconstruction by Voronoi filtering", Discrete and Computational Geometry, pp.481-504. 1999.

[2] N. Amenta, S. Choi, R. Kolluri, "The power crust, unions of balls, and the medial axis transform", Computational Geometry: Theory and Applications, pp.127-153, 2001.

[3] J. D. Boissonnat, "Shape reconstruction from planar cross sections", Computer Vision, Graphics, and Image Processing, pp.1-29, 1988.

[4] M. de Berg; O. Cheong, M. van Kreveld, M, Overmars, "Computational Geometry: Algorithms and Applications", Springer-Verlag, 2008.

[5] C.L. Bajaj, E.J. Coyle, K.N. Lin, "Arbitrary topology shape reconstruction from planar cross sections", Graphical Models and Image Processing, pp. 524-543, 1996.

[6] Y. Bresler, J.A. Fessler, and A. Macovski, "A Bayesian approach to reconstruction from incomplete projections of a multiple object 3d domain. IEEE Transaction on Pattern Analysis And Machince Intelligence, pp.840-858, 1989.

[7] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, "The ball-pivoting algorithm for surface reconstruction", IEEE Transactions on Visualization and Computer Graphics, pp.349-359, 1999.

[8] J. E. Bresenham, "Algorithm for computer control of a digital plotter", IBM Systems Journal, pp.25-30, 1965.

[9] P.J. Broser, R. Schulte, A. Roth, F. Helmchen, S. Lang, G. Wittum, and B. Sakmann, "Nonlinear anisotropic diffusion filtering of threedimensional image data from two-photon microscopy", Journal of Biomedical Optics, pp.1253-1264, 2004.

[10] P. Cignoniy, C. Rocchiniz and R. Scopignox, "Metro:measuring error on simpli ed surfaces", Technical Note (Short contribution), Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Pisa, Italy, 1998.

[11] M. J. Carlotto, "Histogram analysis using a scale-space approach", IEEE Transaction of Pattern Analanysis and Machince Intelligence, pp.121-129, 1997.

[12] S. W. Cheng and T. Dey, "Improved constructions of delaunay-based contour surfaces", ACM Symposium on Solid Modeling and Applications", pp.322-323, 1999.

[13] H.N. Christiansen and T.W. Sederberg, "Conversion of complex contour line definitions into polygonal element mosaics", Computer Graphics, SIGGRAPH, pp.187-192, 1978.

[14] T.K. Dey, S. Goswami, "Tight Cocone: A water-tight surface reconstructor", Journal of Computing and Information Science in Engineering", pp. 302-307, 2003.

[15] A. Dima, M. Scholz, and K. Obermayer, "Automatic segmentation and skeletonization of neurons from confocal microscopy images based on the 3D wavelet transform", IEEE Transaction on Image Processing, pp.790-801, 2002.

[16] H. Edelsbrunner and E.P. Mucke, "Three-dimensional alpha shapes". ACM Transaction on Graphics, pp.43-72, 1994.

[17] A. B. Ekoule, F. C. Peyrin, and C. L. Odet, "A triangulation algorithm from arbitrary shaped multiple planar contours", ACM Transactions on Graphics, pp.182-199, 1991.

[18] J.F. Evers, S. Schmitt, M. Sibila, and C. Duch, "Progress in functional neuroanatomy: precise automatic geometric reconstruction of neuronal morphology from confocal image stacks", Journal of Neurophysiology, pp.2331-2342, 2005.

[19] T.J. Fellers, M.W. Davidson, "Introduction to Confocal Microscopy". Olympus Fluoview Resource Center. National High Magnetic Field Laboratory, 2007

[20] F. Fleuret, P. Fua, "Dendrite Tracking in Microscopic Images using Minimum Spanning Trees and Localized E-M", Technical Report, Computer Vision Lab, EPFL, 2006.

[21] B. Geiger, "Three dimensional modeling of human organs and its application to diagnosis and surgical planning", Technical Report, INRIA, Sophia-Antipolis, France, 1993.

[22] C. Giertsen, A. Halvorsen, and P. R. Flood, "Graph-directed modelling from serial sections", The Visual Computer, pp. 284-290, 1990.

[23] R.C. Gonzalez, R. E. Woods, "Digital image processing", 2nd edition, Upper Saddle River, New Jersey, Prentice Hall, 2002.

[24] W. He, T.A. Hamilton, A.R. Cohen, T.J. Holmes, C. Pace, D.H. Szarowski, J.N. Turner and B. Roysam, "Automated Three-Dimensional Tracing of Neurons in Confocal and Brightfield Images. Microscopy and Microanalysis", pp 296-310, 2003.

[25] M. Jones, and M. Chen, "A new approach to the construction of surfaces from contour data". Computer Graphics Forum, pp.75-84, 1994.

[26] E. Keppel, "Approximating complex surfaces by triangulation of contour lines", IBM Journal of Research and Development, pp.2-11, 1975.

[27] K.A. Al-Kofahi, S. Lasek, D.H. Szarowski, C.J. Pace, and G. Nagy, "Rapid automated three-dimensional tracing of neurons from confocal image stacks", IEEE Transactions on Informations Technology in Biomedicine, pp.171-187, June 2002.

[28] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", Proceedings of the American Mathematical Society, pp.48-50, 1956.

[29] J. N. Kapur, P. K. Sahoo, A. K. C. Wong, "A new method for gray level picture thresholding using the entropy of the histogram. Computer Vision, Graphics, and Image Processing, pp.273-285, 1985.

[30] D. Levin, "Multidimensional reconstruction by set-valued approximation", IMA Journal of Numerical Analysis, pp.173-184, 1986.

[31] W. Lorenson, and H. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", ACM Computer Graphics, pp.163-169, 1987.

[32] C. K. Leung and F. K. Lam, "Performance analysis of a class of iterative image thresholding algorithms", Pattern Recognition, pp.1523-1530, 1996.

[33] E. Meijering, M. Jacob, J.C.F. Sarria, M. Unser, "A Novel Approach to Neurite Tracing in Fluorscence Microscopy images", International Conference on Signal and Image Processing, pp 491-495, 2003.

[34] D. Meyers, S. Skinner, K. Sloan, "Surfaces from contours", ACM Transactions on Graphics, pp.228258, 1992.

[35] D. Meyers, "Reconstruction of surfaces from planar contours", Doctoral dissertation, University of Washington, 1994.

[36] L.G. Nonato, A. Castelo, J.P.P. Campos, H.H. Biscaro, R. Minghim, "Tetrahedron Topological Characterization with Application in Volumetric Reconstruction", International Journal of Shape Modeling, 2005

[37] L.G. Nonato, A.J.C. Vargas, R. Minghim, M.C.F. Oliveira, "Beta-connection: Generating a family of models from planar cross sections", ACM Transactions on Graphics, pp.1239-1258, 2005

[38] L.G. Nonato, R. Minghim, M.C.F. Oliveira and G.Tavares, "A novel approach for Delaunay 3D reconstruction with a comparative analysis in the light of applications", Computer Graphics Forum, pp 161-174, 2001.

[39] J. C. Olivo, "Automatic threshold selection using the wavelet transform", Graphical Models and Image Processing pp. 205-218, 1994.

[40] N. Otsu, "A threshold selection method from gray level histogram". IEEE Transactions on Systems, Man, and Cybernetics, pp.6266, 1979.

[41] N. R. Pal, "On minimum cross-entropy thresholding", Pattern Recognition, pp.575-580, 1996.

[42] H. Peng, F. Long and G. Myers, "Automatic 3D neuron tracing using all-path pruning", Bioinformatics, pp 239-247, 2011.

[43] T. W. Ridler and S. Calvard, "Picture thresholding using an iterative selection method", IEEE Transaction on Systems, Man and Cybernactics, pp.630-632, 1978.

[44] A. Rodriguez, D. Ehlenberger, K. Kelliher, M. Einstein, S.C. Henderson, J.H. Morrison, P.R. Hof, and S.L. Wearne, "Automated reconstruction of three-dimensional neuronal morphology from laser scanning microscopy images", Methods, pp.94-105, 2003.

[45] Amir Sadeghipiur, "Algorithms of rutomatic reconstruction of neurons form the confocal images", Masters thesis, 2008.

[46] S. Schmitt, J.F. Evers, C. Duch, M. Scholz, and K. Obermayer, "New methods for the computer-assisted 3D reconstruction of neurons from confocal image stacks", NeuroImage, pp.1283-1298, 2004.

[47] M. I. Sezan, "A peak detection algorithm and its application to histogram-based image data reduction," Graphical Models and Image Processing, pp.47-59, 1985.

[48] M. Shantz, "Surface definition for branching, contour-defined objects", Computer Graphics, pp.242-270, 1981.

[49] A. Santamaria, I. Kakadiari, "Automatic Morphological Reconstruction of Neurons from Optical Imaging", Micscoscopy Image Analysis and Applications in Biology Workshop", 2007.

[50] Y. Shinagawa, T.L. Kunii, and Y. L. Kergosien, "Surface coding based on Morse theory", IEEE Comuter Graphics and Applications, pp.66-78, 1991.

[51] B.I. Soroka, "Generalized cones from serial sections". Computer Graphics and Image Processing, pp.154-166, 1981.

[52] K.R. Sloan Jr. and J. Painter, "Pessimal guesses may be optimal: A counterintuitive search result", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.949-955, 1988.

[53] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging, pp.146-165, 2004.

[54] C. Uehara, C. Colbert, P. Saggau, I.Kakadiaris, "Towards automatic reconstruction of dendrite morphology from live neurons", IEEE Engineering in Medicine and Biology Society, 2004.

[55] S. Urban, S.M. OMalley, B. Walsh, A. Santamara-Pang, P. Saggau, C. Colbert and I. A. Kakadiaris, "Automatic Reconstruction of Dendrite Morphology from Optical Section Stacks", Computer Vision approaches to Medical Image Analysis, Lecture Notes in Computer Science, pp 190-201, 2006

[56] J.M. White and G.D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction", IBM Journal of Research and Development. pp.400-411, 1983.

[57] S.L. Wearne, A. Rodriguez, D.B. Ehlenrger, A.B. Rocher, S.C. Henderson, and P.R. Hof, "New techniques for imaging, digitization and analysis of three-dimensional neural morphology on multiple scales", Neuroscience, 2005.

[58] T. Zhao, J. Xie, F. Amat, N. Clack, P. Ahammad, H. Peng, F. Long, E. Myers, "Automated Reconstruction of Neuronal Morphology Based on Local Geometrical and Global Structural Models", Neuroinform, pp 247-261, 2011.

[59] Y. Zhang, X. Zhou, J. Lu, J. Lichtman, D. Adjeroh and ST. Wong, "3D Axon structure extraction and analysis in confocal fluorescence microscopy images", Neural computation, pp.1899-1927, 2008.