An Interactive Framework for Reconstructing 3D Neuronal Structures

Kanuj Kumar [a]
kanujkumar@csa.iisc.ernet.in

Vijay Natarajan [a, b]
vijayn@csa.iisc.ernet.in

S. K. Sikdar [c]
sks@mbu.iisc.ernet.in

Kalyan V Srinivas [c]
kalyan0000@gmail.com

a Computer Science And Automation, Indian Institute of Science, Bangalore, India
b Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore, India
c Molecular Biophysics Unit, Indian Institute of Science, Bangalore, India

ABSTRACT

Neuron reconstruction is a complex and tedious task and neurobiologists today have to rely on time-consuming manual methods. Methods that entail manual tracing may introduce systematic inaccuracies. This necessitates the use of automated methods for reconstruction of neuronal structures. Despite recent advancements, the automation of reconstruction process either does not exhibit robustness against noise of microscopy images or fails to capture precise dendritic structures. This motivates the development of semi-automated methods that require crucial but minimal expert user input. In this paper, we present a fast and interactive framework for reconstruction of neuronal structures that employs automatic methods while allowing a user to provide expert input if the results are unsatisfactory. The framework is designed as a multi-stage pipeline, where in the user provides numerical input parameters to guide the reconstruction process and validates the output of automated methods visually. The user is also assisted by the framework in calculating the optimal values of required parameters for reconstruction. The framework is also able to handle discontinuities and produce a connected geometric model of the neuron structure.

Keywords

3D Neuron Reconstruction, Interactive Framework

1. INTRODUCTION

The reconstruction of a neuronal structure using a semi-automatic or manual system has always been a time consuming task because of the complex morphologies of neuronal cells and noise imposed by the imaging conditions. While a few techniques of varying degrees of automation have been suggested to reconstruct neuron morphologies in the past [9, 11, 12, 13], their precision is limited by quantization errors arising from insufficient imaging resolution. Thus, the objective of decreasing the user interaction often conflict to ensure the accuracy and the topological correctness of the models. Considering the incomplete results of the automatic procedures, neuro-biologists 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, USA) 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 data-sets becomes overly tedious and time consuming, thus necessitating the need of flexible system. Framework introduced in this paper uses automated methods with driving parameters to guide the reconstruction of neuronal surface. Further, we aim to reconstruct a connected surface even in the presence of discontinuities in the input images caused by uneven dye distribution. The framework also supports an automated pipeline with calculation of optimal driving parameters. In

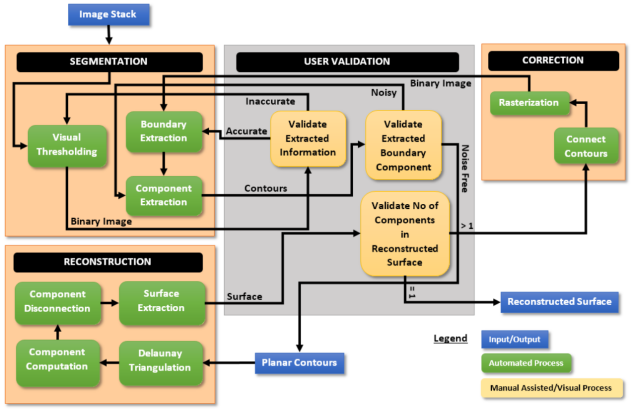


Figure 1: A flexible framework for reconstructing the neuronal surface from a stack of 2D images.

ICVGIP '14, December 14-18, 2014, Bangalore, India

http://dx.doi.org/10.1145/2683483.2683502

particular, key contributions of this paper include

- a flexible system for extracting boundary contours of the neuron within each 2D slice
- a modification of the beta-connection algorithm to reconstruct the neuronal surface from a set of contours that includes a correction component to ensure that the output is connected
- an integrated framework that includes the above methods for neuronal surface reconstruction with minimal but critical user input

2. RELATED WORK

Several algorithms have been proposed for 3D reconstruction of neurons from optical microscopy data, which can be broadly divided into two categories as centerline-extraction based and boundary-extraction based.

Centerline-extraction based methods[5, 8, 11, 12, 13]. 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 irregularities of dendrites' surfaces are ignored during extraction of centerline. Also, this approach is computationally intensive since most of the operations are performed on every voxel. Moreover, in contrast to boundary-extraction based methods, it involves both local and global characterization while modeling neurites. Few of the methods categorized under this approach are described below. Zhao et al.[13] 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 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 where 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 method fails to trace complete segments when there are discontinuities in neurites.

Fleuret et al.[5] describe a method for tracing of dendrites pattern by computing an optimal tree using a modified minimum spanned tree procedure 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 problem with the approach involves high computational cost, especially to handle the list of all possible voxels.

Urban et al.[11] proposed a skeletonisation algorithm to produce the segmentation of the structures in the images. The authors explicitly determine the voxels that belong to the pipette and the soma employing the information that these two correspond to brightest features (voxels) in the images. To extract the medial axis of the segmented neuron, they employ a distance transform based skeletonisation method that guarantees to be tree-structured, which is then used to cull false dendrites from the neuron. An approximating spline is then fitted 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 using dendrite widths along each spline. The framework proposed by the authors is unable to handle apparent dendrite gaps and irregularities in dendrites.

Boundary-extraction based methods[4, 6, 10]. 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. It requires further post-processing to link the broken edges. The linking algorithms may introduce unnecessary ambiguity and incorrect links of noisy data. Recent work[4] has adapted the approach to include global information to resolve these ambiguities.

Uehara et al.[10] describe a reconstruction algorithm based on a wave propagation. The algorithm generates a field indicating the probability of each voxel belonging to a cylindrical structure. A digital wave is then propagated through this field to provide dendrite paths. However, the multi-scale gradient analysis used is computationally expensive for large volume data.

Hamilton et al.[6] 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 minimum spanning tree. Resulting fragments are connected through a semi-interactive method where user specifies the region of error. The surface is computed by checking presence of vacant voxel in neighborhood connectivity of 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.

3. THE FRAMEWORK

The framework divides the task of reconstruction of the neuron structure from image data into three modules shown in Figure 1. The segmentation module extracts the boundary of neurite structures within each image. It starts with computation of boundary polygons based on locally adapted threshold for an image and computes a connected component to filter the noisy structures. The choice of input parameters is assisted via immediate visualization of results from the extraction process. The goal of the reconstruction module is to find a best surface consistent with the extracted polygons in segmentation module. It achieves this via construction of a triangulation from the extracted boundary polygons. This triangulation is extended further to create

correspondence between polygons of adjacent sections. Discontinuities, often emerging from insufficient resolution of images are handled in correction module. Connection module first computes the minimum spanning tree (MST) over the connected-contour graph and then establish a link between contours if the corresponding pair is connected by an edge in the MST. This linking of contours is represented in binary image via rasterization of shortest line segment connecting these contours. New binary images are then fed as input to segmentation module to recompute the extracted contours.

To demonstrate the performance of the framework, we

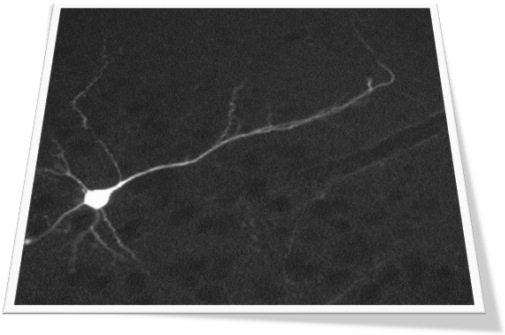


Figure 2: Confocal microscopy scan of subicular pyramidal neuron cell

have acquired a dataset that consists of noise and artifacts inherent from the mechanism used for imaging. As evident, the dataset (Figure 2) consists of diffraction effects causing a spread of a point object in the image characterized by its point spread function. Also, variable contrast is exhibited in images due to uneven dye distribution within the cell causing apparent discontinuity in structures. Also, many features of interest (such as thin dendrites) are at the limit of imaging resolution.

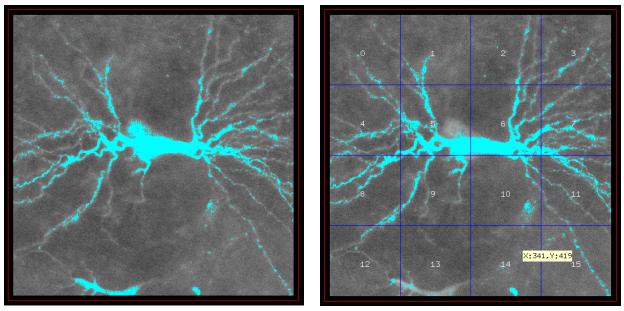
Dataset. The dataset (Figure 2) shows the structure of single subicular pyramidal neuron cells from hippocampal region in the rat brain slice preparation. Images were acquired with confocal microscopy with the neuron filled with Biocytin through a patch pipette electrode and stained with streptavidin conjugated with Fluorescein fluorophor dye. The dataset consists of 47 sections with voxel resolution of $0.65\mu m$ in the $x-y$ axis and $0.5\mu m$ in the z axis. Excitation wavelength was set to $488nm$ with index of refraction corresponding to that of water. The depth of image is 8 bits/pixel with resolution of 512×512 in $x-y$ scale.

3.1 SEGMENTATION

The segmentation module was designed to enable fast extraction and visualization of the structures from the images. Based on the approaches used, it is applicable to data obtained from any imaging modality. This module considers each image as a 2D matrix, where each component of the matrix represents the intensity value of a pixel. In this context, we represent the image stack as a function $I(x, y, z)$ that maps voxel coordinates onto their intensity values.

3.1.1 *VISUAL BINARY SEGMENTATION*

To segment the structures from these images we chose thresholding as it is one of the simplest and least computationally intensive technique for image segmentation. However, low imaging resolution and presence of arbitrary noise often acts contrary to choice of optimal threshold value. Thus, we incorporated visual tools in framework to produce a reliable segmentation. To ensure that all the possible signals are captured, framework first produce an initial overcomplete segmentation with threshold as the decay point of the peak in intensity histogram of the entire image. The segmented image is overlay-ed on source image and easily manipulated in real-time by varying the threshold value to extract more information. Figure 3a) shows the screen-shot of the visual tools incorporated in the framework. User is



(a) (b)

Figure 3: (a) Initial overcomplete segmentation (b) Local threshold selection with bounding box

also given the choice to specify local threshold for a resizeable bounding region (Figure 3b). This way user can even extract small structures which may not be otherwise covered with a global threshold for the entire image. Additionally, user can also cull objects or artifacts, within the specified region by choosing higher threshold for that region. The size of region is made configurable that can be expanded in z depth if user chooses to specify threshold for the set of images in the local region.

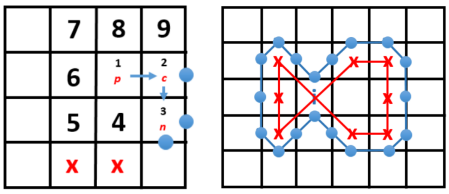
To remove artifacts such as presence of small holes or objects in the segmented image, framework provides user with choice of morphological operations to be applied on these segmented image. Each morphological operation comes either with 8-connective or 4-connective neighborhood templates. User can visually verify the application of morphological operation on these images and has the option of choosing the template and the number of operations as tunable features.

3.1.2 *BOUNDARY EXTRACTION AND CONTOUR TRACING*

The reconstruction module requires the boundary of the extracted structure. In order to extract the boundary pixels, we check the neighborhood of each pixel for its label (foreground or background). If the 4-connective neighborhood of a pixel p contains the background label, then we mark p as a boundary pixel. Each boundary pixel is linked to the nearest boundary pixel to create a contour. A contour is either internal or external depending on whether its consists of inner or outer boundary pixels. To trace a contour from boundary pixels, the algorithm scans the image from top to bottom and left to right while for an unscanned boundary pixel. Starting at the first unscanned pixel, the algorithm

traverses its neighborhood in clockwise order, starting from position 2 in the pixel neighborhood graph in Figure 5a. While traversing the neighborhood, the algorithm locates the next unscanned boundary pixel n and previous boundary pixel p. The algorithm uses predefined path templates to trace the contour given a configuration of three unscanned boundary pixels p, c, n.

A path template (Figure 4a) is a predefined ordering of con-



(a) (b)

Figure 4: (a) Path-template for the given p, c, n ordering of pixels (b) Traced polygons (represented by circles) from path-template based algorithm

tour vertices that are linked together while traversing the neighborhood of an unscanned pixel p. Each contour vertex (blue dot) is located at the mid point of an edge of the corresponding unscanned pixel. These path-templates are designed in a way that ensures that the resulting contour is single pixel wide. There are 68 path templates in total. The use of path templates simplifies the boundary extraction process and ensures non-intersecting contours. Figure 4b shows a path template (represented by dots) for one of the configurations of p, c, and n. Here x denotes other unscanned pixels. After adding the contour vertices from the path-template, the algorithm searches for the next unscanned pixel until the algorithm returns to the first unscanned pixel. Internal contours are traced in the same manner by moving in anti-clockwise order of neighborhood traversal. Figure 4b shows a linked boundary (blue) by using path templates and creation of non-intersecting contours as compared to contour generated with Moore's Neighbor Tracing algorithm (red).

3.1.3 *COMPONENT EXTRACTION*

While user may have sufficiently extracted the information, often extracted stack may contain contours that are not part of interest. To identify and eliminate these contours, framework employs connected component analysis on the contour stack. To compute these connected components, we build an undirected graph $G = (V, E)$ on the set of con-

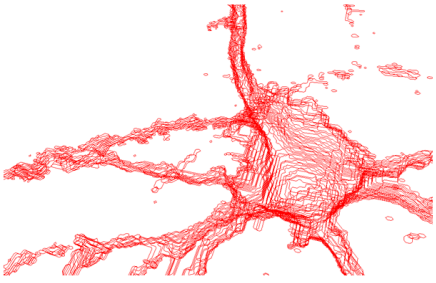


Figure 5: Visualization of Contour Stack in 3D

tours where edges of the graph have an associated weight equal to the geodesic distance between contours. We define this geodesic distance between two contours C_1 and C_2 as $dis_g(C_i, C_j) = min(dis(v_p, v_q)$ where $v_p \in C_i, v_q \in C_j)$, and $dis(v_p, v_q)$ is equal to the Euclidean distance between vertex v_p and v_q. A pair of contours C_i and C_j ($C_i \in$ section S_a and $C_j \in$ section S_b), have an edge between them *iff* $a = b$ or S_a, S_b are adjacent sections and $dis_g(C_1, C_2) < t_0$, where t_0 is the chosen threshold. Breadth-first search (BFS) is then performed on the graph G to determine the connected components. Usually, the selection of the largest component is sufficient to represent the 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 5) and if necessary, can vary the threshold in the viewer. The choice of threshold t_0 is simplified via analysis of components over all possible values of thresholds. Framework collects the number of components and the length of the largest component information as parameters for analysis and optimal threshold value is chosen from the range of values where slope of these two graphs approximates to zero. Figure 6 shows the component analysis plot for the input where the optimal range for t_0 is $20 - 24$.

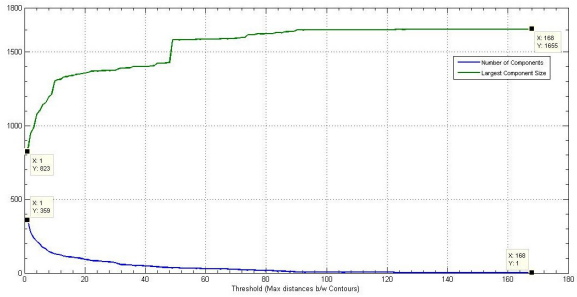


Figure 6: Plot of number of connected components and size of largest connected component of the component graph

3.2 RECONSTRUCTION

The reconstruction module in the framework is an adaptation of β-connection algorithm[7] with underlying tetrahedron disconnection method, modified to correctly handle the branching problem for reconstruction of neuronal structures. The algorithm relies on the construction of the Delaunay Triangulation (DT) which generates a volumetric representation connecting the contours of adjacent cross sections. Although, similar approaches (Das et al.[3]) have been proposed before, β-connection algorithm is particularly adequate to our reconstruction module as it is capable of generating multiple alternatives when establishing region correspondence and intrinsically handles tiling and branching problem using DT. In addition, the algorithm also deals with the topological singularities that may appear during the process of reconstruction so that the reconstructed object is a manifold (which is a highly desirable property for numerical simulations). More importantly, all the operations in the reconstruction process are combinatorial enabling a more robust and efficient implementation.

3.2.1 BETA CONNECTION

The basis of the β-connection algorithm was initially introduced by Boissonnat[1] who proved that the regions which are geometrically well positioned can be found through topological tests on the DT. Later, Nonato et al.[7] 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 allowed the construction of different correspondence for a given set of contours. As a part of the algorithm, it is necessary to dis-

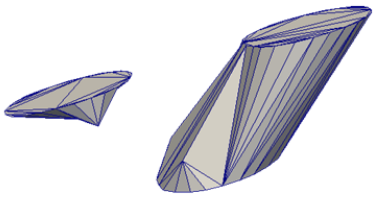


Figure 7: Vertex Displacement for inter-component disconnection

connect different β components completely by translation of vertices to intermediary level between sections. This translation is done to We compute the centroid of tetrahedra whose vertices are being translated and move the vertices towards the centroid to avoid intersections in the reconstructed surface (see Figure 7).

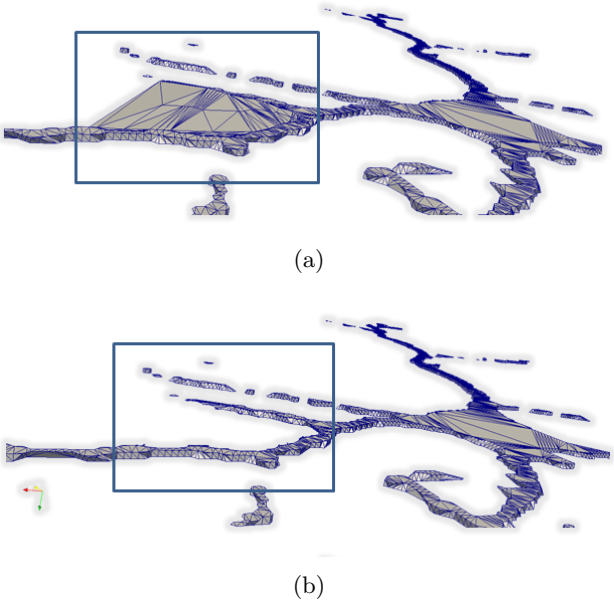
3.2.2 DISCONNECTION ALGORITHM

The β-connection algorithm in itself reconstructs the object that satisfies the 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 sections, generating branches at that intermediary level. However with the subdivision process, the β-connection algorithm generates branches (as shown in Figure 8a) in inner regions of adjacent sections with presence of tetrahedra belonging to same β-component. We handle the branching problem by completely avoiding the connections in inter-section region, by introducing a systematic disconnection method which also satisfies the manifold condition. We disconnect tetrahedra whose elimination generate singularities based on their orientation and edge classification:

Tetrahedra with one vertex in a section and other vertices in adjacent section. For these tetrahedra having two of its edges as external, we divide along the external and intra-section edges and then translate the vertices along these edges (as shown in Figure 9a where *cb* and *bd* are the external edges). For tetrahedra having three of its edges lying in external region, we follow the division criteria as in Figure 9b where cb, dc and bd are the external edges.

Tetrahedra with two edges in adjacent sections. For these tetrahedra having two of edges as external, we subdivide along all of its external edges and translate the vertices along these edges (as shown in Figure 9b where *ad* and *bc* are external edges). For tetrahedra having two of its edges on contour boundary and one external edge, we subdivide as shown in Figure 9a, where *dc* is the external edge.

In our algorithm, neighborhood around the vertex of edges



(a)

(b)

Figure 8: (a) Reconstruction of surface between adjacent sections (b) Branching in contours from our disconnection method

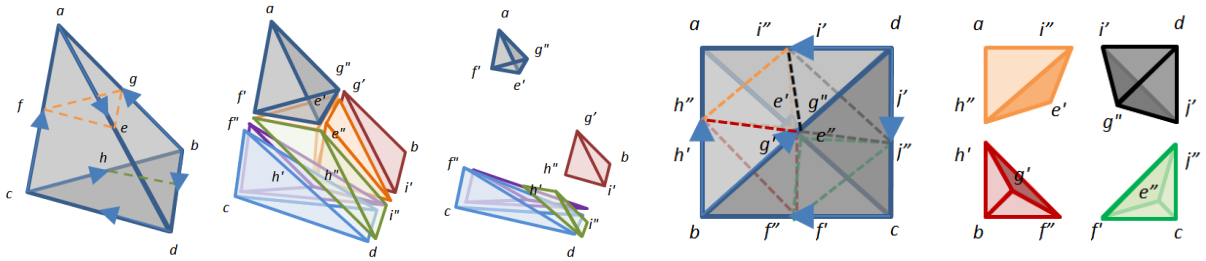
which are lying on the contour is always maintained, which inturn ensures that reconstructed object is a manifold. Figure 8b illustrates the results of disconnection algorithm applied to a pair of sections in the dataset.

3.3 CORRECTION

Correction module comes in action if the reconstruction module is unable to generate a connected component with a given contour set. This is generally due to segmentation process producing a set of spatially distant contours. Varying the β parameter in reconstruction module to establish a higher correspondence creates unwanted connections in these regions since the value remains same for an intersection region. Framework solves this problem by computing the minimum spanning tree (MST) of the components in contour stack. Extracted contours are then merged to create new contour stack. 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.

3.3.1 CONTOUR COMPONENT GRAPH

To compute this MST, we first build a graph G_1 among the contours in cross-sections with its edges formed by relation $dis_g(C_j, C_k) > 1$ for all $C_j, C_k \in$ sections S_i, S_{i+1}. Next, we compute the components by performing a BFS on graph G_1. These components (shown in Figure 10a) are then used as nodes for constructing the graph Gc. The edges of graph Gc are assigned a label $lab(Co_j, Co_k) = min(dis_g(C_m, C_n)$ for all $C_m \in Co_j \cap S_i, C_n \in Co_k \cap S_i; i = 1, ..., k)$, where Co_j and Co_k are the nodes of graph G_c. Next, we filter the edges of graph G_c by the relation $lab(Co_j, Co_k) < t_1$, where t_1 is the chosen threshold. The selection of threshold is again done through component analysis plot (see Figure 10b). Finally, we compute the minimum spanning tree on



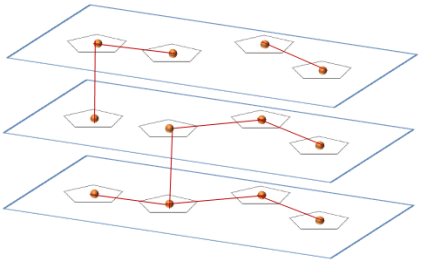
(a) (b)

Figure 9: Disconnection of (a) tetrahedron with two external edges (cb and bd) lying on a section and tetrahedron with two edges lying on contours in adjacent section(b) tetrahedron with two external edges (ad and bc)

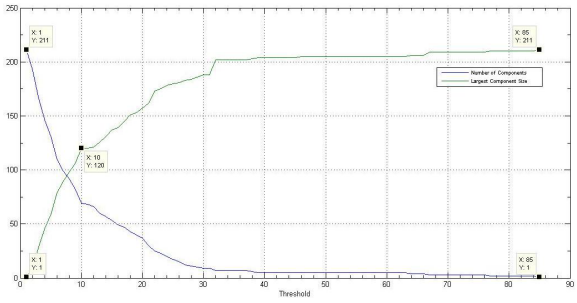
the filtered component graph G_c. The resultant output of the algorithm is either a single component or a set of connected components, determined by connectedness of the filtered component graph G_c. To establish the correspondence among the contours of obtained forest (or tree), we rasterize the segmented images along the shortest path connecting the two contours. These binary images are then re-fetched to the segmentation module to extract the boundary. The reconstruction process is then repeated in pipeline.

4. RESULTS AND DISCUSSION

We now discuss the reconstruction of chosen dataset along the different stages of the pipeline in the framework and how varying parameters in the different modules of affects the reconstructed structur. Based on histogram analysis defined in 3.1.1, an initial estimate of threshold, $g_0 = 22$ with an overcomplete segmentation is provided by the framework. We further tune the threshold using visual tools in selected regions to extract the small structures. Once we sufficiently extract the information in thresholding process, we use morphological operations on the images to patch holes and vacant neighbourhood pixels in the images. The results of morphological operations (two operations of openings followed by two operations of closing) with available neighborhood template are shown in Figures 11a and 11b. Next, we extract the boundaries in the images to form the required contour stack. Figure 13a shows the result of component filtration algorithm. Contours shown in red have been selected for extraction with the chosen threshold value. We selected the threshold value from component analysis plot shown in Figure 6 with optimal parameter range of $20 - 24$. The reconstructed mesh for the contour stack is shown in Figure 13b with chosen β value of 1. As evident in the reconstructed model, module was unable to generate a connected component for extracted contours of dendritic branches regions of neuron. Thus we employ correction module. Figure 13c shows the new extracted contour stack with merged contours shown in blue. We chose threshold value of $t_1 = 10$ from component analysis plot 10b of extracted contour stack. Figure 12 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. The overall total time taken by our framework for the reconstruction of given dataset was arround $624.07sec \approx 10$ minutes. The framework was deployed on HP workstation xw6600 machine (equipped with an Intel Xeon quad-core



(a)



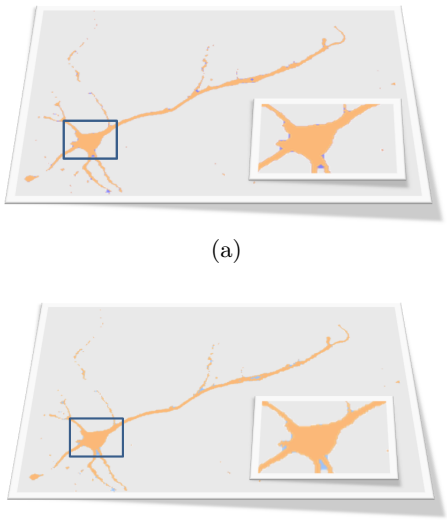
(b)

Figure 10: (a) Components of Contour Graph G_1 (b) Plot of number of connected components and size of largest connected component of the component graph.

processor E52053 1.86 GHz, 6 MB L2 cache, 1066 MHz FSB and 8 GB of RAM). On comparison with commercial softwares such as NeuroLucida, it takes an average time of 4-6 hours to manually trace the boundaries and generate the surface of the given dataset. Moreover, 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 reconstruction of structures of different scale and size.

5. ERROR ANALYSIS

To validate the accuracy of reconstruction from our framework, we constructed a virtual neuron model using 3D modelling and compared the results of reconstruction with the original mesh. The virtual model contains a soma, axon and flow of dendrites in 3D space (as shown in Figure 14a).



(a)

(b)

Figure 11: Application of morphological operations to segmented image with (a) 4-adjacent neighborhood template (b) 8-adjacent neighborhood template

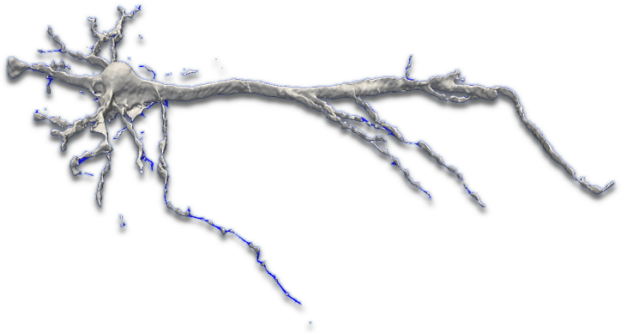
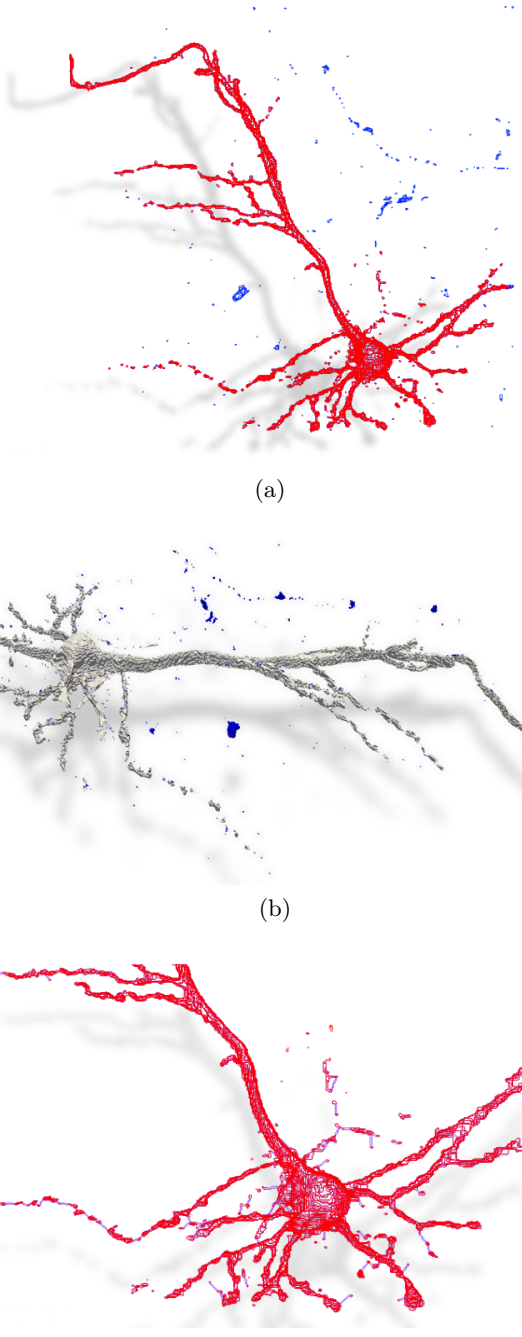


Figure 12: Final reconstructed mesh with newly created connections(blue)

Following the design, we computed the cross sections of the 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 crossover image. To achieve the partial volume effect and intensity decay, we applied the lens blur effect to the images. This process helped us achieve realistic challenges often present in the images due to uneven dye distribution and point spread of microscope optics. Finally, we fed these images to our framework and obtained the reconstructed mesh (Figure 14b) of the virtual neuron model.
For analysis, we used a proprietary tool called Metro[2] which numerically compares two triangular meshes by computing the difference of each sampled vertex of the surface of the source mesh to the target mesh. The output from the Metro tool consists of mean and RMS distance as the two measures to evaluate the approximation. We used both reconstructed mesh (backward) and original mesh (forward) as the sampled mesh and evaluated the results. We computed percent-



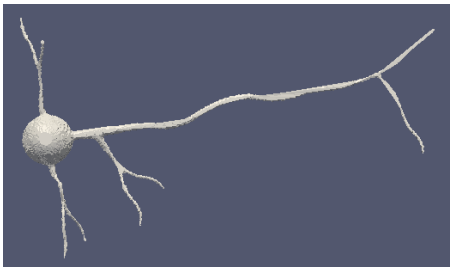
(a)

(b)

(c)

Figure 13: (a) Selected contours (marked in red) from stack with the threshold, $t_0 = 22$ (b) Reconstructed mesh with noisy components (marked in blue) (c) New contour stack with merging of contours (marked in blue) in corrections module for threshold $t_1 = 10$

(a)



(b)

Figure 14: (a) Virtual Neuron Model (b) Reconstructed Neuron model with framework

Table 1: 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

age error on the non-sampled mesh as value of mean distance with respect to diagonal of the bounding volume of sampled mesh. The results reported in Table 1, indicate that the errors are low.

6. CONCLUSIONS

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 post-processing. We feel that results from our framework were more than adequate for the chosen dataset and can be applied for functional visualization of other neuron structures. From a neuro-biologists 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 may be derived from our reconstructed mesh.

7. ACKNOWLEDGEMENTS

This work was partially supported by the DST Center for Mathematical Biology, IISc, under Grants SR/S4/MS:419/07 and SR/S4/MS:799/12.

8. REFERENCES

[1] J. D. Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, pages 1–29, 1988.

[2] P. Cignoni, C. Rocchini, and R. Scopigno. Metro:measuring error on simplied surfaces. *Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Pisa, Italy*, 1998.

[3] K. Das, A. Majumder, M. Siegenthaler, H. Keirstead, and M. Gopi. Automated analysis of remyelination therapy for spinal cord injury. *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 314–321, 2010.

[4] A. Dima, M. Scholz, and K. Obermayer. Automatic segmentation and skeletonization of neurons from confocal microscopy images based on the 3-d wavelet transform. *IEEE Transaction on Image Processing*, pages 790–801, 2002.

[5] F. Fleuret and P. Fua. Dendrite tracking in microscopic images using minimum spanning trees and localized e-m. Technical report, Computer Vision Lab, EPFL, 2006.

[6] W. He, T. Hamilton, A. Cohen, T. Holmes, C. Pace, D. Szarowski, J. Turner, and B. Roysam. Automated three-dimensional tracing of neurons in confocal and brightfield images. *Microscopy and Microanalysis*, pages 296–310, 2003.

[7] L. Nonato, A. Vargas, R. Minghim, and M. Oliveira. Beta-connection: Generating a family of models from planar cross sections. *ACM Transactions on Graphics*, pages 1239–1258, 2005.

[8] A. Sadeghipiur. Algorithms of rutomatic reconstruction of neurons form the confocal images. Master's thesis, 2008.

[9] A. Santamaria and I. Kakadiari. Automatic morphological reconstruction of neurons from optical imaging. *Micscoscopy Image Analysis and Applications in Biology Workshop*, 2007.

[10] C. Uehara, C. Colbert, P. Saggau, and I.Kakadiaris. Towards automatic reconstruction of dendrite morphology from live neurons. *IEEE Engineering in Medicine and Biology Society*, pages 1798–1801, 2004.

[11] S. Urban, S. 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*, pages 190–201, 2006.

[12] Y. Zhang, X. Zhou, J. Lu, J. Lichtman, D. Adjeroh, ST, and Wong. 3d axon structure extraction and analysis in confocal fluorescence microscopy images. *Neural computation*, pages 1899–1927, 2008.

[13] T. Zhao, J. Xie, N. C. F. Amat, P. Ahammad, H. Peng, F. Long, and E. Myers. Automated reconstruction of neuronal morphology based on local geometrical and global structural models. *Neuroinform*, pages 247–261, 2011.