

---

# Quality Meshing of a Forest of Branching Structures

Chandrajit Bajaj<sup>1</sup> and Andrew Gillette<sup>2</sup>

<sup>1</sup> Department of Computer Sciences & Institute for Computational Engineering and Sciences, University of Texas at Austin, USA

`bajaj@cs.utexas.edu`

<sup>2</sup> Department of Mathematics & Institute for Computational Engineering and Sciences, University of Texas at Austin, USA

`agillette@math.utexas.edu`

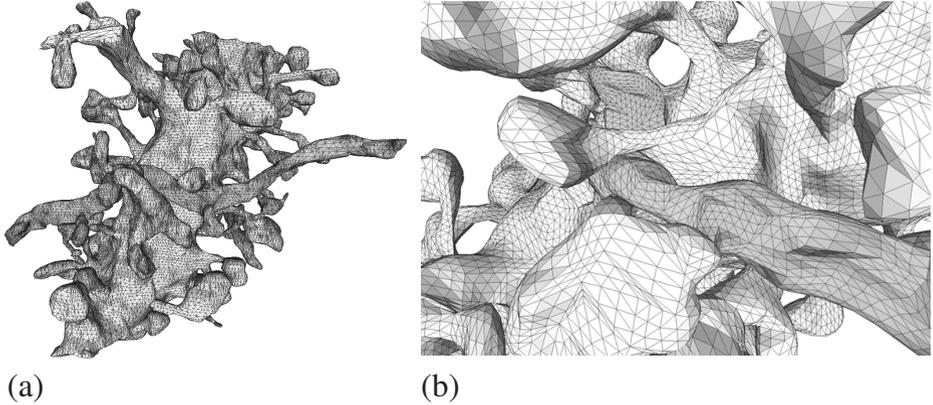
**Abstract.** Neurons are cellular compartments possessing branching morphologies, with information processing functionality, and the ability to communicate with each other via synaptic junctions (e.g. neurons come within less than a nano-meter of each other in a specialized way). A collection of neurons in each part of the brain form a dense forest of such branching structures, with myriad inter-twined branches, inter-neuron synaptic connections, and a packing density that leaves only 5% - 10% volume fraction of exterior-cellular space. Small-scale variations in branching morphology of neurons and inter-neuron spacing can exert dramatically different electrical effects that are overlooked by models that treat dendrites as cylindrical compartments in one dimension with lumped parameters. In this paper, we address the problems of generating topologically accurate and spatially realistic boundary element meshes of a forest of neuronal membranes for analyzing their collective electrodynamic properties through simulation. We provide a robust multi-surface reconstruction and quality meshing solution for the forest of densely packed multiple branched structures starting from a stack of segmented 2D serial sections from electron microscopy imaging. The entire 3D domain is about 8 cubic microns, with inter-neuron spacing down to sub-nanometers, adding additional complexity to the robust reconstruction and meshing problem.

## 1 Introduction

Biological modeling problems have long been an inspiration for geometry processing and meshing research. A relatively recent technique known as serial section transmission electron microscopy (ssTEM) presents a new set of challenges to the computational geometry community. The technique employs a CCD camera to capture high resolution images of parallel slices of a collection of cells, usually neurons. The resolution on a single image slice is 5-10 nanometers, small enough to identify features on the boundaries of the cells, as well as organelles within the cells. [18, 36] Trained neuro-anatomy aware users can approximate the cell boundaries by visual inspection and label their polygonal contours consistently through the stack so that slice images of a single cell (dendrite, glial, axon, etc.) can be traced through the entire volume, as shown in Figure 1. This suggests the following computational problem: develop a method to create a watertight mesh



appropriate solution for neuronal models. Image data reveals that neurons have widely varying cross-sectional areas and boundary shapes and are packed very densely (an example is shown in Figure 3), with as little as 5-10% of the image representing extra-cellular space [26, 35]. Even if the contours approximating the boundaries on each slice are non-overlapping, an under-constrained surface reconstruction might create intersections in three dimensions.

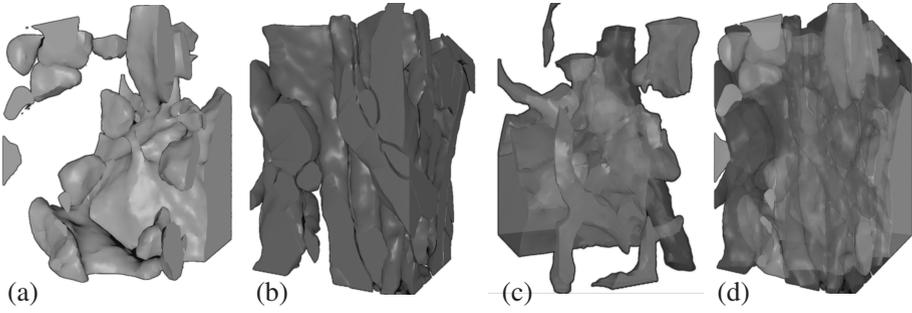


**Fig. 2.** The approach provided in this paper allows for quality and accurate meshing of intertwined and branching structures, such as the neuronal meshes shown here. **(a)** A meshed dendrite is shown in grey with an adjacent axon passing nearby, shown in green. **(b)** A zoomed in portion of the meshes shows the triangles of the mesh are of good quality, making the mesh fit for electro-physiological simulations.

In this paper, we present a solution to the general problem of simultaneously reconstructing a densely packed forest of intertwined branching surfaces from point samples lying on a finite set of parallel slices through the volume. Our approach uses the medial axis on each slice to help construct a 3D fencing between contours, thereby preventing the intersection of distinct surfaces. In Section 2, we formalize the problem and provide additional explanation of background concepts. In Section 3, we describe in detail an algorithm to resolve the problem and prove its correctness. In Section 4, we show some initial mesh results. Finally, in Section 5 we conclude and describe the various uses these types of meshes will have in our future work.

## 2 Problem Statement and Background

In Section 2.1, we formally state the multi-component reconstruction problem we aim to solve and fix notation for use in the description of our solution. In Section 2.2 we describe the three main problems of the single component problem - correspondence, tiling, and branching - which are relevant to the multi-component problem.



**Fig. 3.** A visualization of a portion of neuronal data indicating the difficulty in the forest of branching structures problem. The reconstruction visualization was prepared by Justin Kinney, Thomas Bartol, and Terrence Sejnowski of the Salk Institute using our tiling program [4]. **(a,b)** Surfaces representing dendrites and axons, shown in yellow and green, respectively. **(c)** Surfaces representing glial cells are shown in purple in this view, along with some of the axonal and dendritic surfaces for reference. **(d)** All three types of surfaces share this densely packed volume. Since each surface is reconstructed independently of the others, we have to resolve any possible intersections that result when the individual models are assembled in the same volume. Here, the green surface has been made partially transparent to show how the surfaces wind and branch around each other. The glial cells are present but not visible in this view.

In Section 2.3 we explain the basic theory behind the medial axis which is used prominently in our solution.

### 2.1 Formal Problem Statement

With the goal in mind of designing an algorithm for neuronal modeling, we now describe the more general surface reconstruction problem we aim to solve.

Suppose we are given a positive integer  $M$  and the input  $(\{Z_i\}, \{g_i\}, \{c_i^j\}, G)$  as follows:

- **Input 1:** A set of  $M$  horizontal planes  $Z_1, \dots, Z_M$  where  $Z_i$  is the plane  $z = z_i \in \mathbb{R}$  with  $z_1 < \dots < z_M$ .
- **Input 2:** A set of curves (contours) in each  $Z_i$  plane described implicitly by  $g_i(x, y) = 0$ .
- **Input 3:** A list  $\{c_i^j\}$  of the contours of the set  $g_i(x, y) = 0$  where  $c_i^j$  denotes the  $j^{\text{th}}$  contour on the plane  $Z_i$ .
- **Input 4:** A directed graph  $G$  with vertices  $\bigcup_{i,j} c_i^j$  and edges only pointing to an element of list index incremented by one. That is, every edge of  $G$  can be written as  $(c_i^{j_1}, c_{i+1}^{j_2})$  for some indices  $i, j_1$ , and  $j_2$ .

Our goal is to construct a smooth function  $g : \mathbb{R}^3 \rightarrow \mathbb{R}$  such that the following properties hold:

- **Property 1:** The surface  $g(x, y, z) = 0$  is a compact 2-manifold.
- **Property 2:** The function  $g$  restricts to  $g_i$  on  $Z_i$ . That is, for all  $i$ ,  $g(x, y, z_i) = g_i(x, y)$  for all  $(x, y) \in \mathbb{R}^2$ .

- **Property 3:** The surface  $g(x, y, z) = 0$  has local connectivity corresponding to the graph  $G$ . That is, if  $(c_i^{j_1}, c_{i+1}^{j_2})$  is an edge in  $G$ , then  $c_i^{j_1}$  and  $c_{i+1}^{j_2}$  are homologous (meaning one smoothly deforms into the other) on the surface

$$\{(x, y, z) : g(x, y, z) = 0, \quad z \in [z_i, z_{i+1}]\}$$

We note that the set  $\{g^{-1}(0)\}$  is the desired surface passing through the contours with the correct connectivity. A general analytical solution to this problem is difficult and unnecessary for implementation, hence we make the following simplifying assumptions based on the application context.

**Definition 1.** (adapted from [1]) A homeomorphism  $h$  of  $\mathbb{R}^3$  is **isotopic to the identity** if there is a homotopy  $H : \mathbb{R}^3 \times [0, 1] \rightarrow \mathbb{R}^3$  such that for each  $t \in [0, 1]$ ,  $h_t := H(\cdot, t) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  is a homeomorphism,  $h_0$  is the identity and  $h_1 = h$ . A mesh  $M \subset \mathbb{R}^3$  of a surface  $S \subset \mathbb{R}^3$  is called an **isotopic mesh** if and only if there exists a homeomorphism  $h$  of  $\mathbb{R}^3$  carrying  $M$  to  $S$  with  $h$  isotopic to the identity.

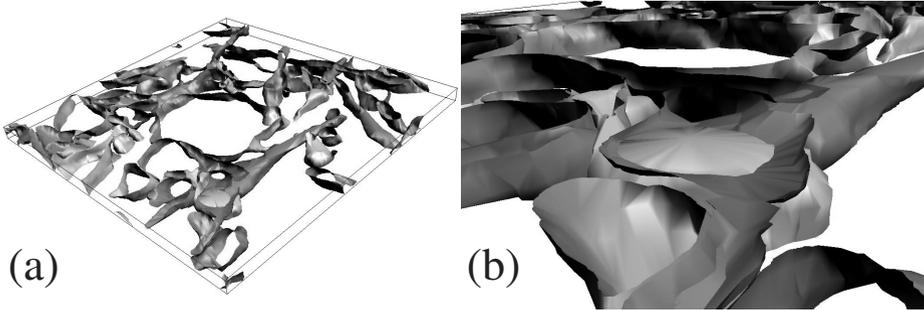
- **Assumption 1:** The contours are described by simple polygons whose vertices lie on the contour. If the geometry of a particular polygon needs to be refined, the appropriate  $g_i$  can be referenced to increase the sampling density.
- **Assumption 2:** If  $M$  is a compact piecewise linear 2-manifold such that it restricts to the contours on each slice and has local connectivity corresponding to the graph  $G$ , then  $M$  is an isotopic mesh of the surface  $g(x, y, z) = 0$ .
- **Assumption 3:** The distance between consecutive  $z_i$  values is small enough that the slice sampling of each component in the surface  $g(x, y, z) = 0$  meets the requirements of the single-component meshing algorithm.

Each assumption is based on practical considerations from neuronal data. To satisfy Assumption 1, we fit each polygonal contour data as tagged by biologists with a regular algebraic spline curve called an  $A$ -spline. By using the error-bounded spline method described in [5], we can construct smooth approximations of the contours while preventing overlaps between adjacent contours. The splines are defined locally based on a scaffolding mesh, allowing us to efficiently increase the sampling of a contour in a particular region if needed. Assumption 2 is made to distinguish the surface meshing problem from the smooth surface construction problem. In this paper, we are only interested in how to create an isotopic mesh of the smooth surface approximation and leave the surface fitting to future work. Assumption 3 is stated so that we can tackle the multi-component problem without inheriting existing difficulties from the single component problem. As is discussed in [4], it is desirable to produce a mesh such that any vertical line between two adjacent slices passes through the mesh at most once. We clarify this criterion in the next subsection.

## 2.2 Correspondence, Tiling, and Branching

The single component surface reconstruction problem faces three major sub-problems: the correspondence problem, the tiling problem and the branching

problem. Our solution to the multi-component problem requires an effective single component solver, hence we will review the three problem types here. The single component solver we use is denoted SINGLESURFRECON and resolves the problems in full generality under the assumptions previously stated. The method is summarized below, but is given in full detail in [4]. To describe the problems, we consider two planes  $Z_1$  and  $Z_2$  with  $z_1 < z_2$  and let  $\{c_1^j\}$ ,  $\{c_2^k\}$  denote the lists of contours in the respective planes. Figure 4 illustrates some of the difficulties in meshing such intricate data.



**Fig. 4.** (a) The region between two consecutive slices is shown, with portions of many dendrites (grey) and axons (green) present. The mesh has been filled in and smoothed slightly for the purpose of visualization. The contours on the top plane must be tiled to the contours on the bottom, subject to the correspondence constraints of the data. (b) Zooming in on a portion of the previous image reveals that branching has occurred in an axon and dendrite in close geometric proximity, resulting in this unwanted surface intersection. We will present a method for detecting and removing such intersections in Section 3.

The *correspondence problem* is to decide how the contours of  $\{c_1^j\}$  will match up to the contours of  $\{c_2^k\}$ . This requires either *a priori* knowledge of the contours' connectivity or a geometric criterion for declaring correspondence. In the context of our problem domain, the correspondence problem for SINGLESURFRECON is resolved by consulting the graph  $G$  as it prescribes the connectivity between contours.

The *tiling problem* is to decide, given contours  $c_1 \in \{c_1^j\}$  and  $c_2 \in \{c_2^k\}$ , how  $c_1$  and  $c_2$  will be joined in the interslice region. In the context of meshing,  $c_1$  and  $c_2$  are given as polygons and the problem is to decide how to add edges between them so that a suitable mesh of the ribbon surface between the contours is produced. A line segment connecting  $c_1$  to  $c_2$  is called a *slice chord* and a triangle formed by two slice chords an edge of a contour is called a *tiling triangle*. Even in very simple cases, there are many choices available for how to choose slice chords and tiling triangles. One resolution to this problem is to define a quality measure on possible tilings and seek a tiling with the optimal quality [22, 15]. A summary of different quality metrics used in early methods is given in [32]. Alternative approaches, such as the one we use, project contours from  $Z_2$  to  $Z_1$

and use planar geometry properties to lift a watertight surface mesh from the projection [4, 34, 28, 6].

The choice of tiling method is highly relevant to the types of guarantees one can provide on the output meshes. We have selected the method of [4] because it only outputs surfaces that meet the following three criteria.

- **Criterion 1:** The reconstructed surface forms a piecewise closed surface of polyhedra.
- **Criterion 2:** Any vertical (meaning parallel to the  $z$  axis) line segment between two adjacent slices intersecting the reconstructed surface does so at exactly one point or along exactly one line segment.
- **Criterion 3:** Resampling of the reconstructed surface on any slice reproduces the original contours.

Criterion 1 ensures that self-intersecting surfaces and other incorrect meshes are not formed. Criterion 3 ensures that all the contours are interpolated and no new ones are created. Criterion 2 is especially important because it ensures that the reconstructed surface is functional from its nearest planes. That is, barring the case of vertical surface patches, the projection of any triangle in the mesh to either of its two nearest  $Z_i$  planes is a one-to-one mapping. This prevents unlikely topologies from being formed and aids in the proof of the correctness of the multi-component method as discussed in Section 3.7. We note that Criterion 2 may not be satisfiable if the distance between consecutive  $z_i$  values is too large relative to the surface feature size, however, by Assumption 3, this is not the case. In practice, exceptions to Assumption 3 are rare and therefore Criterion 2 is not particularly restrictive on the input type.

We will now summarize the approach that the SINGLESURFRECON algorithm uses to resolve the tiling problem. We will appeal to intuitive notions of the *left side* and *right side* of a vertex on a contour relative to its clockwise (CW) or counterclockwise (CCW) orientation; these definitions are made explicit in [4]. Let  $\{c_2^{k'}\}$  denote the projection of  $\{c_2^k\}$  vertically onto  $Z_1$ . We compute the points of intersection between  $\{c_1^j\}$  and  $\{c_2^{k'}\}$ ; without loss of generality we assume that these points of intersection are vertices on contours in both sets. Such vertices that are common to both sets are called an *overlapping* vertices; all other vertices are called *non-overlapping*. If a tiling of  $\{c_1^j\}$  and  $\{c_2^k\}$  satisfies the three Criteria, the following two theorems must hold.

**Theorem 1.** ([4], Theorem 2) *Let  $v$  be a vertex on a contour in  $c_1 \in \{c_1^j\}$  and  $T$  a slice chord from  $v$ . (i) If  $v$  is a non-overlapping vertex, the projection of  $T$  onto  $Z_1$  is contained entirely on one side (right or left) of  $c_1$ . The side is determined by the orientation of the nearest enclosing contour from the set  $\{c_2^k\}$  when  $v$  is projected to  $Z_2$ . (ii) If  $v$  is an overlapping vertex,  $v$  also belongs to some  $c'_2 \in \{c_2^{k'}\}$  by hypothesis. In this case, the projection of  $T$  onto  $Z_1$  does not intersect the region to the left of both  $c_1$  and  $c'_2$  at  $v$ , nor the region to the right of both  $c_1$  and  $c'_2$  at  $v$ .*

**Theorem 2.** ([4], Theorem 4) *Let  $T$  be a slice chord and  $c_1 \in \{c_1^j\}$ . Then the projection of  $T$  onto  $Z_1$  cannot intersect both the inside and outside of  $c_1$*

The tiling algorithm proceeds as follows. For each vertex  $v \in \{c_1^j\}$ , make a list of all the slice chords that could be formed to a vertex of  $\{c_2^k\}$  (based on the resolution of the correspondence problem). Select the shortest length chord from this list which satisfies the results of Theorems 1 and 2. If no chord from the list satisfies both Theorems, tag the vertex as “untiled.” The boundaries of untiled regions are later collected and meshed separately while resolving the branching problem.

The *branching problem* arises when the result of the correspondence problem yields a matching of more than one contour in  $\{c_1^j\}$  to any number of contours in  $\{c_2^k\}$  (or vice versa). Solutions to the problem include adding edges to the contours on one of the planes or adding vertices in between the planes to create an appropriate mesh. Since the former approach violates Criterion 3, we employ the latter. The branching problem occurs only in regions previously tagged as “untiled” by SINGLESURFRECON, so we collect the boundaries of these regions and project them to a plane half way between the two planes in consideration. We triangulate these untiled regions based on their Edge Voronoi Diagram using the algorithm of Lee [29]. Any new vertices added are given a  $z$  value of  $.5(z_1 + z_2)$ . A summary of alternative approaches to the branching problem and further details are given in [4].

The output of SINGLESURFRECON is a mesh with the desired Properties and satisfying the given Criteria. In practice, SINGLESURFRECON can be implemented in a numerically stable manner and will provide an output for many types of real data.

### 2.3 Medial Axis

The main tool employed by our algorithm to ensure accurate surface reconstruction is the medial axis of the region exterior to the contours. We give a thorough explanation of the medial axis in Appendix A. The medial axis of a set  $O$  is often approximated based on a point sampling  $P$  of the boundary  $\partial O$  and the Voronoi and Delaunay diagrams  $\text{Vor } P$  and  $\text{Del } P$ . A definition of these diagrams and



**Fig. 5.** A collection of contours representing cell boundaries and their interiors are shown in red along with the approximate medial axis of intracellular space  $\mathfrak{M}_P$  shown in blue

how to compute them can be found in any standard computational geometry textbook, e.g. [19]. The medial axis is approximated by the graph of the vertices of Vor  $P$  along with those edges of Vor  $P$  not intersecting the contours. We denote this graph  $\mathfrak{M}_P$ ; an example is shown in Figure 5. For 2D cellular contour data,  $\mathfrak{M}_P$  will have one component for each cell interior plus a component for the extracellular region, assuming  $P$  is a sufficiently dense sampling of  $\partial O$ . We discuss sampling density and our robust medial axis computation method further in Section 3.2.

### 3 Algorithm

In this section we describe an algorithm to solve the multi-component problem under the Assumptions given in Section 2.1. In the next six subsections, we will explain the input to the algorithm and its five major steps. We conclude this section with a proof of the correctness of the approach.

#### 3.1 Algorithm Input

As stated in Section 2.1, our input is the following objects:  $(\{Z_i\}, \{g_i\}, \{c_i^j\}, G)$ . By our stated Assumptions, we may suppose that the contours  $\{c_i^j\}$  are simple polygons with sufficiently dense sampling of the level sets  $g_i(x, y) = 0$ . Define  $P_i = \{v : \exists j \text{ such that } v \text{ is a vertex on } c_i^j\}$ . We pass these polygonal contours and the associated point sets  $P_i$  to our algorithm instead of the functions  $g_i$ .

For ease of description and to maintain consistency with our neuronal data, we explain the algorithm for the case where  $G$  has just two components, with one colored yellow and one colored green. Each contour can thus be uniquely assigned as belonging to the set of yellow contours  $YC$  or green contours  $GC$ , i.e.

$$\{c_i^j\} = YC \amalg GC$$

The notation we will use for calling the algorithm with this input is

$$\text{MULTISURFRECON} \left( \{Z_i\}, \{c_i^j\}, \{P_i\}, G \right)$$

#### 3.2 Step 1: Construct and Partition 2D Medial Axes

##### Overview

Compute the approximated medial axis  $\mathfrak{M}^{P_i}$  for each  $1 \leq i \leq M$  and discard the portions interior to the contours. Each remaining edge  $e$  separates two unique contours, call them  $c_{e1}$  and  $c_{e2}$ . Partition these edges into three sets:

$$\begin{aligned} E_{YY} &= \{e : c_{e1}, c_{e2} \in YC\}, & E_{GG} &= \{e : c_{e1}, c_{e2} \in GC\}, \\ E_{GY} &= \{e : c_{e1} \in YC, c_{e2} \in GC \text{ or } c_{e1} \in GC, c_{e2} \in YC\} \end{aligned}$$

*Details*

It is crucial that the medial axes on each slice be computed robustly and with topological accuracy for the remainder of the algorithm to work. Thus, we turn to a criterion established by Edelsbrunner and Shah known as the *closed ball property* [20] which we state here for polygonal contours in a plane. Let  $P$  be the vertices of all the contours and  $\Sigma$  the union of all the contours on a particular plane. A Voronoi object  $V$  from  $\text{Vor } P$  of dimension  $k$  satisfies the closed ball property if and only if  $V \cap \Sigma = \emptyset$  or  $V \cap \Sigma$  is homeomorphic to a closed ball of dimension  $k - 1$ . If all  $V \in \text{Vor } P$  satisfy the closed ball property and  $\Sigma$  intersects each  $V$  transversally, then  $\Sigma$  is a subgraph of  $\text{Del } P$  [20]. This implies that  $\mathfrak{M}_P$  is a subgraph of  $\text{Vor } P$ . Cheng, Dey, Ramos, and Ray used this criterion to create Delaunay-conforming meshes without approximating local feature size [13]. We provided an efficient algorithm exploiting their approach for use with image data [25] and we adopt that algorithm for computation of  $\mathfrak{M}_P$ . Figure 5 shows a zoomed in picture of a slice of extracted contours with  $\mathfrak{M}_P$  displayed.

By construction,  $\mathfrak{M}_P$  is a multi-component piecewise linear graph. We discard all components of  $\mathfrak{M}_P$  which are interior to the contours, leaving a single medial axis approximation of extracellular space. From this point on, we will take  $\mathfrak{M}_P$  to mean this single connected graph. Each edge of  $\mathfrak{M}_P$  is a Voronoi edge and thus its dual Delaunay edge connects two points of  $P$ . We assign each edge to  $E_{GG}$ ,  $E_{YY}$ , or  $E_{GY}$  based on whether its associated points of  $P$  are both on green contours, yellow contours, or one of each, respectively.

**3.3 Step 2: Construct Medial Surface***Overview*

Dilate the edges in  $E_{GY}$  so that they form contours and color these contours red. Run `SINGLESURFRECON` on the red contours, producing a partial, approximate medial surface  $\mathfrak{M}S$ .

*Details*

On each plane, we consider only those edges of  $\mathfrak{M}_P$  which belong to  $E_{GY}$ . We dilate  $E_{GY}$  by setting a distance parameter  $\epsilon$  and defining the contours to be the locus of points at distance  $\epsilon$  from  $E_{GY}$ . For small enough values of  $\epsilon$ , this produces a collection of contours to which we assign the color red. We denote the dilated contours  $RC$ . Once we have done this for all the slices, we call `SINGLESURFRECON` on the whole set  $RC$  to produce a partial medial surface approximation denoted  $\mathfrak{M}S$ . In this computation, we also compute the intersection of  $\mathfrak{M}S$  with each intermediate plane  $z = .5(z_i + z_{i+1})$ .

**3.4 Step 3: Create Single Component Surfaces**

Run `SINGLESURFRECON` on  $YC$  and  $GC$  separately, producing meshes of the yellow and green contours. In this run, we also keep a list of those  $i$  values for

which vertices were added on the intermediate plane  $z = .5(z_i + z_{i+1})$ . This occurs whenever there are dissimilar contours or branching in a portion of the tiling.

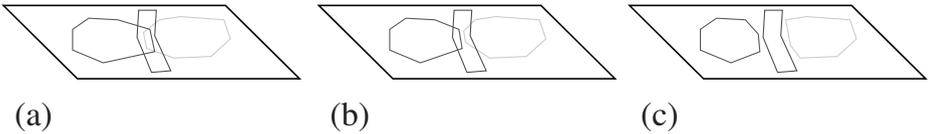
### 3.5 Step 4: Remove Overlaps on Intermediate Planes

#### Overview

For those  $i$  values on the list created in Step 3, compute the yellow, green, and red contours that occur on plane  $z = .5(z_i + z_{i+1})$ , which we denote  $Z_{i,i+1}$ . Do a plane sweep to detect overlaps between contours and remove them so that the mesh now avoids overlaps on each intermediate plane. Output the modified meshes of  $YC$  and  $GC$ .

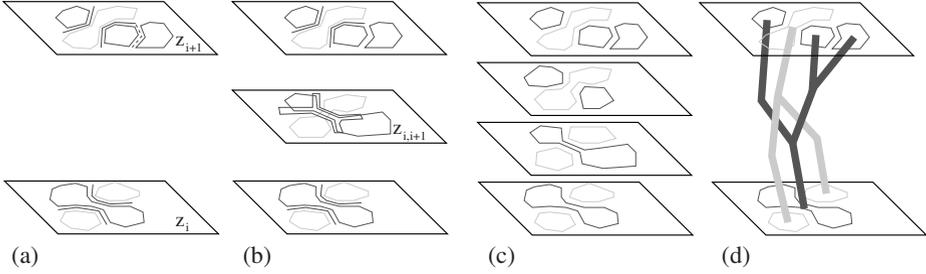
#### Details

From Steps 2 and 3, we have calculated red, yellow, and green contours on each intermediate plane  $Z_{i,i+1}$ . Geometrical problems may arise on those  $Z_{i,i+1}$  planes which have yellow and/or green vertices. For those planes, we do a plane sweep to detect overlap regions. We distinguish between two types of overlaps: simple transversal overlaps and exotic overlaps. A simple transversal overlap is one in which the border of the region is exactly two colors and each of these colors forms a connected component of the border. We resolve simple transversal overlaps by first computing the medial axis associated to the interior of the overlap. This is then used as a dividing line to guide vertex movement; the overlap is resolved by moving each vertex on the boundary across this axis. Note that this method does not add or remove vertices, does not change the mesh connectivity, and does not change the  $z$  value of any vertex.



**Fig. 6.** (a) A few overlaps have occurred on an intermediate plane  $Z_{i,i+1}$  between green contours, yellow contours, and red medial surface contours. We attempt to resolve them by searching for simple transversal overlaps such as the green/yellow overlap in this case. (b) After removing the green/yellow overlap, we are left with only red/yellow and red/green overlaps. (c) Removing the final overlaps, all the contours are separated.

An exotic overlap in a region  $Q$  can often be resolved by finding simple transversal overlaps within  $Q$  and resolving them in the manner described above. Such a case is shown in Figure 6. If this is not possible, we resolve the exotic overlap by changing the  $z$  values of vertices in the overlap region to either  $.25(z_i + z_{i+1})$  or  $.75(z_i + z_{i+1})$ . An example of such a case and how it is resolved is shown in Figure 7.



**Fig. 7.** A toy example of how our algorithm works in the case of exotic overlaps (see Section 3.5). **(a)** Green and yellow contours are detected in adjacent planes  $Z_i, Z_{i+1}$  and the medial axis  $\mathfrak{M}_P$  in each plane is computed, shown in red. Those portions of the medial axis which lie between same color contours are discarded, such as the dashed red portion in the upper plane. **(b)** An approximate medial surface  $\mathfrak{M}_S$  is computed by dilating the  $\mathfrak{M}_P$  into contours and calling SINGLESURFRECON. The intersection of  $\mathfrak{M}_S$  and the separately computed green and yellow surfaces with the intermediate plane  $Z_{i,i+1}$  is shown. The overlap of the contours cannot be resolved by removing simple transversal overlaps. **(c)** To remove the overlap, we move the green vertices on  $Z_{i,i+1}$  to a lower  $z$  value and the yellow vertices to a higher  $z$  value, without changing connectivity of the mesh. We show the intersections of the yellow and green surfaces on the two intermediate planes where the vertices have been moved. **(d)** The skeletons of the yellow and green meshes are shown to illustrate that the overall topology is now correct and no geometrical overlaps occur.

### 3.6 Step 5: Quality Improvement

We decimate and improve the quality of our output meshes in the following manner. For decimation, we first do edge contractions based on the Delaunay diagram, using the QSlim software by Michael Garland [23]. We then do normal-based triangle decimation based on the method presented in [3]. Finally, we improve the shape of triangles in the mesh using a geometric flow technique [38] which is a library in Level Set Boundary Interior and Exterior Mesher (LBIE) [16], part of the Volume Rover (VolRover) [17] software developed by our lab.

### 3.7 Correctness of the Algorithm

The goal of MULTISURFRECON is to output an isotopic mesh  $M$  of the multi-component surface  $S$  representing neuronal membranes such that the Hausdorff distance between  $M$  and  $S$  is small. Since SINGLESURFRECON produces such meshes for individual components, the main question is whether the unioning and separating processes employed by MULTISURFRECON truly remove all intersections between the component surfaces in three dimensions. We will show that removing mesh intersections on certain horizontal planes suffices to preclude any 3D intersections in the complete mesh. To state this more precisely, we introduce the following definitions and lemmas.

**Definition 2**

- An **original contour** is any contour  $c_i^j$  from Input 3 (described in Section 2.1). By Assumption 1, each  $c_i^j$  is a simple polygon.
- A **branching point** is a vertex added by MULTISURFRECON whose  $z$ -value is not one of the  $z_i$  values from Input 1.
- An **intermediate plane** is a horizontal plane which, after running MULTISURFRECON, contains at least one branching point.
- The **branching set** of a mesh component  $C$  on an intermediate plane  $P$  is the collection of branching points on  $P$  which belong to  $C$ , along with any edges of  $C$  between these points.
- An **intermediate contour** is a contour formed by intersecting  $M$  with an intermediate plane  $P$ . Note that an intermediate contour may contain some of the branching set of the component in which the contour lies.
- Let  $c_1, c_2$  be contours in the same plane and  $I(c_1), I(c_2)$  their respective interiors. Then  $c_1$  and  $c_2$  are said to **overlap** if and only if  $(I(c_1) \cup c_1) \cap c_2 \neq \emptyset$  and  $(I(c_2) \cup c_2) \cap c_1 \neq \emptyset$ . Note that if one contour is completely contained in the other, they do not overlap.

**Lemma 1.** *Suppose that for each  $i$ , there are no pairwise overlaps between the original contours on plane  $Z_i$ . Then the output of SINGLESURFRECON run on any subset of the original contours is not self-intersecting.*

Lemma 1 is a consequence of the three Criteria laid out in Section 2.1 and the tiling method; if the original contours do not overlap, SINGLESURFRECON cannot create self-intersections.

Next, observe that for the output of MULTISURFRECON with components  $YC$  and  $GC$ , there are three types of overlap that may occur on an intermediate plane  $P$ :

- An intermediate contour of  $YC$  intersects an intermediate contour of  $GC$ .
- An intermediate contour of  $YC$  intersects the branching set of  $GC$  (or vice versa).
- The branching sets of  $YC$  and  $GC$  intersect.

This characterization of types of overlaps extends naturally to outputs with more than two components. The following lemma shows that detecting and removing these types of overlaps is equivalent to removing self-intersections of the entire mesh. The lemma makes use of Assumption 2 from Section 2.1, without which, problems could arise from contours on consecutive planes being nested inside each other in contradictory ways. Since this should not happen with real data, we feel we are justified with the Assumption as stated.

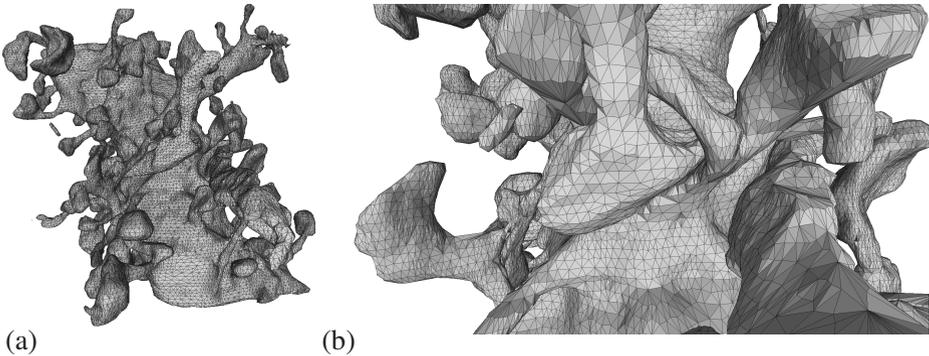
**Lemma 2.** *If there are no overlaps of any of the three types on any intermediate plane, then the output of MULTISURFRECON has no self-intersections in the entire volume. Conversely, if the output has no self-intersections in the volume, it does not have any overlap on any intermediate plane.*

*Proof.* For the forward direction, consider the stack of all the original and intermediate planes in the output. By hypothesis, there are no overlaps and hence no surface intersections on any of these planes. Further, between any pair of consecutive planes there are no branching problems, as otherwise the planes would not be consecutive in the stack. Therefore, the surface between consecutive planes is a linear interpolation of the contours by Criterion 2 from Section 2.2. By Assumption 2 from Section 2.1, the true surface is a 2-manifold in this region meaning the linear interpolation is not self-intersecting. Therefore, the entire output is not self-intersecting. For the converse, if the output is not self-intersecting, the components do not intersect each other and therefore do not overlap on any intermediate plane.  $\square$

By Lemma 2, the removal of simple transversal and exotic overlaps done in Step 4 of the algorithm suffices to separate the components of the forest of branched surfaces in the entire volume. In future work, we plan to examine whether knowing the type of overlap on an intermediate plane can be used to simplify the procedure of Step 4.

## 4 Implementation and Results

We show examples of our final, quality improved meshes in Figures 2 and 8. The entire computational pipeline - imaging  $\rightarrow$  contours  $\rightarrow$  tiling  $\rightarrow$  quality improvement - is handled by the Volume Rover (VolRover) [17] software developed by our lab, including a library for the Level Set Boundary Interior and Exterior Mesher (LBIE) [16].



**Fig. 8.** (a) A quality-improved mesh of three components - a dendrite shown in grey and two axons. (b) A zoomed in portion of the meshes.

## 5 Conclusion and Future Work

In this paper, we have presented a solution for creating isotopic meshes resolving the forest of branching structures problem and have demonstrated the feasibility of our approach. This is only the first step, however, in the simulation-based

morphological studies we plan to explore. With topologically accurate and error-bounded meshes of neuronal forests, we can begin to quantify properties of these forests in a variety of manners, e.g. computing the packing density of the cells in the volume and measuring the size and dimension of various spines along the dendritic structures. These quantitative measurements can be used for comparison among different brain samples; studies have already found that certain neurological disorders correlate with atypical dendritic spine formations and densities [21]. Additionally, the meshes we create can be used to simulate voltage potentials traveling along a dendrite or axon. The models will be calibrated with real electrophysiological measurements and then used to study morphological dependence on neuronal potentials.

**Acknowledgements.** We would like to thank our collaborators at UT Austin, Drs. Kristen Harris, Daniel Johnston, and Clifton Rumsey from the Section of Neurobiology and the Center for Learning and Memory for their high resolution data and the numerous discussions on the dependence of neuronal morphology and plasticity on synaptic activity where this meshing effort would have utility. Thanks also to Drs. Justin Kinney, Thomas Bartol, and Terrence Sejnowski of the Salk Institute for their help and encouragement in producing quality meshing results. Finally, thanks to Jose Rivera for his assistance in generating many of the images in this paper. This research was supported in part by NSF grants 0636643, IIS-0325550, CNS-0540033 and NIH contracts P20-RR020647, R01-EB00487, R01-GM074258, R01-GM07308.

## References

1. Armstrong, M.A.: *Basic Topology*. Springer, New York (1983)
2. Attali, D., Boissonnat, J.-D., Edelsbrunner, H.: Stability and computation of medial axes: a state of the art report. In: Möller, B.H.T., Russell, B. (eds.) *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Mathematics and Visualization. Springer, Heidelberg (2007)
3. Bajaj, C., Xu, G., Holt, R., Netravali, A.: Hierarchical multiresolution reconstruction of shell surfaces. *Comput. Aided Geom. Des.* 19(2), 89–112 (2002)
4. Bajaj, C.L., Coyle, E.J., Lin, K.-N.: Arbitrary topology shape reconstruction from planar cross sections. *Graph. Models Image Process.* 58(6), 524–543 (1996)
5. Bajaj, C.L., Xu, G.: Regular algebraic curve segments (iii): applications in interactive design and data fitting. *Comput. Aided Geom. Des.* 18(3), 149–173 (2001)
6. Barequet, G., Goodrich, M.T., Levi-Steiner, A., Steiner, D.: Straight-skeleton based contour interpolation. In: *SODA 2003: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, pp. 119–127. Society for Industrial and Applied Mathematics (2003)
7. Barequet, G., Sharir, M.: Piecewise-linear interpolation between polygonal slices. In: *SCG 1994: Proceedings of the tenth annual symposium on Computational geometry*, pp. 93–102. ACM, New York (1994)
8. Barequet, G., Vaxman, A.: Nonlinear interpolation between slices. In: *SPM 2007: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pp. 97–107. ACM, New York (2007)

9. Boissonnat, J.-D.: Shape reconstruction from planar cross sections. *Comput. Vision Graph. Image Process.* 44(1), 1–29 (1988)
10. Boissonnat, J.-D., Memari, P.: Shape reconstruction from unorganized cross-sections. In: *SGP 2007: Proc. of the fifth Eurographics symposium on Geometry processing*, pp. 89–98 (2007)
11. Bresler, Y., Fessler, J.A., Macovski, A.: A bayesian approach to reconstruction from incomplete projections of a multiple object 3d domain. *IEEE Trans. Pattern Anal. Mach. Intell.* 11(8), 840–858 (1989)
12. Chazal, F., Lieutier, A.: Stability and homotopy of a subset of the medial axis. In: *Proc. 9th ACM Sympos. Solid Modeling and Applications*, pp. 243–248 (2004)
13. Cheng, S.-W., Dey, T., Ramos, E., Ray, T.: Sampling and meshing a surface with guaranteed topology and geometry. In: *SCG 2004: Proc. of the 20th Annual Symposium on Computational Geometry*, pp. 280–289 (2004)
14. Cheng, S.-W., Dey, T.K.: Improved constructions of delaunay based contour surfaces. In: *SMA 1999: Proceedings of the fifth ACM symposium on Solid modeling and applications*, pp. 322–323. ACM, New York (1999)
15. Christiansen, H.N., Sederberg, T.W.: Conversion of complex contour line definitions into polygonal element mosaics. *SIGGRAPH Comput. Graph.* 12(3), 187–192 (1978)
16. CVC. LBIE: Level Set Boundary Interior and Exterior Mesher,  
<http://cvcweb.ices.utexas.edu/ccv/projects/project.php?proID=10>
17. CVC. Volume Rover,  
<http://cvcweb.csres.utexas.edu/ccv/projects/project.php?proID=9>
18. Dailey, M.E., Smith, S.J.: The Dynamics of Dendritic Structure in Developing Hippocampal Slices. *J. Neurosci.* 16(9), 2983–2994 (1996)
19. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*. Springer, Berlin (1997)
20. Edelsbrunner, H., Shah, N.: Triangulating topological spaces. *Intl. Journal of Comput. Geom. and Appl.* 7, 365–378 (1997)
21. Fiala, J., Spacek, J., Harris, K.M.: Dendritic spine pathology: Cause or consequence of neurological disorders? *Brain Research Reviews* 39, 29–54 (2002)
22. Fuchs, H., Kedem, Z.M., Uselton, S.P.: Optimal surface reconstruction from planar contours. *Commun. ACM* 20(10), 693–702 (1977)
23. Garland, M.: QSlm,  
<http://graphics.cs.uiuc.edu/~garland/software/qslim.html>
24. Geiger, B.: Three-dimensional modeling of human organs and its application to diagnosis and surgical planning. Technical report, INRIA (1993)
25. Goswami, S., Gillette, A., Bajaj, C.: Efficient delaunay mesh generation from sampled scalar functions. In: *Proceedings of the 16th International Meshing Roundtable*, pp. 495–511. Springer, Heidelberg (2007)
26. Harris, K., Perry, E., Bourne, J., Feinberg, M., Ostroff, L., Hurlburt, J.: Uniform serial sectioning for transmission electron microscopy. *J Neurosci.* 26(47), 12101–12103 (2006)
27. Herman, G.T., Zheng, J., Bucholtz, C.A.: Shape-based interpolation. *IEEE Comput. Graph. Appl.* 12(3), 69–79 (1992)
28. Klein, R., Schilling, A., Strasser, W.: Reconstruction and simplification of surfaces from contours. *Graph. Models* 62(6), 429–443 (2000)
29. Lee, D.T.: Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-4*(4), 363–369 (1982)
30. Lieutier, A.: Any open bounded subset of has the same homotopy type as its medial axis. *Computer-Aided Design* 36, 1029–1046 (2004)

31. Liu, L., Bajaj, C., Deasy, J.O., Low, D.A., Ju, T.: Surface reconstruction from non-parallel curve networks. *Computer Graphics Forum* 27(2), 155–163 (2008)
32. Meyers, D., Skinner, S., Sloan, K.: Surfaces from contours. *ACM Trans. Graph.* 11(3), 228–258 (1992)
33. Narayanan, R., Johnston, D.: Long-term potentiation in rat hippocampal neurons is accompanied by spatially widespread changes in intrinsic oscillatory dynamics and excitability. *Neuron* 56, 1061–1075 (2007)
34. Oliva, J.-M., Perrin, M., Coquillart, S.: 3d reconstruction of complex polyhedral shapes from contours using a simplified generalized voronoi diagram. *Computer Graphics Forum* 15(3), 397–408 (1996)
35. Park, M., Salgado, J.M., Ostroff, L., Helton, T.D., Robinson, C.G., Harris, K.M., Ehlers, M.D.: Plasticity-induced growth of dendritic spines by exocytic trafficking from recycling endosomes. *Neuron* 52(5), 817–830 (2006)
36. Sorra, K.E., Harris, K.M.: Overview on the structure, composition, function, development, and plasticity of hippocampal dendritic spines. *Hippocampus* 10(5), 501–511 (2000)
37. Turk, G., O’Brien, J.F.: Shape transformation using variational implicit functions. In: *SIGGRAPH 2005: ACM SIGGRAPH 2005 Courses*, p. 13. ACM, New York (2005)
38. Zhang, Y., Bajaj, C., Xu, G.: Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering* 24 (in press, 2008)

## A Medial Axis Definition

There is some disagreement in the literature about the definition of the medial axis as its intuitive notion is not the most general or accurate formulation. We will give the most general definition of the medial axis as used in [30, 12]. Let  $O$  be an open subset of  $\mathbb{R}^n$ . The *medial axis*  $\mathfrak{M}$  is defined to be the set of points  $x \in O$  for which there are at least two closest points to  $x$  on the complement  $O^c$ . For the 2D images of cells we consider,  $O$  will denote the union of all inter- and intracellular regions. This makes  $O^c$  exactly the same as  $\partial O$ , the boundary of  $O$ , which is the collection of closed curves meant to represent the cellular boundaries. Note that authors frequently refer to the “medial axis of  $\partial O$ ” to mean  $\mathfrak{M}$  (or sometimes just a subset of  $\mathfrak{M}$ ) so our definition encompasses commonly held notions. The *skeleton*  $S$  is defined to be the locus of centers of maximal inscribed balls in  $\mathbb{R}^n \setminus \partial O$  where a *maximal ball* is an open ball in  $\mathbb{R}^n \setminus \partial O$  which is not contained in any other open ball in  $\mathbb{R}^n \setminus \partial O$ . In two dimensions,  $\mathfrak{M}$  and  $S$  are often nearly identical and so we will treat them as such; we refer readers to [2] for a careful comparison of the two concepts.

