AUTOMATED 3D RECONSTRUCTION OF NEURONAL STRUCTURES FROM SERIAL SECTIONS

A Thesis

by

BRENT P. BURTON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 1999

Major Subject: Computer Science

AUTOMATED 3D RECONSTRUCTION OF NEURONAL STRUCTURES FROM SERIAL SECTIONS

A Thesis

by

BRENT P. BURTON

Approved as to style and content by:

Bruce H. McCormick
(Chair of Committee)

Nancy M. Amato
(Member)

Donald H. House
(Member)

Wei Zhao
(Head of Department)

August 1999

ABSTRACT

Automated 3D Reconstruction of Neuronal Structures from Serial Sections. (August 1999)

Brent P. Burton, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Bruce H. McCormick

A fast automated system for tracing neurons in parallel is described, adequate to support a quantitative analysis of neuron morphology. The system described here automates digitized neuron feature extraction and reconstruction, thereby replacing current largely manual techniques for tracing individual neurons.

Serial sections of brain tissue are created by physical sectioning. Sections are processed during scanning to determine regions of interest (ROIs) and to quickly cull unnecessary image data. An aggressive data culling and compression scheme reduces the original volumetric data into a ROI-based image collection that makes temporary secondary storage feasible. Neighboring ROIs are then matched from successive sections for segment and fiber tracing. Reconstructed segments created from these matches are disambiguated, resulting in an abbreviated structural description of the tissue's neurons and fiber tracts.

ACKNOWLEDGMENTS

I thank Dr. Bruce H. McCormick for his ideas, patience, and unending help on this work. I am also grateful to my wife Michele for her constant support and help, and to my parents for their encouragement throughout. Thanks for standing behind me.

Software development for this project was made simpler by several freely-available software packages. Thanks to Linus Torvalds for the Linux operating system, Brian Paul for his MesaGL library, Jef Poskanzer for releasing the NetPBM raster image library, and the Free Software Foundation for their GNU compilers, libraries, and tools.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Since the late 1960's, neuroscientists have investigated various techniques for neuron reconstruction. Understanding neuron morphology requires measureable data or, at the least, an adequate visual model. Such models were not easily created and early representations were artists' renderings, solid wax models, and film-mounted animations. The growth of digital computing saw camera-lucida techniques combined with the computer's random-access storage to record detailed neuron morphology. All of these techniques enhanced visual understanding of neuron structure but they were neither convenient nor fast. For instance, solid wax models took months to create.

The three-dimensional (3-D) reconstruction of neurons has remained largely manual, despite the rapid advancement of computer technology. Current techniques for tracing neurons and mapping their mutual connections are too slow to support an adequate quantitative analysis of brain morphology at the cellular and tissue levels. On a per-neuron basis, locating and staining requires several minutes while manual tracing may require several days (40 hours), even with computer assistance [1].

Automating the tracing procedure would enable the construction of large databases of traced neurons, providing support for improved models of neuron morphology. However, each scanned block of stained neural tissue creates an enormous volumetric data set. Efficient ways of representing, managing and processing this data are crucial to automating

Journal model is *IEEE Transactions on Visualization and Computer Graphics.*

neuron tracing. One method attempts this by reducing planar contour information into a triangulation, which is extended to create matches in adjacent slices [2]. For large data sets, comparing each section in this manner is unnecessary.

This thesis presents the development of an automated neuron tracing system. Chapters II and III introduce the project. Chapter IV covers the data set types used during system development, while Chapters V through VII discuss the preparatory steps of the data analysis prior to reconstruction. The reconstruction process is detailed in Chapters VIII through X, and finally results and conclusions follow in Chapters XI and XII, respectively.

CHAPTER II

OBJECTIVES

A complete automated reconstruction system consists of two major components: (1) the data acquisition subsystem (mostly hardware), and (2) the data reconstruction subsystem (software)[1] [2].

The objectives of the 3-D reconstruction system are:

1. Parallel tracing and reconstruction of dendritic processes and fibers.

2. Data compression for maximal scan rate.

3. L-System representation of neurons and fibers.

4. Automating the three-dimensional reconstruction.

A Parallel Tracing and Reconstruction of Dendritic Processes and Fibers

Serially scanned at the limit of optical resolution (250nm), cortical tissue can create large volumetric data sets representing the tissue. The reconstruction system is expected to operate on volumetric data digitized at 150nm to 1000nm per pixel. For example, a section of rat brain sampled at the limit of optical resolution requires 3.2GB. To compress the size of these data sets, each section must be filtered and segmented, saving a concise structural representation of all tissue features. After image segmentation, the structural

[1] The Scientific Visualization Laboratory does not presently have the equipment to implement a full reconstruction system; hence the research described here emphasizes the data reconstruction subsystem.

[2] Scientific Visualization Laboratory is in the Department of Computer Science, Texas A&M University, College Station, TX 77843-3112.

representations are processed sequentially, following dendritic processes and fiber tracts in parallel from section to section.

B Data Compression for Maximal Scan Rate

The large volumetric data sets pose storage and access problems due to the amount of data acquired. McCormick [3] describes a tissue scanner design with a maximum data rate of 100Mpixel/s into the data acquisition computer. Data compression must be used during acquisition to match the slow 5MB/s transfer rate of secondary storage. The required effective compression ratio of at least 20:1 also reduces the storage requirements of archival tertiary storage. Data compression, combined with the data reduction inherent with image segmentation, can satisfy this transfer and storage requirement.

C Automating the Three-dimensional Reconstruction

Neuroscientists have identified neuron reconstruction as a candidate for automation [4, 5]. Past projects, some of which have been described as automated, are best considered aids to the manual process, in which a human operator is required [6, 7].

Large scale neuron tracing is only possible when the process can be completed with little or no human direction. Tissue scanning, image processing, and feature detection have been automated in other projects [8, 9]. Availability of precision computer-controlled 3-D stages (developed for VLSI manufacturing), sectioning mills, high-resolution confocal laser scanning microscopes, and CCD cameras ease the mechanical limitations associated with interfacing hardware and software [9, 3]. Once traced, the final transformation of the volumetric data set into its parametric representation as a forest of neurons and fibers involves data smoothing and modeling, fully automatable processes.

The methods developed here extend to fiber tract tracing, but there is limited experience in automating that process. Fiber tracing, furthermore, may require some user input to properly guide the search. Such assistance should later prove valuable in developing heuristics to achieve fully automated tracing of fiber tracts.

CHAPTER III

BACKGROUND

In the past, various techniques have been applied to 3-D neuron reconstruction.

In 1972, Levinthal and Ware created neuron reconstructions and "fly-throughs" by aligning tissue slices and transferring them to 35mm film, creating an animation [10]. A computer was used to manually trace the neuronal processes and store information at each point. After this data collection, the computer generated 3-D stick models for analysis and visualization. Surface modeling was not available as process contours were not recorded from the sectioned tissue. In this work, the computer automated nothing, but served as a data collection and presentation tool.

In 1974, Rakic et. al. used reconstruction techniques to analyze developing neuron morphology in the fetal monkey (Macaque rhesus) [11]. Tissue was sectioned into 80nm thick slices, and the electron microscope images were manually traced by technicians due to the "considerable human judgement involved." Automated feature tracing was deemed not worthwhile to implement. Visualization of the reconstructed cells was performed in 3-D by drawing each contour line with depth-cueing (closer lines are thicker and farther lines thin). Analysis of each contour included determining volume and surface area of the slice.

Neurolucida[TM] and past commercial systems rely on user interaction for feature isolation. Neuron tracing, in this case, is aided by computer but not automated.

Reconstruction of 3-D structures solely by computer has been thoroughly investigated using an isosurface approach, where the 3-D data set is analyzed to determine a surface matching distinguishing criteria [12, 13]. Such surfaces are routinely generated from MRI data, where the isosurface represents a given tissue density (bone, flesh, etc). Lorensen and Cline developed the marching cube algorithm for generating such surfaces, and their algorithm has been implemented in many visualization systems [14, 15]. Isosurface generation (reconstruction) has also been applied to neurons with good results [4]. However, in all of the isosurface techniques, no underlying structure is extracted from the data, only an outer surface representation.

Boissonnat and Geiger used planar Delauney triangulation to summarize a contour's shape [2]. Then, by extending tetrahedra into the neighboring section, they create matches. This method also only builds an isosurface and is dependent upon processing every slice with a pairwise comparison. Their intention was to process volumetric data without having to process all the data, and while the algorithm achieves this goal, it is not suitable for extracting structural information.

On the biological side, staining techniques are improving, making the automated identification of regions easier. Of note is the local staining technique of Gerfen and Sawchenko [16]. Their PHA-L stain completely identifies isolated neurons, with no leakage into nearby passing fibers, and produces a high-contrast stain.

CHAPTER IV

MODELS OF THE DATA SET

To aid the development of reconstruction algorithms, simulated data is needed. Simulated data sets contain known geometry and reconstruction results can be accurately verified. Nslicer and Datagen are two custom tools written to create simulated data.

A Requirements of Simulated Data

While developing neuron tracing algorithms, "clean" data sets were needed to test the reconstruction algorithms. The main benefits to using well-defined data are:

1. No image processing or pattern recognition tasks are required. Development effort can be directed to the tracing and reconstruction algorithms proper, leaving image processing as a separate problem. In the case of serial sections from transmission electron microscopy (TEM), image filtration and "closed-object operators" have already been defined to enhance neural tissue features [17].

2. The algorithms can be tested against specific situations. The use of specific scenarios over sectioned neuron models allows concentration on "problem situations," a possible but most likely infeasible task using full data sets.

3. The reconstruction can be easily verified manually or with visualization tools. Specialized data sets are sufficiently small to trace manually as the reconstruction program reports interconnections. If further verification is needed, visualization tools can superimpose reconstructed models over the original neuron geometry.

In short, small, targetted data sets allow rapid development and testing of reconstruction algorithms.

B Neuron Slicing with the Nslice Tool

This section describes a tool that simulates sectioned data from existing neuron models. Nslicer ("neuron slicer") is a tool that reads existing descriptions of 3-D neuron models and creates a digital image sequence that simulates sectioned tissue. Accurate neuron models of pyramidal and motor neurons have been created by Mulchandani [18], and software to create 3-D models from these descriptions is freely available over the Internet (the L-Sys program [19]). Nslicer is written in C and utilizes the OpenGL graphics library for rendering [20].

Nslicer loads the neuron description file, building a cylindrical representation of the neuron's processes. Neural processes are represented by a number of 3-D line segments, with each segment modeled by polygonal cylinders of 5 to 9 sides. The entire collection of polygons is saved into a callable OpenGL display list for ease of rendering. An OpenGL display list allows a program to treat complex geometrical objects as a single entity and render them in a single step without reprocessing data to build the object.

As the description is read, X, Y and Z maxima and minima are recorded for configuring the graphics display and determining bounds of the sectioning. The coordinate system used is a right-handed system with Z coming out of the screen, X horizontal to the right, and Y up. Once loaded, Nslicer sets the position to the maximum Z value, which is "in front" of the neuron(s) from the camera's point of view.

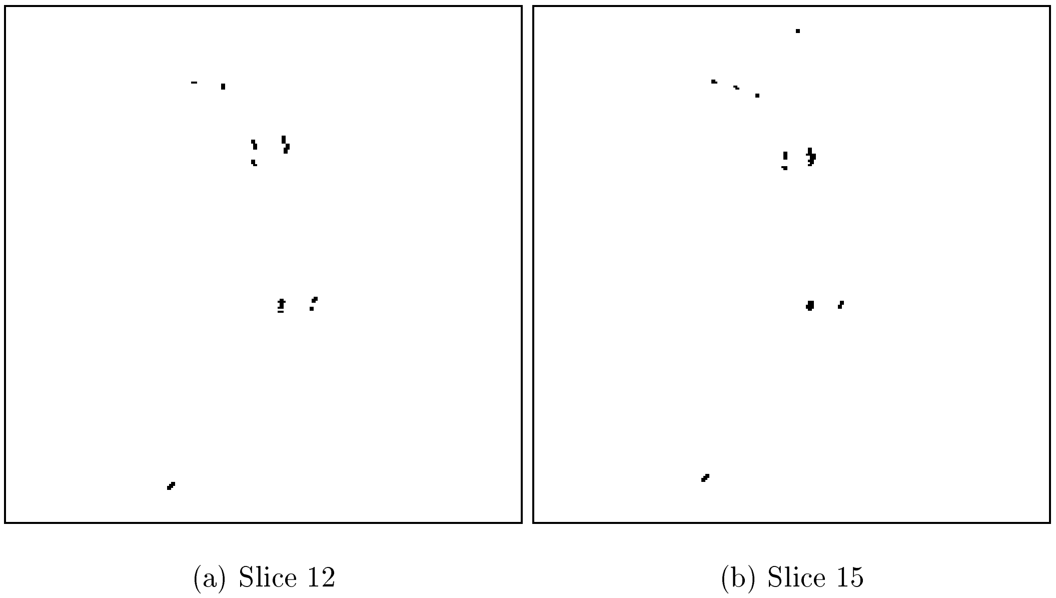
Sectioning is performed by setting OpenGL's front and back clipping planes close together, and moving the Z position from maximum (in front) to minimum (in back), rendering the neuron segments at each step. Each image is retrieved from OpenGL's framebuffer and saved to disk in the PPM format [21]. Four sequential sample slices taken from a pyramidal model are displayed in Fig. 1.

Section thickness is set to simulate a 1μm sectioning depth according to the neuron model's morphological type and its Z-length. For example, a pyramidal cell model (average span of 250μm), with its axis lying in the X-Y plane, might have Z maximum of 13 and minimum of -11 yielding a simulated 1μm section by setting the clipping planes $(13 + 11)/250\mu m = 0.096$ apart in world coordinates.

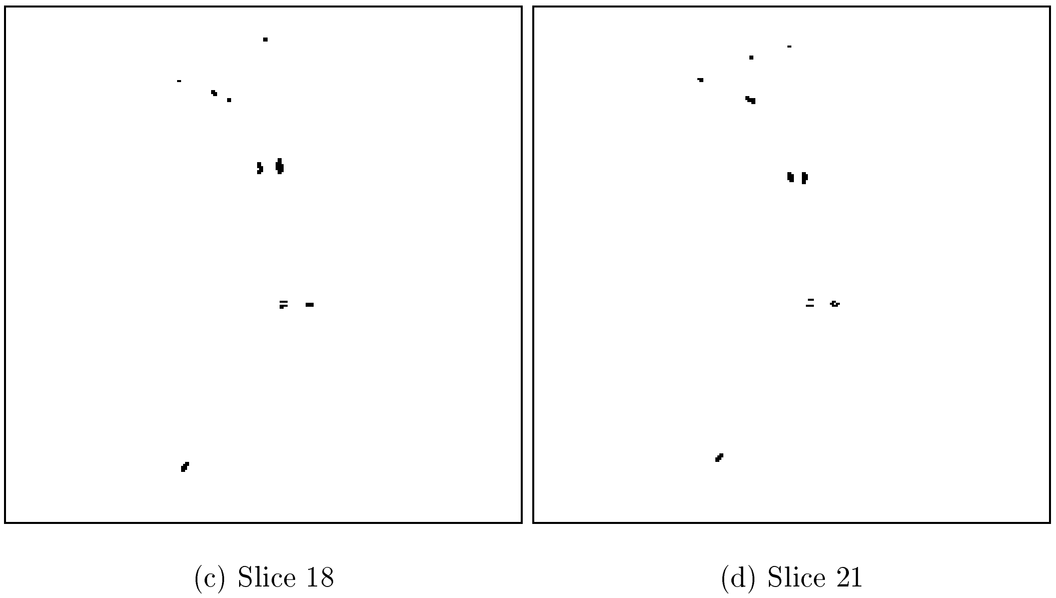
As a development tool, Nslicer's resulting image sequence leaves much to be desired. First, the polygonal modeling technique creates rendering artifacts which are not conducive to early developmental reconstruction systems. The artifacts, primarily discontinuous features, create image segmentation problems and complicate algorithm development, although using a higher image resolution would improve data quality. However, these artifacts can be reduced by simply rendering cylinder models with more than 5 to 9 polygons. Second, sliced images from established models are too complex for algorithm testing. To aid reconstruction algorithm development, a tool to generate situational data sets would be better. Nslicer's place, however, may be for testing established algorithms against more complex scenarios.

C The Datagen Tool

The need for improved test data sets prompted the development of another tool to create serial section data. This section describes the Datagen tool that, like Nslicer, creates serial section data sets, but does so from mathematically defined dendritic segments and fibers, not existing neuron models. In addition to serially-sectioned image data, Datagen



(a) Slice 12 (b) Slice 15



(c) Slice 18 (d) Slice 21

Fig. 1. Sections 12, 15, 18, and 21 produced by Nslicer from a sample neuron.

also has the ability to write out neuron geometry files, in the same format that Nslicer reads.

C.1 Types of Data Created

Several data types are created by Datagen to provide a variety of test cases. These types are delineated below:

Stochastic pipe. This data type creates a single neuron segment initially parallel to the Z axis. As the segment extends through the sectioning planes, each X-Y position is displaced by a Gaussian jitter to simulate neuron process wandering. A derivative of this displacement is the concept of *coherence distance*, which is the average length along the segment where the initial and final X-Y positions are less than one segment diameter apart. The coherence distance imposes limits on predictive path finding.

Splits (bifurcations). This set creates a section sequence starting with a straight parent segment (not necessarily perpendicular to the sectioning plane) which splits into two or three daughter straight segments. The bifurcation angle is adjustable to test narrow and wide bifurcations.

Joins. The join data set merges two or three straight segments into a single segment.

Helix. The helix type creates a N-helix with the number of segments adjustable. The segments are parallel, evenly spaced, and concurrently revolve about the Z axis. This data set tests the ability of the tracing algorithms to follow a constantly changing trajectory while correctly discriminating between multiple segments.

Braid. Finally, the braid datatype goes a step beyond the helix data set. A flat braid of three segments is created along the Z axis, and is designed to test the ability of a tracing algorithm to predict subsequent contour intersections. Compared to the helix set, where a segment's direction vector uniformly rotates about a vertical axis, a braid's segment direction vector continuously changes. In addition, the braid segments are closer together, further testing discrimination techniques. During growth, bundles of neuronal segments can form tangled cables which this data set simulates. Presently set to create three tangled segments, this model can be enhanced to create any number of intermingled segments.

C.2 Superimposing Reconstructed Models over Original Models

By default, each of the data types created by Datagen results in a sequence of images. As an option, the five data types can be saved in the textual neuron file format, creating a geometric description of the image set. Once a data set is reconstructed, the resulting reconstructed geometry and the reference geometry files can be compared by visualizing both at the same time using translucent surfaces for rendering. Any discrepancies in the reconstruction are easily found using this technique.

CHAPTER V

DATA ACQUISITION

The previous chapter described how to synthesize data when data acquisition hardware is unavailable or when specific test cases are required during software development. This chapter describes the data acquisition process for existing microscopy hardware.

Achieving usable data for the reconstruction of neural structures is a multi-step process, described in Fig. 2.

$$\begin{tikzcd}\text{Tissue Staining} \arrow[r] & \text{Tissue Sectioning} \arrow[r] & \text{Digitizing} \arrow[r] & \text{Data Transfer} \arrow[r] & \text{Preliminary Filtering}\end{tikzcd}$$

Fig. 2. Overview of data acquisition.

A Tissue Staining

Tissue sectioning is preceded by a staining process that enhances neuronal features of the tissue. Depending on the type of microscopy and stain used, the stain adjusts the contrast or the reflectivity of the neuronal structures. Two stain techniques stand out for neuron reconstruction: Golgi and PHA-L.

Golgi staining was developed more than a century ago and remains one of the best contrast stain techniques developed. This method permits limited but accurate glimpses of the nervous system structure [22]. Its use is limited due to the lack of distribution control of the Golgi stain and once applied, only a small percentage of the cells is stained (one study found a high of 2.26%, with an average of 1.29% of the cells stained [23]). However, these stained cells have high contrast and detailed feature isolation, providing for accurate tissue observation.

The second method anterogradely stains tissue with PHA-L (*Phaseolus vulgaris*-leucoagglutinin), a kidney bean lectin. As described in [16], PHA-L offers many advantages over other staining techniques, the most relevant to neuron identification and reconstruction being the following. First, using iontophoresis, small sites (50-100μm) can be isolated and included neurons are completely filled with label. Second, PHA-L clearly stains morphological features such as cell bodies, axons, dendritic arbors and dendritic spines. Further, PHA-L is not absorbed and propagated by fibers of passage. PHA-L provides good identification and tight targetability, traits necessary for reliable neuron reconstruction and fiber tracing.

B Tissue Sectioning

Two techniques exist to section or slice tissue: optical sectioning and physical sectioning. Optical sectioning is performed with confocal laser scanning microscopy (CLSM), which uses a laser and a small aperture to control depth of focus. Tissue further or closer than that in the focal plane is undetected, thus the CLSM instrument allows selection of a specific layer of the tissue for imaging [24]. Physical sectioning, however, only provides imaging in a front-to-back manner, with the frontmost tissue layers physically removed to reveal deeper tissue. Physical sectioning is inherently destructive, but is not depth-limited.

C Digital Scanning

Electronic detectors and CCD cameras sample with 8 to 14 bits per pixel (bpp) providing from 256 to 16,384 intensity levels, with image resolutions being 512x512, 768x512 and 1024x1024 in practice. Sampling at 8 bits, these resolutions produce raw data of 0.25MB, 0.37MB, and 1.0MB per image. Newer CCD arrays offer 2Kx2K (1K=1024) and 4Kx4K resolutions, with linear detectors offering 8K scanning. Digital scanning of tissue is performed at a resolution of 150–1000nm per pixel.

The 1MB image sizes are quite easy to work with independently. However, when the area of the sampled tissue is larger than the current field of view, then many images taken at the current sectioning plane must be tiled together. The resulting mosaics are large (several megabytes to several gigabytes) and create processing problems for later steps.

D Data Transfer

The interface between the digital camera (or CLSM) and the data acquisition computer is a specialized hardware interface such as VME or a proprietary interface, as is common in PC-class machines. Data transfer is initiated by the computer, and for 1024x1024 images took 20 seconds per frame in older systems [25, 9].

More recent CCD technology drastically reduces these scanning times by two orders of magnitude, transferring video from a 1024x1024 CCD camera at 30Mpixels/s, or 0.33s per image [3]. Video advances have further pushed this to 100Mpixels/s [3]. The data acquisition computer captures the video feed and stores it initially directly in memory.

E Preliminary Filtering

Modern microscropy 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, as early as possible. Preliminary filtering performs noise reduction and image normalization to reduce the storage space of each image. Filtering also enhances features in the data.

Simple methods have been used in several systems to enhance image data. Image normalization through Laplacian filtering, histogram analysis and thresholding produces binary data that represents the basic skeleton of image features [9, 8]. Image erosion and dilation has been employed to remove small artifacts (air bubbles or dust) by min-selecting then max-selecting neighboring pixels [8, 26]. These methods have proven effective at increasing the signal-to-noise ratio in digitized images, making the algorithms required for image segmentation simpler.

CHAPTER VI

DATA RECONSTRUCTION

After data acquisition, the process of segment reconstruction begins. An implementation of the process was written and is called the Recon System, described in this chapter. The following chapters discuss each step of reconstruction and how they were implemented in the Recon System.

A Overview of Reconstruction

$$\begin{tikzcd}\text{Image Segmentation} \arrow[r] & \text{Segment Tracing} \arrow[r] & \text{Structural Representation} \arrow[r] & \text{Junction and Bend Detection}\end{tikzcd}$$

Fig. 3. Overview of data reconstruction.

The reconstruction process, illustrated in Fig. 3, is a data flow pipeline. After tissue sectioning and digitizing, the first step is image segmentation, which identifies and isolates ROIs. Once ROIs are created, segments are traced through consecutive sections by matching neighboring ROIs. This step generates ROI interconnection information, storing it in the structural representation. After segment tracing, the structural representation is scanned for junctions and bends to detect and resolve ambiguous situations.

B The Recon System

The Recon System implements the automated neuron reconstruction system using an object-oriented approach. The reconstruction problem is especially suited for an object-oriented design for three reasons: (1) the various data types have real world conceptual correspondences, (2) the interactions between the types and algorithms are clearly defined, and (3) rapid development and testing demands easy code changes. The interchangability provided by abstract and derived classes allows easy testing of different algorithms and internal representations, and supports multiple data formats with minimal source code changes.

The basic types of the Recon System correspond to the objects of the reconstruction problem. There are tissue sections, consisting of digital images and contours, and various algorithms, such as image segmenters and reconstruction algorithms. The more important objects are listed in Table 1 with descriptions of each type below.

TABLE 1

Object Definitions Used by the Recon System

Object name	Description	Contains...
Object	Basic object type, Superclass of all other objects.	
Section	Implements concept of a tissue section; manages Image and Contours for this tissue slice.	Image, Segmenter, Contour, and Z position in data set
Image	Abstract class for digital images.	
Segmenter	Abstract algorithm object to perform image segmentation.	
Contour	A sequence of Point2i's enclosing the ROI.	Point2i
OBB	Finds a minimal enclosing rectangle about a Contour.	
Segment	Sequence of 3D points to represent fiber trajectory.	Point2i with Section-Z
Reconstructor	Abstract algorithm object to perform reconstruction from Sections. Generates Segments.	
Point2[i,f]	2-D integer and real points.	

Object. This object acts as the superclass for all other object types used in this system.

This abstract class provides equality and identity checks for objects as well as polymorphism for data structures. This design creates a monolithic class hierarchy which is not required in C++, the language being used. However, should the Recon System be translated to another object-oriented language with a monolithic hierarchy such as Java, this design will minimize architectural changes.

Section. A Section object represents one tissue section and stores information about that section such as Z-position, thickness, the digital image corresponding to this section, and the list of feature Contours extracted from the image. It also ensures that the particular subclass of Image is correctly matched with a segmentation algorithm for that type (monochrome or grayscale).

Image. A digital image from the data acquisition subsystem is stored in an Image class. Presently, two Image subclasses exist for storing grayscale and bitmap images, respectively: PGMImage and PBMImage. An Image object also performs memory management.

Segmenter. This algorithm object performs image segmentation on the Section to which it is assigned. Segmenter objects detect neuron segment intersections and create a list of Contour objects representing the features. Subclasses of Segmenter operate on specific subclasses of Image.

Contour. The Contour type is generated by the image Segmenter class, and stores the boundary points from a neuron segment intersection. It maintains other information such as contour center, a convex hull of the boundary points, and a rotation that describe the Contour geometry for later path-following tests. Point data is recorded in a list of Point2i objects.

OBB. The OBB (oriented bounding box) object encapsulates an algorithm and data to determine a minimal, aligned rectangle that tightly encloses a Contour. This boundary is used to minimize secondary storage needs, and is fully described in Chapter VII.

Segment. During the segment tracing process, points along each segment path are saved in this class. Like the Contour object, it provides no functionality but is used for data organization.

Reconstructor. This abstract class defines a basic interface for specific reconstruction algorithms to follow. Reconstructor objects implement segment tracing algorithms, applying them to an array of Section objects. This type provides no data storage, and provides only algorithmic functionality. Subclasses of Reconstructor implement different reconstruction strategies. Reconstructors "read" Section objects and create a structural representation of the neuron segment paths.

Point2i, Point2f. These small objects represents a 2-D point and provide utility operations such as distance measurements. Integer and floating-point representations are used throughout the Recon System.

Several other utility objects exist for data collections and representations, but the above set is most representative of the reconstruction problem.

CHAPTER VII

SECTION SEGMENTATION

Section segmentation is the process of locating and extracting interesting features from section data.

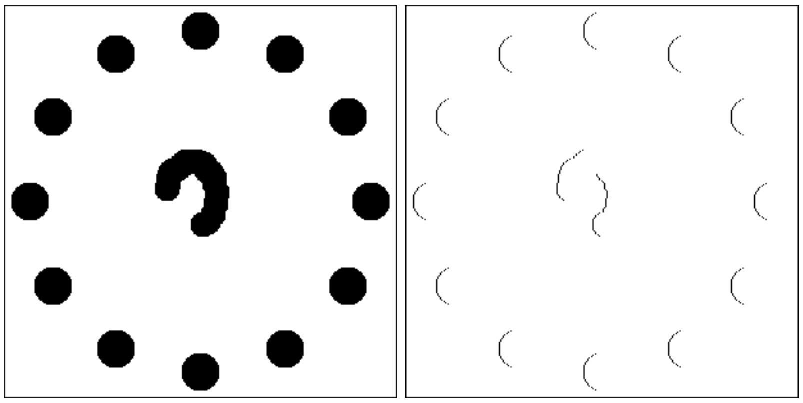
This chapter describes how features present in digitized section images are detected and how their enclosing boundary, the ROI, is determined. Using the ROI, data storage requirements are reduced.

A Feature Detection and Isolation

As mentioned in Chapter V feature detection and isolation is performed by image histogram analysis, thresholding, and erosion-dilation. Further, advanced algorithms based on the Hough transform have been developed to detect closed objects such as contour intersections [17].

Test data set images are monochrome images with black features on a white background. Segmentation involves determining the bounds and outline points of each feature, and returning the bounding contours generated.

A straightforward contouring algorithm is used with monochrome images. It starts by creating a blank "edge image" of the same resolution as the section, then it finds the left edges of image features by locating white-to-black color transitions. If pixel (i, j) is white and the right neighbor $(i+1, j)$ is black, the edge image's $(i+1, j)$ pixel is set black. After finding all transitions, the edge image contains relatively few black pixels. A section from a 12-segment helix data set (with a nonconvex feature added) and its left-edge image are shown in Fig. 4. To trace a contour, the algorithm searches each row from the top-left corner of the edge image to find a black pixel, (i_B, j_B). Starting at the corresponding point in the section image, the algorithm follows the white-black border clockwise around the feature, recording the points and eventually returning to (i_B, j_B). Direction checks during the trace prevent the algorithm from looping. As each point (i, j) is recorded, the (i, j) pixel in the edge image is set to white. This removes the edge from the edge image which means each contour can be detected and traced only once. After tracing a feature, the scan of the edge image for black pixels continues where it left off, at pixel (i_B, j_B). The contour tracing algorithm is capable of following convex and nonconvex shapes no matter how thin.



(a) Section Data (b) Left Edges

Fig. 4. A 12-segment section image and its left-edge image.

Data reduction consists of generating the convex hull of the contour points using the Graham scan algorithm (from [27]), which significantly decreases the number of contour points. Reducing the number of contour points accelerates later reconstruction tests.

B Creation and Storage of ROIs

A *region of interest* is simply a rectangular subimage that contains a feature to record. In the past, an ROI's boundaries were parallel to the X and Y image axes, but to reduce ROI storage requirements a bounding box oriented to the feature itself is used.

To prevent confusion, two terms will refer to the specific type of bounding box strategy used. "ABB" denotes the axis-aligned bounding box and "OBB" denotes the oriented bounding box aligned to the feature. "ROI" now refers to the region in general, which is specifically described by either the OBB or ABB. OBBs used in the Recon System are a 2-D simplification of the OBB-Trees developed by Gottschalk et. al. [28].

Once a feature has been isolated and its boundary determined by image segmentation, the ABB is created by finding the minimum and maximum X and Y values of the feature's boundary points. An example feature is in Fig. 5. In Fig. 6, the feature's elliptical boundary is contained by the ABB.

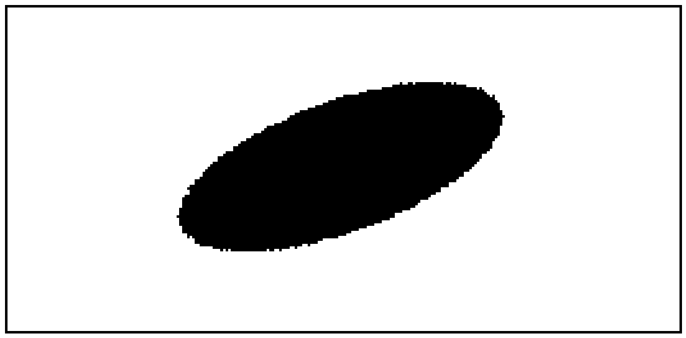


Fig. 5. Elliptical contour of a neuronal segment.

Next, the OBB is created by the following steps. The major axis of the feature's contour is determined by a least-squares line fit to the contour points. A coordinate system is then attached to the feature based on this axis, and a transformation from image coordinates to OBB coordinates is created (called "img2obb" in the Recon System), along with the

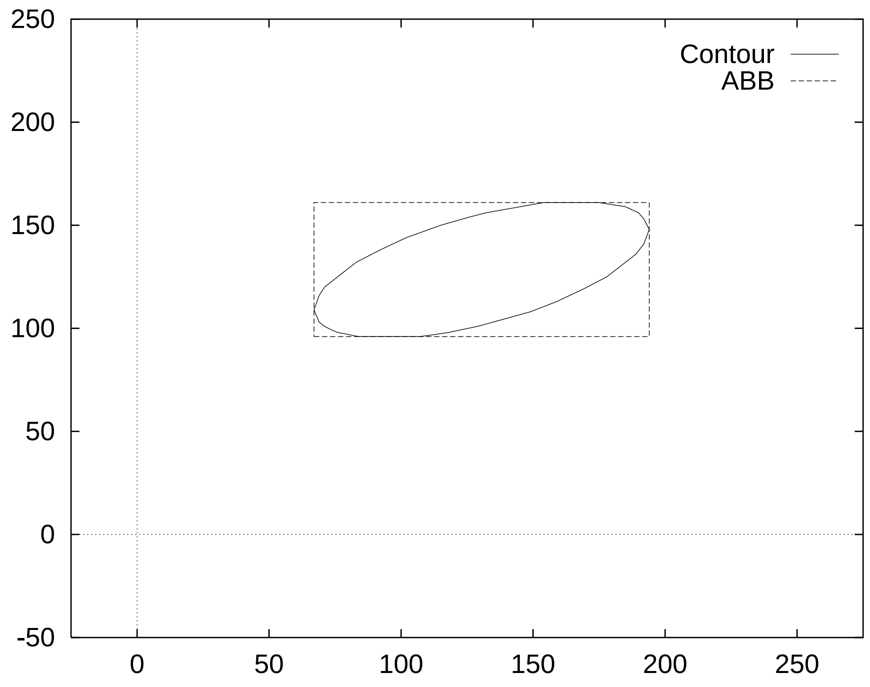


Fig. 6. ABB creation as the min-max box of a contour.

inverse transformation (referred to as "obb2img"). The transformation describes a rotation and translation into the local OBB coordinates. The contour points are then transformed into local OBB coordinates to determine the size of the oriented bounding box. Fig. 7 shows the contour, its main axis, and the associated OBB. The contour and its ABB and OBB are illustrated in Fig. 8. In this example, the OBB has a smaller area than the original ABB.

To prevent unnecessary work, two checks are performed after determining the OBB. First, if the OBB's major axis is aligned with either the section's X or Y axis, no rotation is specified. A second check compares the area enclosed by the OBB with that enclosed by the ABB. If the area in the OBB is smaller than the ABB's area, then the OBB is choosen for the ROI. Otherwise the ABB is selected to represent the ROI. Both tests prevent unproductive filtering and the latter ensures the smallest amount of image data containing a contour is saved.

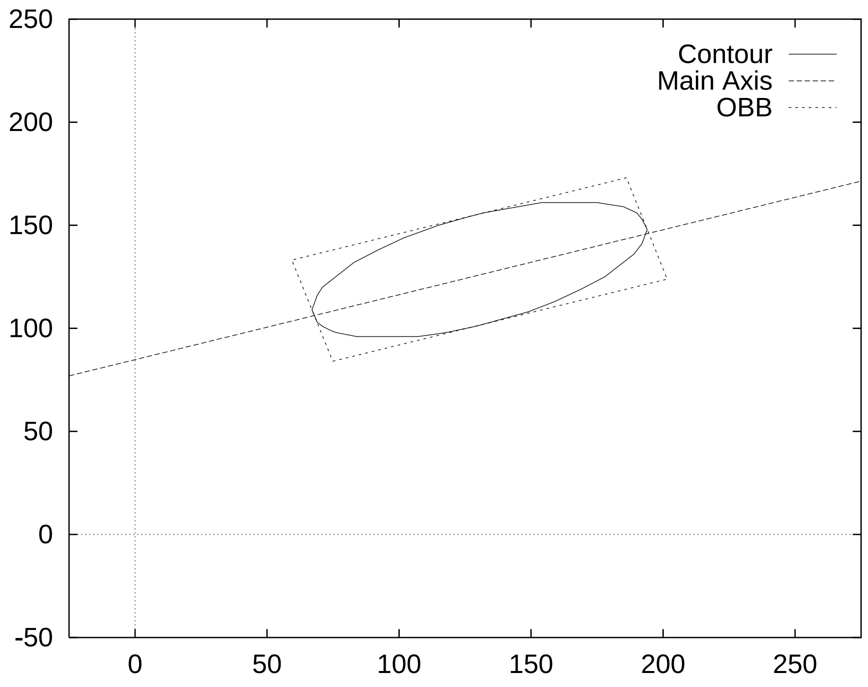


Fig. 7. OBB creation as an oriented min-max box of a contour.

Each ROI contains one contour and is assigned a unique serial number. The ROI serial number is used to identify ROIs and acts as a key for later retrieval. A description is also written that describes the ROI's rotation (if any) and position in the original section image. Finally, the subimage data contained within the ROI is extracted and saved through a straightforward map.

C Data Compression

After determining the ROI, the enclosed image data is extracted, saved, and compressed. Compression is by the Limpel-Ziv algorithm as implemented in the GNU zlib library [29]. Data culling via OBB processing increases the effective compression ratio by reducing the amount of image data saved.

Dense brain tissue such as the human cortex contains approximately 100,000 neurons in a 1mm x 1mm x 3mm volume. A 1024x1024 scan of the section at 0.3μm/pixel covers

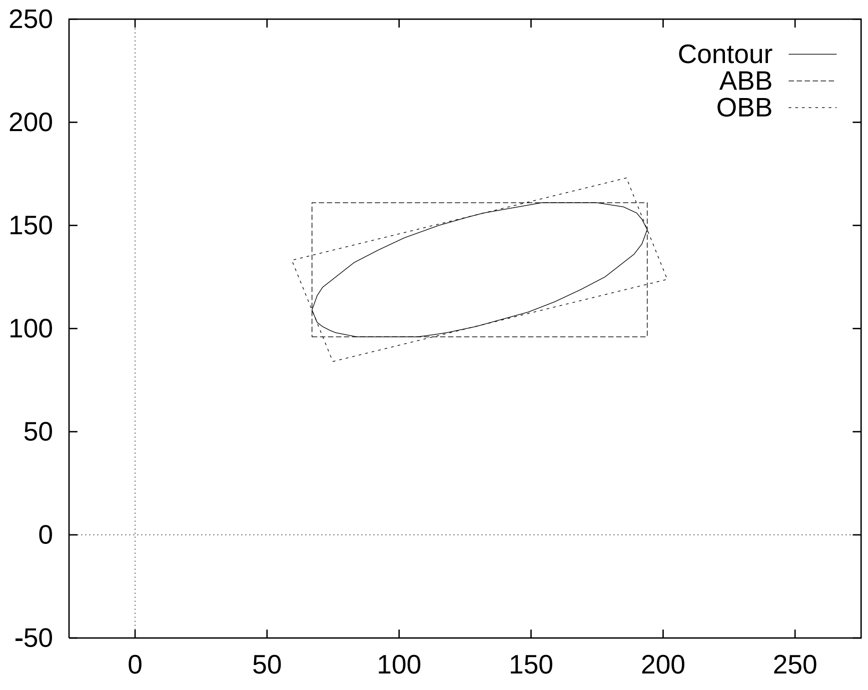


Fig. 8. The feature's contour, ABB, and OBB.

about 300x300μm^2, or about 0.10mm^2. Golgi staining labels 2% of the present neurons, so the total number of segments to expect in one image is:

$$100,000 \text{ neurons/mm}^2 \times 0.10\text{mm}^2 \times 2\% = 200 \text{ neurons.}$$

This density of 10,000 segments per 1024x1024 image translates to

$$\frac{\text{1024x1024 pixels}}{10,000 \text{ segments}} \approx 104 \text{ pixels/segment}$$

or an ROI dimension of 10x10. At 0.30μm per pixel, this ROI is from a segment of 3μm diameter, assuming the segment is perpendicular to the sectioning plane.

Using this size, compression ratios can be determined for the section containing 200 contours. Table 2 lists two cases of 3μm fibers: the 10x10 ROI size represents a segment perpendicular to the sectioning plane, while the 14x10 ROI size is from a segment at a 45^o angle to the plane. The basis of the ratio is a 1024x1024 image, 1MB in size.

TABLE 2

Effective Compression Ratios for 200 Segments

Fiber	ROI	Uncompressed		Compressed	
diameter	resolution	ROI size	Ratio :1	ROI size	Ratio :1
3μm	10x10	158	33	106	49
3μm	14x10	198	26	133	39

CHAPTER VIII

SEGMENT TRACING

Which ROIs should be bundled together to form a segment lends itself to a variety of selection algorithms. A simple greedy algorithm does not provide the discriminating ability of more sophisticated algorithms. The various segment tracing methods investigated are described in this chapter.

A Minimal Distance Selection

In the minimal distance selection algorithm, a reconstructed segment's next point is selected by finding the ROI of the next section with the minimal X-Y planar displacement from the current segment position. In other words, the contours' center positions from section n are projected onto section $n + 1$ to determine 2-D distances. Segment positions are denoted by each segment's center which is determined by averaging that contour's convex hull points. As each contour match is tested, the pair with minimal distance between centers is selected.

Selecting minimal distance is context-free. At each section, the segment's next contour is picked without considering its current trajectory. Further, this technique handles branch points poorly.

B Intersecting Contour Selection

The second algorithm determines pathways by simply finding overlapping contours from adjacent sections. Overlaps are found by testing if the contour outlines (or more efficiently, their convex hulls) intersect. While in general it is not accurate, the algorithm becomes more accurate if the sectioning thickness is sufficiently thin to allow these contour intersections to exist.

A problem with this and the previous algorithm is the inherent dependence upon thin sectioning for accurate results. Thin sectioning minimizes the X-Y offset of neighboring ROIs for the minimal distance method and guarantees polygonal intersections in the second. Thin sectioning, by oversampling, also increases the volume of data, and it may be desirable to process only one of every several slices, which thwarts the benefits provided by thin sectioning.

Another problem with minimal distance selection is how easy the technique is fooled. Fig. 9 shows three contours overlaid from two sections. The solid circle represents a contour in the current section while the dashed circles are contours in the next section. Minimal-distance selection will choose the A-C pairing over the A-B match, although the correct match is A-B. Because contours A and B intersect, the contour-intersection method will select correctly. This situation also illustrates that differences in contour size between successive sections is another factor to consider in the match test. The present implementation in the Recon System does not perform this check.

Both techniques are improved by limiting the search to regions proportional to current segment diameter, but more tolerant techniques, discussed next, exist.

C Path Prediction

The path prediction technique uses the past few ROI positions in each segment to predict where in the next slice a contour may be. The heuristic is that subsequent ROIs lie within some expected distance from the predicted point. Prediction uses polynomial

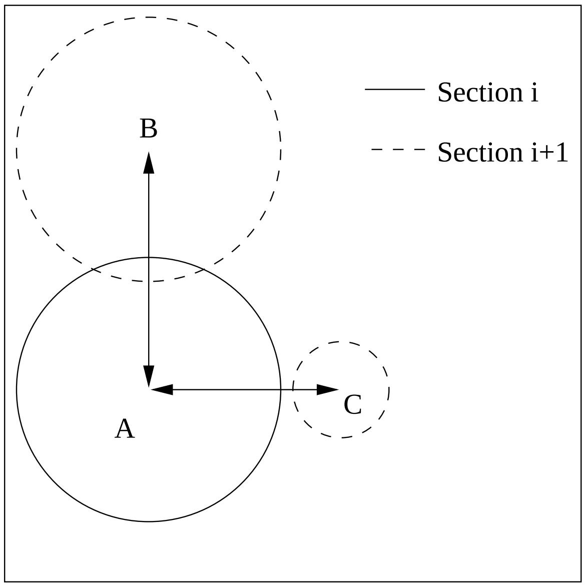


Fig. 9. Minimal distance selects pair A-C over A-B.

extrapolation based on the previous 3–5 segment positions. At each segment test, the polynomial is evaluated at the depth of the next section to create a test point, which is then tested to reside within a contour.

Two polynomials, parameterized by slice depth Z, are used to create a (X,Y) position in a subsequent slice. The X-function is interpolated through the X positions for the previous Z positions, and the Y-function is treated similarly. Neville's algorithm as implemented in [30] is used for generating the functions.

A predictive method initially has no basis for generating predictions, so this method uses the contour-intersection technique for the first few points. Once enough points are collected for extrapolation, the intersection method is no longer used.

D Improvements to Prediction

If neuron processes were guaranteed to be cylindrical, path prediction would be trivial. The cylindrical shape, when sliced by a sectioning plane, is elliptically distorted. The amount of distortion depends on the intersecting angle of the segment with the sectioning plane. Distortions for intersecting angles from 5^o to 85^o are shown in Table 3. The values in the table represent the length of the major axis of the feature; the minor axis size remains equal to segment diameter.

TABLE 3

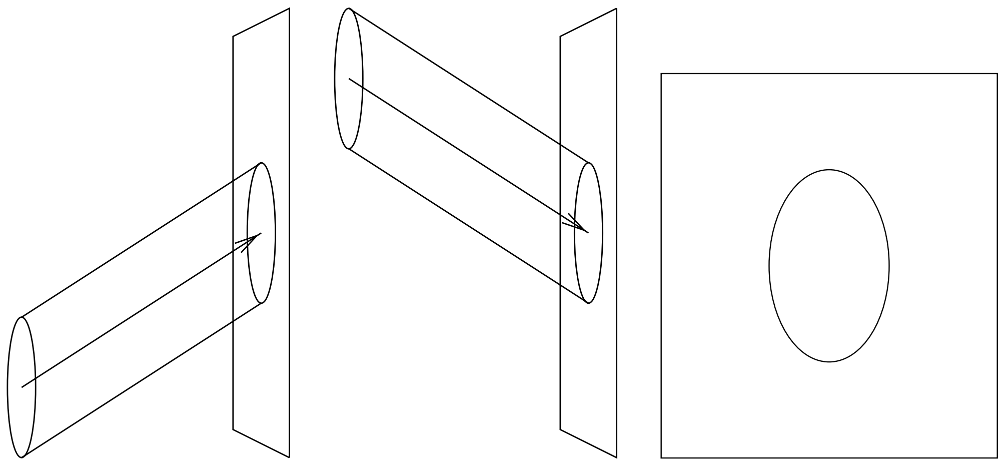
Projected Width of Segments with Respect to Segment Angle

D	5	15	25	35	45	55	65	75	85	degrees
1μm	1.0	1.0	1.1	1.2	1.4	1.7	2.4	3.9	11.5	
3μm	3.0	3.1	3.3	3.7	4.2	5.2	7.1	11.6	34.4	
5μm	5.0	5.2	5.5	6.1	7.1	8.7	11.8	19.3	57.4	
7μm	7.0	7.2	7.7	8.5	9.9	12.2	16.6	27.0	80.3	
9μm	9.0	9.3	9.9	11.0	12.7	15.7	21.3	34.8	103.3	
11μm	11.0	11.4	12.1	13.4	15.6	19.2	26.0	42.5	126.2	
13μm	13.0	13.5	14.3	15.9	18.4	22.7	30.8	50.2	149.2	
15μm	15.1	15.5	16.6	18.3	21.2	26.2	35.5	58.0	172.1	
17μm	17.1	17.6	18.8	20.8	24.0	29.6	40.2	65.7	195.1	
19μm	19.1	19.7	21.0	23.2	26.9	33.1	45.0	73.4	218.0	
21μm	21.1	21.7	23.2	25.6	29.7	36.6	49.7	81.1	240.9	

By determining the eccentricity of the feature's outline, the intersecting angle is resolved. Further, calculating the major axis of the ellipse gives the segment direction that, although ambiguous, is easily corrected. Figs. 10a and 10b show the ambiguity that arises using one oblique intersection. Fig. 10c is the orthogonal view of the intersection.

The ambiguity of the cylinder-plane intersection is clarified by examining where in the sectioning plane the previous intersection is located. That position, once a trajectory has been calculated, determines which intersection scenario is most likely.

Because neural processes are not cylindrical in general, using projected feature outlines to determine segment directions is unreliable. Natural shape irregularities as well as dendritic spines may foil this method. However, two situations arise that benefit from this


(a) Upward trajectory

(b) Downward trajectory

(c) Orthogonal view

Fig. 10. Ambiguous plane-segment intersections.

method. The first is when starting the reconstruction process, before previous intersections are available to estimate directions. Inferring direction from intersection outline orientation provides two starting possibilities. The second situation occurs when a contour outline has no neighboring, intersecting outline. (The segment intersection of section n intersects no contours of section $n + 1$.) Determining segment trajectory from outline orientation in section n gives a position in section $n + 1$.

E Efficiency Concerns

Early reconstruction methods performed a brute force check against all ROI pairings between the sections to find all contour intersections. Exhaustive searching finds all joins and splits, but is inefficient, since for n ROIs in each slice this approach has time complexity $O(n^2)$. This is tolerable for very small numbers of ROIs, but reconstruction of rat brain tissue, with section resolutions of 30,000 by 60,000 pixels (representing 1cm x 2cm sections), can contain more than 1,000,000 ROIs. The brute force searching method is infeasible.

An improved approach is to take advantage of the contours' Y ordering. During image segmentation each ROI is created and saved in increasing Y order. Applying binary search allows a reconstruction algorithm to test against contours most likely to match. This reduces the time complexity for the number of contour intersection tests to $O(n \log n)$ where n is the number of contours. However, the actual intersection test imposes large constants on the runtime, which is dependent on the number of points in each contour. Total complexity can be expressed as $O(NM \ n \log n)$, where N and M are the number of points in the two contours being tested. The test data sets, while small, have large constants for this complexity.

An even better approach to finding intersections is to perform all contour intersection tests at once. Each feature contour, represented by a convex polygon, can be deconstructed into its constituent line segments. Finding contour intersections now becomes a problem of finding line segment intersections. For each contour c, attach a label to its line segments to identify each segment with c. A plane-sweep algorithm [31] is run over the line segments. The resulting list of line intersections is translated back to contour intersections by examining each intersecting pair of line segments – those that have different labels indicate two contours intersected. Calculating all line segment intersections can be expected to take time $O(n(N + M) \log n(N + M))$. For data sets containing about 200 identifiable contours per section, this translates to about 11% of the previous improvement's work, and roughly 1% of the original brute force technique.

F Section Skipping

Testing for segment paths is further accelerated by skipping the tests for adjacent sections based on the current segment diameter and section intersection angle. This reduces the adjacent section dependencies and thus the need to analyze every section. Given a segment-plane intersection angle (β), the section diameter (D), and section thickness (t), the number of skippable sections (SS) is determined by:

$$\mathrm{SS} = \frac{\mathrm{D}}{\mathrm{t}\sin\beta}$$

Table 4 contains the number of sections to skip, based on segment diameter and intersection angle. The numbers are based on the pipe model with 1μm sectioning, and for actual data more conservative values should be used. In general, as the segment becomes more perpendicular to the plane, more sections can be skipped. Reconstruction algorithms use these values to reduce the number of tests performed per section pair. Segments with large diameters are tested sporadically while smaller-diameter segments are tested more often.

TABLE 4

Number of Sections to Skip Given Segment Diameter D and Angle

D	5	15	25	35	45	55	65	75	85	degrees
1μm	11	3	2	1	1	1	1	1	1	
3μm	34	11	7	5	4	3	3	3	3	
5μm	57	19	11	8	7	6	5	5	5	
7μm	80	27	16	12	9	8	7	7	7	
9μm	103	34	21	15	12	10	9	9	9	
11μm	126	42	26	19	15	13	12	11	11	
13μm	149	50	30	22	18	15	14	13	13	
15μm	172	57	35	26	21	18	16	15	15	
17μm	195	65	40	29	24	20	18	17	17	
19μm	218	73	44	33	26	23	20	19	19	
21μm	240	81	49	36	29	25	23	21	21	

CHAPTER IX

STRUCTURAL REPRESENTATION OF NEURONS AND FIBERS

The branching structure of neuron morphology hints at a binary tree representation, a representation that has been used in other computer-aided reconstruction and growth modeling systems [32, 33]. For an automated system, a better data representation is required to facilitate connectivity tests and post-processing. This chapter introduces the Rete diagram and the string representation of the reconstructed neuronal fibers.

The *Rete diagram* is analogous to the Feynman diagram of particle physics, in that it shows, from a segment's initial detection, the path and intersections the segment possesses through the volumetric data set. The Rete diagram is not definitive of a fiber's structure, it only records per-segment pathways and leaves higher level structure determination to disambiguation techniques, discussed in the next chapter.

After considering various link-based structures, a simple 2D square bitmap was determined sufficient to record segment pathways, and is referred to as the "connection matrix." After image segmentation, the Recon system has recorded the number of Contours (N) present in the entire data set across all sections, and creates a connection matrix of that dimension (N x N). Each element is initially zero.

As segment reconstruction progresses, element (i, j) of the connection matrix is set to 1 when a path extends from contour i to contour j. This is all the information explicitly stored in the connection matrix.

The implementation as an C++ object allows simple optimizations to be added to prevent O(n^2) lookup times. Each row and each column has a counter to record the number of entries in the row or column. This simple addition ensures disambiguation checks can occur in constant time. As for storage space, this matrix is inherently sparse (and upper-triangular) and benefits from an appropriate internal representation. However, the development version uses a byte array to store the N^2 bits to mark each connection. This simple representation has proved adequate during testing and will continue to be used.

CHAPTER X

JUNCTION AND BEND DETECTION

In the previous chapters, assigning contours serial numbers was discussed, as was the connection matrix to record inter-contour connection information. This chapter introduces methods for detecting branches, joins, and bends using the connection matrix, as well as creating a string representation of the connection information.

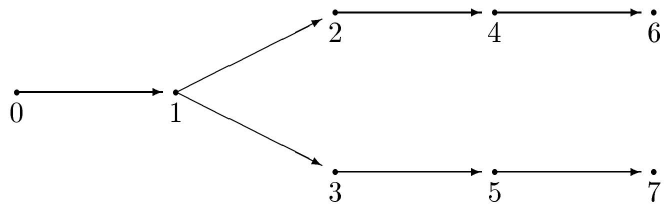
A Junction Classification

Earlier stages of the reconstruction followed segment paths through contours by predictive methods and as the connections were identified, the corresponding connection matrix entries were set. There were no semantic checks during this construction phase.

In figure 11a, a neuron process is traced from contour 0 to contour 1, where it bifurcates. The two new processes go through contours 2, 4, and 6, and 3, 5, and 7. The connection matrix for these contours is in Fig. 11b. Where the bifurcation occurs, at contour 1, the corresponding row contains two entries. Thus, bifurcations can be identified by searching the connection matrix for rows with multiple entries.

Fiber merges can similarly be detected. In Fig. 12a, two fibers join at contour 4, then continue as a single path through contours 5 and 6. The connection matrix for this junction in Fig. 12b represents the junction with multiple entries in column 4. All joins can be identified by searching the matrix for columns with multiple entries.

Planar sectioning of neural tissue will uncover "bends" in the fibers, where it appears as if the traced fiber may turn around inside a section. This situation can also be detected



(a) Interconnections of a bifurcation

	0	1	2	3	4	5	6	7
0	0	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

(b) Connection matrix for bifurcation

Fig. 11. Representation of a bifurcation.

with the connection matrix. As an example, the two fibers in Fig. 13a form a bend at contour 6. The corresponding connection matrix in Fig. 13b has two entries in column 6 for rows 4 and 5. At first sight this situation is no different from the join above, where a column has multiple entries. However, the identifying test is checking whether the fiber continues from contour 6 (or contour 4 in the prior join example). The test is simply seeing if row 6 has any entries; if not, classify the junction as a bend, else a join if the fiber continues.

The semantic tests of the connection matrix can be summarized thus:

- Join: if a column i has multiple entries, and row i has one entry, then a join occurs at contour i.

- Bend: if a column i has multiple entries but row i is empty, then a bend occurs at contour i.

$$\begin{tikzcd}
0 \arrow[r] & 2 \arrow[dr] & & & \\
& & 4 \arrow[r] & 5 \arrow[r] & 6 \\
1 \arrow[r] & 3 \arrow[ur] & & &
\end{tikzcd}$$

(a)Interconnections of a join (merge).

$$\begin{array}{c|ccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

(b) Connection matrix for a join

Fig. 12. Representation of a join.

- Bifurcation: if a row i has multiple entries, then a bifurcation occurs at contour i.

Other situations can also be defined. Bends occurring in the opposite direction (i.e., similar to a bifurcation) are detected by an empty column check, similar to the empty row check. Ambiguous situations, such as junctions occurring near the data set boundaries where continuation is impossible to check, can be defined as a bend or a junction.

The connection matrix will be extremely sparse when used on real data. The simple matrix implementation satisfies test cases, but for actual tissue sections a more appropriate data structure is required. Graph representations are a natural way to record the interconnections, and column and row checks equate to in-degree and out-degree comparisons of graph vertices.

$$\begin{tikzcd}
0 \arrow[r] & 2 \arrow[r] & 4 \arrow[dr, bend left] & \\
& & & 6 \\
1 \arrow[r] & 3 \arrow[r] & 5 \arrow[ur, bend right] &
\end{tikzcd}$$

(a) Interconnections of a bend

	0	1	2	3	4	5	6
0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	1	0
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0

(b) Connection matrix for a bend

Fig. 13. Representation of a bend.

B Models of Dendritic Bifurcation

One enhancement to the determination of bifurcations is the use of a model to locate (or reject) candidate ROIs. Consider the bifurcation in Fig. 14, where the "parent" fiber (with diameter D_p) bifurcates into "daughter" fibers (with diameters of D_{d_1} and D_{d_2}).

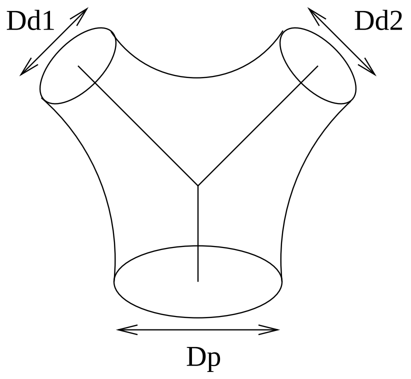


Fig. 14. Dendritic bifurcation with "daughter" and "parent" diameters.

Introduce two parameters $\alpha, \beta \in [0,1]$ such that

$$D_{d_1} = \alpha D_p, D_{d_2} = \beta D_p.$$

For the mutual dependence of α, β, we use the two limiting models described below. The first is Rall's ratio, a ratio determined from passive electrical models of dendrite segments to be about 1 [34]. The second is based on the projected area of segments where the ratio of parent segment area and the sum of daughter segments is also one (constant area before and after the bifurcation).

1. Rall factor conserving:

$$\mathrm{RR} = \frac{(\mathrm{D_{d_1}})^{3/2} + (\mathrm{D_{d_2}})^{3/2}}{\mathrm{D_p}^{3/2}} = 1$$

$$\alpha^{3/2} + \beta^{3/2} = 1$$

2. Projected area conserving:

$$\frac{(D_{d_1})^2 + (D_{d_2})^2}{D_p^2} = 1$$

or,

$$\alpha^2 + \beta^2 = 1$$

We introduce a fill ratio (FR) to represent the ratio of filled area (stained tissue) after bifurcation to the filled area before:

$$\mathrm{FR} \approx \frac{\mathrm{D^2_{d_1}} + \mathrm{D^2_{d_2}}}{\mathrm{D^2_p}} = \alpha^2 + \beta^2$$

For model 2 (area preserving)

$$\mathrm{FR} = 1.$$

For model 1 (Rall's model)

$$\text{FR} = \alpha^2 + \beta^2$$

subject to

$$\alpha^{3/2} + \beta^{3/2} = 1 \quad \alpha, \beta \in [0,1].$$

$$\beta^2 = (1 - \alpha^{3/2})^{4/3}$$

and substituting,

$$\text{FR} = \alpha^2 + (1 - \alpha^{3/2})^{4/3}$$

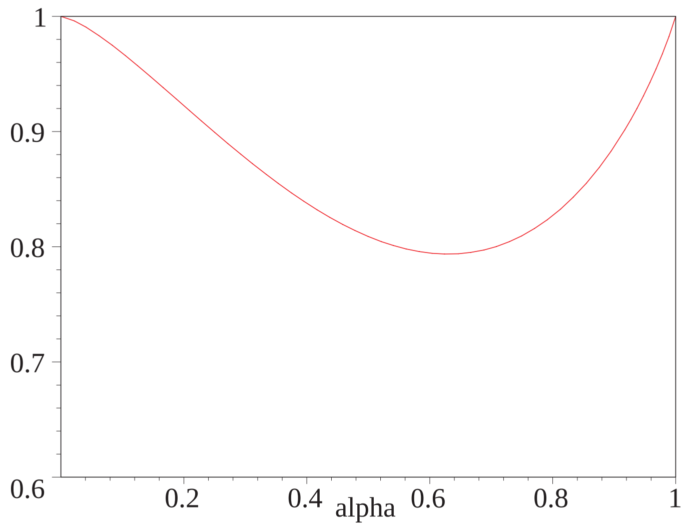


Fig. 15. Plot of Fill Ratio vs. α.

To find the fill factor, that fraction of the unit area that contains stained tissue, we need to combine the fill ratio with the observation that bifurcating dendrites are growing aymptotically more parallel to the cortical surface and here their area is elongated accordingly.

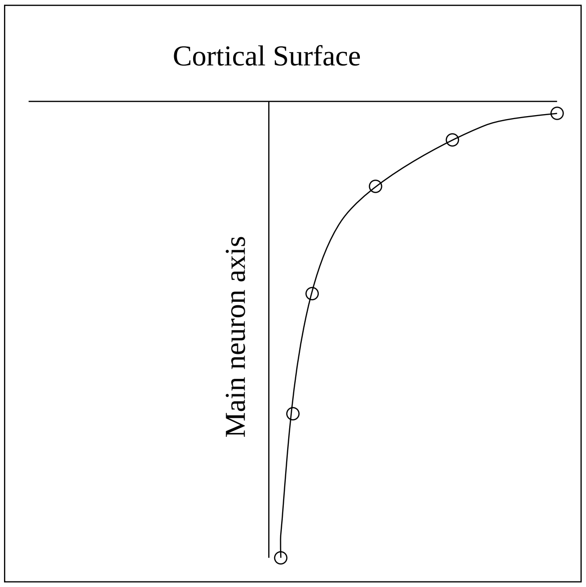


Fig. 16. Asymptotic growth of an apical dendrite w.r.t. cortical surface.

So after each bifurcation,

$$\mathrm{FR} = \mathrm{FF} \cos\theta .$$

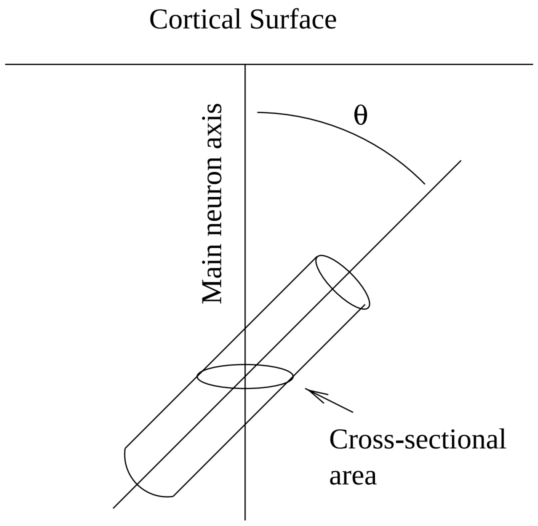


Fig. 17. Fiber sectional area is proportional to angle θ.

CHAPTER XI

RESULTS

Results gathered from running the Recon System on generated data are presented in this chapter.

A Reconstruction of Simulated Data

Test data generated by the Datagen program was reconstructed to find out which reconstruction algorithms worked, what the accuracy was, and how long each took. Results are presented only for the braid and helix data sets.

A.1 Preparation

Data sets consisted of 100 section images ($1\mu m$ thick) for both the braid and helix data types. Image resolutions varied from 512 x 512 to 1024 x 1024 pixels (256K and 1MB in size, respectively). Segment diameters were a constant $15\mu m$ thick. The braid data type is fixed at 3 segments, and successful reconstructions should only report this many. On the other hand, the helix type is more flexible and renders from 1 to 22 segments as needed.

The minimum distance finding, contour intersection, and path prediction algorithms were tested against the same data sets. During testing, it was observed that the minimum distance and contour intersection algorithms are inherently very similar. For the two test data sets, there was no difference in performance between these two methods, and so no results from the minimum distance test are presented.

All tests were performed on the following hardware configuration: 400MHz Intel Pentium II processor, Intel 440BX motherboard with 128MB RAM, 4.3GB EIDE disk. The system was running Redhat Linux v. 5.1 with the X Window System.

A.2 General Results

The first test compared the intersection and path prediction methods against the braid data set. Two views of the braid data set are shown in Fig. 18. Path prediction was run with 3 and 4 points for extrapolation. The results along with execution times (in seconds) are in Table 5. The "skip" number is the number of sections skipped during reconstruction, and "segs" is the number of segments reported reconstructed (3 is correct). What this shows in the contour intersection case is that for this data set, using every fifth section caused this algorithm to miss contours. The path prediction breaks down a bit earlier; when 4 points are used for extrapolation this method begins missing contours when using every fourth section. When 3 points are used the method lasts until every fifth section, then mispredicts, giving worse results than contour intersection.

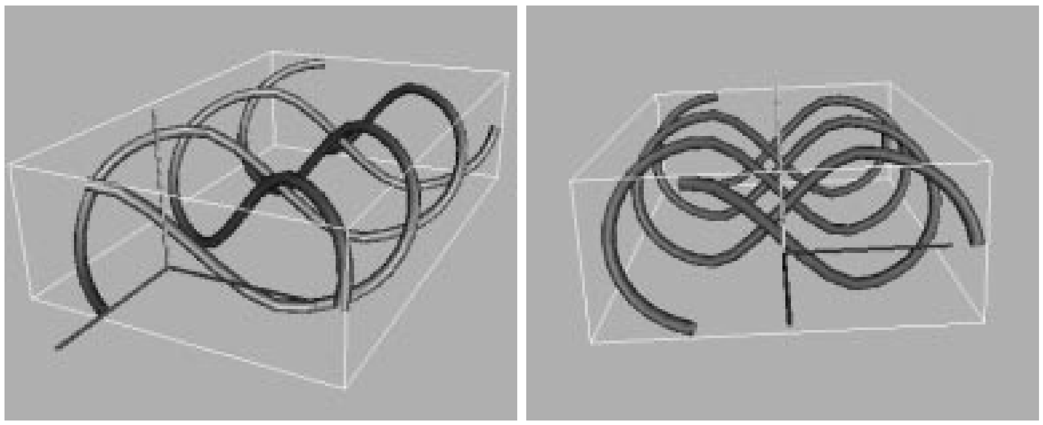


Fig. 18. Two 3-D views of the braid structure.

A.3 Compression Ratios

The reconstruction algorithms do not affect the image segmentation and ROI generation, so data compression rates do not vary with the reconstruction algorithm. Effective

TABLE 5

Results of the Braid Test for Intersection and Predictive Algorithms

	Skip=1		Skip=2		Skip=3		Skip=4		Skip=5	
1024^2	Segs	time	Segs	time	Segs	time	Segs	time	Segs	time
intersection	3	19.9	3	10.4	3	7.02	3	5.18	6	4.26
predict N=4	3	19.4	3	10.2	3	6.60	5	5.20	8	4.09
predict N=3	3		3		3		3		8	

TABLE 6

Braid Compression Ratios for 100 Sections at 1024 x 1024

	Original	Uncompressed ROI	Compressed ROI
size:	100MB	1212KB	616KB
ratio(:1):	1	82.5	162

compression ratios for the braid data set are listed in Table 6. The high ratio of 162:1 is representative of only having three segments to reconstruct – the vast majority of the image is discardable.

An example of the helix data set with 10 segments of 15μm diameter is shown in Fig. 19. The helix data set, with an adjustable number of segments, has considerably more variance in compression ratios, as listed in Table 7. As the number of segments increases, the amount of discardable image data decreases, and the effective ratio drops. The values in this table were created by the Recon System based on 15μm diameter segments.

For real tissue data, sections are expected to contain 200 to 600 maximum segment intersections of 3μm diameter. Using the test cases as a basis for extrapolation, the expected compressed data size is 1500KB with an effective compression ratio of 69:1. At the maximum 600 segments of 3μm diameter, the compressed size will be approximately 4500KB with a

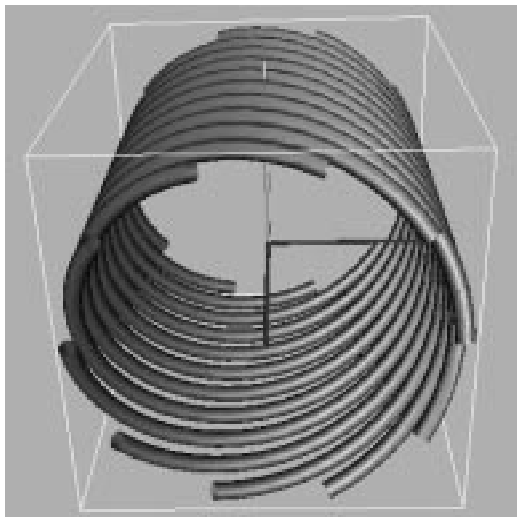


Fig. 19. Helix data set with 10 segments.

TABLE 7

Helix Compression Ratios for 100 Sections at 1024 x 1024

Number of	Uncompressed		Compressed	
Segments	Size(MB)	Ratio (:1)	Size(KB)	Ratio (:1)
2	0.46	215	31.70	3230
6	1.39	71	95.40	1073
10	2.32	43	159.10	643
14	3.25	30	223.10	458
18	4.18	23	287.20	356
22	5.11	19	351.30	291
200			1500.00	69
600			4500.00	22

ratio of 22:1. These values are based on the following expression:

$$\frac{\mathrm{N} = (200, 600)\mathrm{segments}}{\mathrm{section}} \times 100 \text{ sections} \times 76 \text{ bytes} \times \frac{1 \text{ KB}}{1024 \text{ bytes}}.$$

The factor of 76 bytes is the sum of the 21-byte ROI descriptor and the 55-byte compressed ROI image for a $3\mu\mathrm{m}^2$ region.

A.4 Execution Times

Table 5 listed the execution times (in seconds) of the two algorithms for the 3-segment braid. These times are for the full reconstruction process, from reading the images to tracing fibers, and decrease linearly as the number of skipped frames increases. There is no appreciable difference in execution times for reconstructing three segments. However, the reconstruction methods' execution times differ dramatically with an increasing number of segments per section, as plotted in Fig. 20. The high growth rate of the contour intersection algorithm is indicative of excessive testing; reducing the number of contour intersection tests by more sophisticated algorithms would shorten execution times.

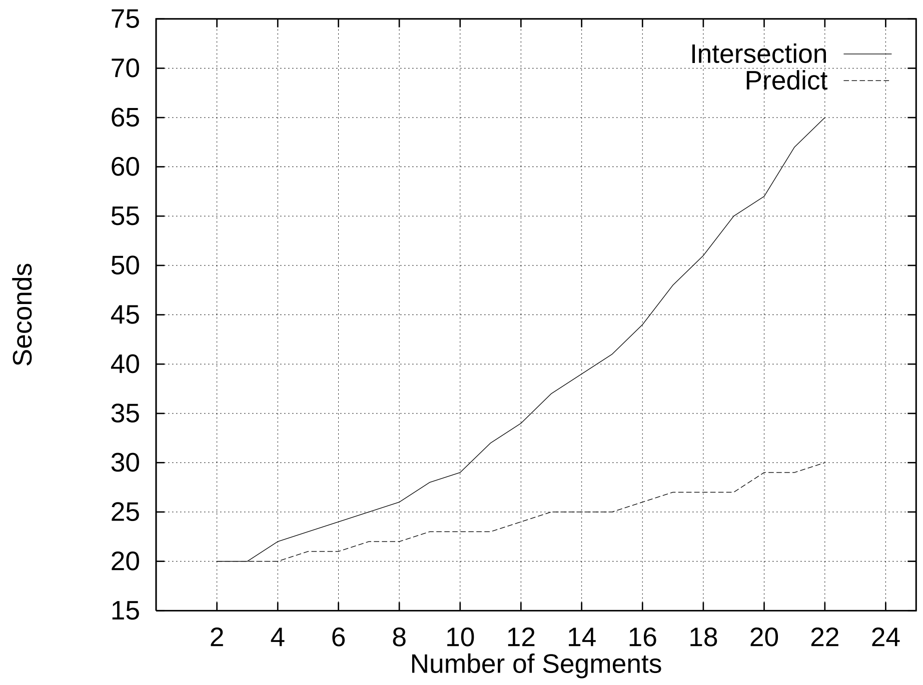
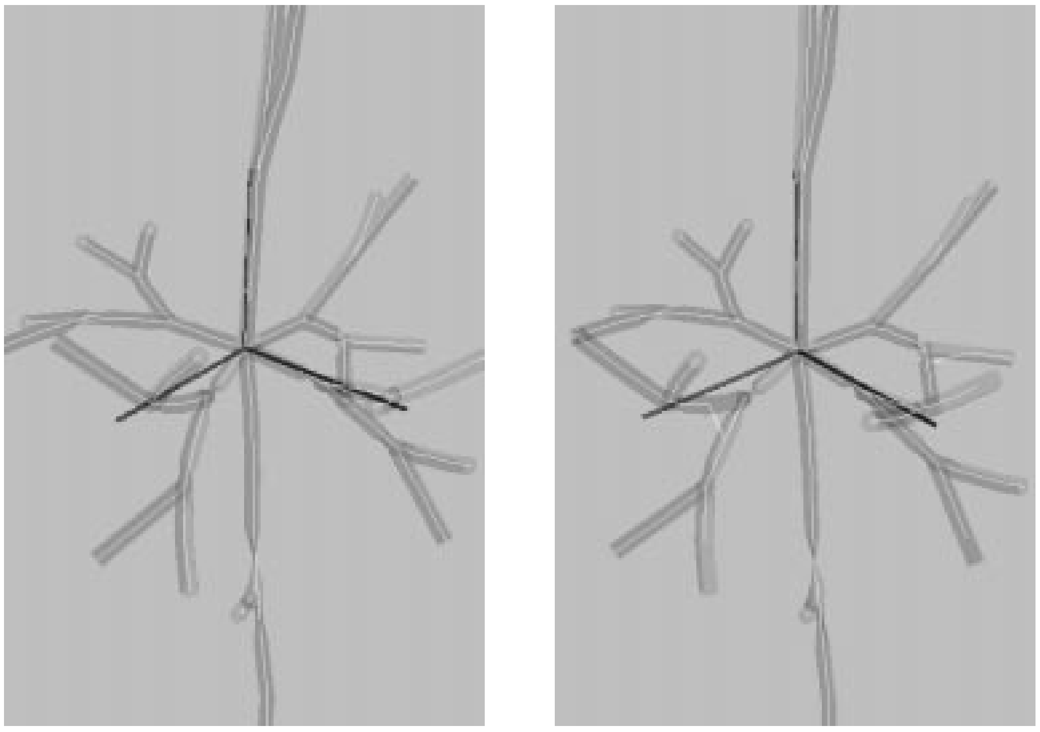


Fig. 20. Execution times w.r.t. the number of segments

B Overview of Visualization Procedure

Tools based on the Visualization Toolkit were written to visualize the differences between neuron geometry files created by `Datagen` and those created by the Recon System. Reconstruction accuracy is determined manually by visual inspection and to this end a simple method of rendering the neurons is used to enhance the differences. In a VTK pipeline the reconstructed neuron is rendered as a wireframe model first, then the original neuron geometry is rendered with transparent cylinders by setting opacity to 10%. This creates a skeleton and skin model. Two comparisons are shown in Fig. 21. On the left is a correct reconstruction, where the reconstructed skeleton extends fully through the cylindrical model and branches correctly. On the right is a topologically correct reconstruction, but branch point locations and segment endpoints are incorrect.

After reconstruction, neuron models can be placed into a virtual environment to visualize the dense neuronal forests of the cortex and the morphology of neighboring cells [35].



(a) Identical neurons (b) Minor problems

Fig. 21. The skeleton-skin visualization technique with axes.

CHAPTER XII

CONCLUSIONS

An automated system for parallel neuron tracing and reconstruction has been implemented and shown to be feasible. Aiding the implementation is a clear separation of the data acquisition tasks and reconstruction tasks through an object-oriented design. The limitations of the reconstruction algorithms were explicitly tested using model data. These techniques help create a system that is adaptable, easily modified, and robust.

A Parallel Tracing and Reconstruction of Dendritic Processes and Fibers

The Recon System is capable of following segment paths, recording the 3-D positions, and performing structure-based reconstruction. Path following is performed in parallel, stepping each neuron segment through the sections, and visiting each section only once.

B Data Compression for Maximal Scan Rate

The required effective data compression rate of 20:1 was surpassed by using data culling and data compression. Since the ROIs extracted from each image are used in reconstruction efforts, the background areas of the section images is discarded. For estimated worst cases of 600 3μm contours per 0.1mm^2, the compression ratio is 22:1, and for fewer contours the ratio is greater. Combining data culling and data compression techniques provide quite pleasing results.

C Automating the Three-dimensional Reconstruction

The Recon System as implemented is run from the computer's command line with a list of files to process, along with a command line argument to select the reconstruction algorithm. The Recon System loads the images and performs all reconstruction tasks without intervention, writing its results to a log file.

D Future Work

The Recon System implemented here has limitations. The reconstruction algorithms (brute force or predictive) perform O(NM) tests per-section, where N and M are the number of contours in two neighboring slices. This growing execution time was graphed in Fig. 20. The fact that contours are created and stored in sorted Y order can be leveraged to fight this growth. Intersection or contour "hit" testing based on binary search of the contours would then take $O(N\log M)$.

Boundary effects come into play when statistics are gathered on reconstructed neurons. Future development should consider how to manage isolated segment parts present in one volumetric data set that belong to neurons in a neighboring volume. This problem also includes matching of segment data between volumetric blocks, and basic inter-data set registration.

Reconstruction can benefit from the statistical analysis present in the N++ system, perhaps using collected data to guide the predictive reconstruction methods [36].

A parametric string representation was considered to describe the reconstructed segments and neurons. A more compact alternative is to use piecewise cubic splines to represent neuronal segments. Dierckx describes these splines in [37].

Past branched structure work was applied to coronary arteries (and other vessels) and other similar tissue. The Recon System was implemented for neuronal fiber reconstruction. Applying the system to other reconstruction domains will require specific predictive reconstruction and disambiguation methods, as well as adding scale-awareness to the system. Image processing appropriate to other domains is also required.

Finally, porting the Recon System to other platforms may benefit from another language, such as Java. The current system was designed with this in mind to ease the translation.

REFERENCES

[1] J. J. Capowski and M. J. Sedivec, “Accurate Computer Reconstruction and Graphics Display of Complex Neurons Utilizing State-of-the-art Interactive Techniques,” *Comp. Biomed. Res.*, vol. 14, pp. 512–532, 1981.

[2] J.-D. Boissonnat and B. Geiger, “Three Dimensional Reconstruction of Complex Shapes Based on the Delauney Triangulation,” Tech. Rep. 1697, INRIA Rapports de Recherche-Sophia Antipolis, 1992.

[3] B. H. McCormick, “Design of a Brain Tissue Scanner,” *Neurocomputing*, in press 1999.

[4] P. V. Belichenko and A. Dahlström, “Confocal Laser Scanning Microscopy and 3D Reconstruction of Neuronal Structures in Human Brain Cortex,” *NeuroImage*, vol. 2, pp. 201–7, 1995.

[5] A. W. Toga, ed., *Three-Dimensional Neuroimaging.* New York, NY: Raven Press, 1990.

[6] P. D. Coleman, C. F. Garvey, J. H. Young, and W. Simon, “Semiautomatic Tracking of Neuronal Processes,” in *Computer Analysis of Neuronal Structures* (R. D. Lindsay, ed.), ch. 5, pp. 91–109, New York, NY: Plenum Press, 1977.

[7] E. W. Stockley, H. M. Cole, A. D. Brown, and H. V. Wheal, “A System for Quantitative Morphological Measurement and Electrotonic Modelling of Neurons: Three-Dimensional Reconstruction,” *J. Neurosci. Methods*, vol. 47, pp. 39–51, 1993.

[8] V. A. Moss, D. M. Jenkinson, and H. Y. Elder, “Automated Image Segmentation and Serial Section Reconstruction in Microscopy,” *J. Microscopy*, vol. 158, pp. 187–196, May 1990.

[9] A. Odgaard, K. Andersen, F. Melsen, and H. J. G. Gundersen, “A Direct Method for Fast Three-Dimensional Serial Reconstruction,” *J. Microscopy*, vol. 159, pp. 335–342, Sept. 1990.

[10] C. Levinthal and R. Ware, “Three Dimensional Reconstruction from Serial Sections,” *Nature*, vol. 236, pp. 207–211, Mar. 1972.

[11] P. Rakic, L. J. Stensas, E. P. Sayre, and R. L. Sidman, “Computer-Aided Three-Dimensional Reconstruction and Quantitative Analysis of Cells from Serial Electron Microscopic Montages of Foetal Monkey Brain,” *Nature*, vol. 250, pp. 31–34, July 1974.

[12] C. Barillot, B. Gibaud, J. Scarabin, and J. Coatreux, “3D Reconstruction of Cerebral Blood Vessels,” *IEEE Computer Graphics and Applications*, vol. 5, pp. 13–19, 1985.

[13] J. K. Udupa and D. Odhner, “Shell Rendering,” *IEEE Computer Graphics and Applications*, vol. 13, no. 4, pp. 58–67, 1993.

[14] W. E. Lorensen and H. E. Cline, “Marching Cubes: A High Resolution 3-D Surface Construction Algorithm,” *Computer Graphics*, vol. 21, pp. 163–169, 1987.

[15] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit.* Upper Saddle River, NJ: Prentice Hall, second ed., 1998.

[16] C. R. Gerfen and P. E. Sawchenko, “An Anterograde Neuroanatomical Tracing Method that Shows the Detailed Morphology of Neurons, Their Axons and Terminals: Immunohistochemical Localization of an Axonally Transported Plant Lectin, *Phaseolus vulgaris* Leucoagglutinin (PHA-L),” *Brain Research*, vol. 290, pp. 219–238, 1983.

[17] K. Montgomery, "Automated Reconstruction of Neural Elements from Transmission Electron Microscope Images," PhD thesis, University of California, Santa Cruz, 1996.

[18] K. Mulchandani, "Morphological Modeling of Neurons," Master's thesis, Texas A&M University, College Station, TX, 1995.

[19] J. Leech, "The Lsys Software Program," Available from `ftp://ftp.cs.unc.edu-/pub/users/leech`, May 1993.

[20] M. Woo, J. Neider, and T. Davis, *OpenGL Programming Guide.* Reading, MA: Addison-Wesley Developers Press, 1997.

[21] J. Poskanzer, "NETPBM Portable Digital Image Software," Available from `ftp://wuarchive.wustl.edu/graphics/packages/NetPBM`, Dec. 1993.

[22] M. Marín-Padilla, "The Pyramidal Cell and its Local-Circuit Interneurons: A Hypothetical Unit of the Mammalian Cerebral Cortex," *J. Cog. Neurosci.*, vol. 2, no. 3, pp. 180–194, 1991.

[23] J. F. Pasternak and T. A. Woolsey, "On The "Selectivity" of the Golgi-Cox Method," *J. Comp. Neurology*, vol. 160, pp. 307–312, Apr. 1975.

[24] T. Wilson, *Confocal Microscopy.* San Diego, CA: Academic Press Ltd., 1990.

[25] K. Carlsson and A. Liljeborg, "A Confocal Laser Microscope Scanner for Digital Recording of Optical Serial Sections," *J. Microscopy*, vol. 153, pp. 171–180, Feb. 1989.

[26] C. Pellot, A. Herment, M. Sigelle, P. Horain, and P. Peronneau, "Segmentation, Modelling and Reconstruction of Arterial Bifurcations in Digital Angiograhy," *Med. & Biol. Eng. & Comput.*, vol. 30, pp. 576–583, 1992.

[27] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* MIT Press, 1994.

[28] S. Gottschalk, M. Lin, and D. Manocha, "Obb-Tree: A Hierarchical Structure for Rapid Interference Detection," in *Proc. ACM Siggraph '96*, pp. 171–180, 1996.

[29] J. Gailly and M. Adler, "The zlib General Purpose Compression Library," Available from `ftp://prep.ai.mit.edu/pub/gnu`, 1998.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, second ed., 1992.

[31] M. deBerg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications.* Berlin: Springer, 1997.

[32] G. Coppini, M. Demi, R. Mennini, and G. Valli, "Three-Dimensional Knowledge Driven Reconstruction of Coronary Trees," *Med. & Biol. Eng. & Comput.*, vol. 29, pp. 535–542, 1991.

[33] J. van Pelt, R. Verwer, and H. Uylings, "Application of Growth Models to the Topology of Neuronal Branching Patterns," *J. Neurosci. Methods*, vol. 18, pp. 153–165, 1986.

[34] W. Rall, "Core Conductor Theory and Cable Properties of Neurons," in *The Nervous System: Cell Biology of Neurons* (E. Kandel, ed.), vol. 1, pp. 39–97, Bethesda, MD: American Physiological Society, 1977.

[35] B. P. Burton, T. S. Chow, A. T. Duchowski, W. Koh, and B. H. McCormick, "Exploring the Brain Forest," *Neurocomputing*, in press 1999.

[36] R. W. DeVaul and B. H. McCormick, "Neuron Developmental Modeling and Structural Representation: The Statistical Model," tech. rep., Scientific Visualization Laboratory, Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, July 1997.

[37] P. Dierckx, *Curve and Surface Fitting with Splines.* Oxford Press, 1993.

VITA

Brent P. Burton received his B.S. degree in Computer Science from Texas A&M University in August 1992. After working as a UNIX system administrator for the Institute for Scientific Computation and the Department of Mathematics (Oct. 1992 – Apr. 1996), he joined the Scientific Visualization Laboratory as a Graduate Assistant Research (Apr. 1996 – Jan. 1998).

Brent is employed by 3dfx in Austin, TX, and can be reached at the following address:

2901 Barton Skyway Apt. 1305

Austin, TX 78746-7555

Email: `bburton@3dfx.com`