STRATEGY FOR CONSTRUCTION OF

POLYMERIZED VOLUME DATA SETS

A Thesis

by

PRATHYUSHA ARAGONDA

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

December 2004

Major Subject: Computer Science

STRATEGY FOR CONSTRUCTION OF

POLYMERIZED VOLUME DATA SETS

A Thesis

by

PRATHYUSHA ARAGONDA

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

MASTER OF SCIENCE

Approved as to style and content by:

Bruce H. McCormick (Chair of Committee)	John Keyser (Member)
Donald H. House (Member)	Valerie E. Taylor (Head of Department)

December 2004

Major Subject: Computer Science

ABSTRACT

Strategy for Construction of Polymerized Volume Data Sets. (December 2004)

Prathyusha Aragonda, B.Tech., Sri Venkateswara University

Chair of Advisory Committee: Dr. Bruce H. McCormick

This thesis develops a strategy for polymerized volume data set construction. Given a volume data set defined over a regular three-dimensional grid, a polymerized volume data set (PVDS) can be defined as follows: edges between adjacent vertices of the grid are labeled 1 (active) or 0 (inactive) to indicate the likelihood that an edge is contained in (or spans the boundary of) a common underlying object, adding information not in the original volume data set. This edge labeling "polymerizes" adjacent voxels (those sharing a common active edge) into connected components, facilitating segmentation of embedded objects in the volume data set. Polymerization of the volume data set also aids real-time data compression, geometric modeling of the embedded objects, and their visualization.

To construct a polymerized volume data set, an adjacency class within the grid system is selected. Edges belonging to this adjacency class are labeled as interior, exterior, or boundary edges using discriminant functions whose functional forms are derived for three local adjacency classes. The discriminant function parameter values are determined by supervised learning. Training sets are derived from an initial segmentation on a homogeneous sample of the volume data set, using an existing segmentation method.

The strategy of constructing polymerized volume data sets is initially tested on synthetic data sets which resemble neuronal volume data obtained by three-dimensional microscopy. The strategy is then illustrated on volume data sets of mouse brain microstructure at a neuronal level of detail. Visualization and validation of the resulting PVDS is shown in both cases.

Finally the procedures of polymerized volume data set construction are generalized to apply to any Bravais lattice over the regular 3D orthogonal grid. Further development of this latter topic is left to future work.

To my parents

ACKNOWLEDGMENTS

I would like to express my sincere and heartfelt gratitude to Dr. Bruce H. McCormick, for his great patience, encouragement and support throughout my research. This thesis was enabled by his vision and ideas. Working under his guidance past these two years helped me to learn a lot that will help in every phase of my career.

I would like to thank Dr. John Keyser for his support and valuable suggestions for this thesis. I would also like to thank Dr. Donald House for his useful suggestions.

I would like to express my great gratitude to my parents, whose love and support lies in every success I achieve. Without their constant encouragement, this journey of my education would have never been possible. I would also like to thank my husband, Motheeswar and my brothers, Pradeep and Kishore, for all their encouragement and support.

I would like to thank all my fellow students in the lab for their great help. In particular, I would like to thank Purna Doddapaneni, David Mayerich, Brad Busse, and Zeki Melek for their contribution to this work.

Special thanks to my wonderful roommate, Vidhya, and my friends, Shruti and Anshul for their tremendous help and support.

I would like to thank Texas Higher Education Coordinating Board (ATP Grant 000512-0146-2001) for supporting my thesis in part.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

Page

Page

Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

The reconstruction and visualization of objects within 3D volumes, as generated from a stack of sequential 2D images, has been an emerging and developing field for many years. This field is especially useful for medical imaging where reconstruction of human anatomy, tissue microcirculation, and as well as proteins and viruses is desired. The work presented in this thesis has been particularly motivated by our attempts to scan and reconstruct brain tissue at a neuronal level of detail. To achieve efficient 3D reconstruction of neuronal data, a new way of segmenting the volume data using Polymerized Volume Data Sets (PVDS) is introduced.

Given a volume data set defined over a regular three-dimensional orthogonal grid, we define a *polymerized volume data set (PVDS)* as follows: In addition to the value assigned to every vertex (voxel), selected edges between neighboring vertices of the grid are given a Boolean label. *Active* edges are assigned the value 1 and the remaining edges, labeled *inactive,* are assigned the value 0. Edge labeling provides information about the likelihood that two vertices sharing a common active edge are contained in the same underlying object. Only local information is used. Boolean labeling {0, 1}, as derived from discriminant functions on the values of neighboring voxels, is a crude estimate of this co-habitation of the edge termini in the same object. This polymerized volume data set aids in efficient compression and subsequent segmentation of the volume data set.

1.1. Goals of the thesis

This thesis develops a strategy for constructing polymerized volume data sets (PVDS). The thesis provides methods for finding Boolean labels for edges in a volume data set, constructing a PVDS, and subsequently using that polymerized volumetric data set to visualize objects within the three-dimensional volume.

This thesis follows the style and format of *IEEE Transactions on Visualization and Computer Graphics*.

The following goals serve as multiposts for a strategy that would construct polymerized volume data sets and provide a statistical basis for polymerization:

i. Establish the local adjacency classes for a volume data set considered as a regular three-dimensional grid.

ii. Formulate a set of discriminant functions (one for each adjacency class) that label active edges.

iii. Construct training sets using a base-line segmentation and train the weight vectors of edge-labeling discriminant functions using these training sets.

iv. Minimize the computational cost of edge-labeling.

v. Visualize the segmentation achieved by PVDS found using the trained discriminant functions.

vi. Develop criteria for evaluating polymerization-induced segmentation and find the degree of agreement/disagreement between this approach and the base-line segmentation.

vii. Test the PVDS construction on synthetic data.

viii. Demonstrate the PVDS construction on volume data sets of mouse brain microstructure at a neuronal level of detail.

1.2. Background and rationale for the polymerized volume data set (PVDS)

Usage of volumetric data in 3D medical imaging started in the1970s. Since then the field of volume visualization has developed and became a prominent area of research within the field of Computer Graphics. Kaufman describes volume visualization as a method of interpreting complex volumetric data. It provides for the analysis and understanding of scanned or experimental 3D volumetric data and for the synthesis of volumetric objects by a computer model [14].

The primary source of volume data is the set of 2D images generated by serial scanning successive sections of a 3D volume. These 2D images are processed to

reconstruct the 3D volume. Due to the high complexity and quantity of volumetric data, efficient ways are needed to process the volume data and to reconstruct and visualize 3D objects.

The work presented in this thesis has been particularly motivated by our attempts to scan and reconstruct brain tissue at a neuronal level of detail. These volume data sets acquired by three-dimensional microscopy have several distinguishing features. Among them are:

- The full volume data set can be extremely large. Raw data set sizes can reach multiple terabytes.
- The data of interest within the volume data sets (i.e., the stained neuronal tissue) tend to be sparse, taking up only a modest portion of the overall volume.
- The neurons to be modeled have filamentary, branching structures; alternatively, the neurons in a scanned volume can be limited to closely-packed cell bodies.
- Data is acquired at a high data rate (typically ~100Mvoxels/s), and quick ways are needed to compress and store the data in a geometrically meaningful way that facilitates future reconstruction.

The current volumetric representation techniques are found to be deficient in addressing at least one of these features [12, 19, 20]. Due to the potential data size, methods that keep the entire volume in memory at once are unrealistic. Several representational methods (such as the well-known octree) are poorly suited for modeling long, thin structures. Medial-axis methods, while good for representing neurites, tend to process too slowly and can require too much data to be stored in memory. Pure image and video compression techniques can work well for compression, but fail to give any meaningful insight into the geometric structure of the objects to be modeled.

To circumvent the above limitations, our group introduced a polymerization strategy for the compression, segmentation, and modeling of filamentary volume data sets [19, 20]. The goals of the polymerization are to:

- Allow data compression in real time, in such a manner as to facilitate subsequent segmentation of the volume data set;
- Provide a data compression and segmentation strategy which exploits the efficiencies of examining successive serial images of the volume data set, yet is independent of the axis chosen for serial sectioning; and
- Separate segmentation of embedded objects in the volume data set clearly from both their geometric modeling and their visualization.

The polymerization strategy starts by polymerizing the volume data set. This thesis sets forth these polymerization methods. As defined previously, a polymerized volume data set (PVDS) labels the edges in the regular three-dimensional grid as either active or inactive. In this thesis, discriminant functions are used to assign Boolean labels to edges in the volume data set. They define the segmentation process and determine whether the segmentation or reconstruction is faithful. These functions can be arbitrarily complex; but here to minimize computational complexity, they are limited to be quadratic in the gray scale values of neighboring voxels.

The data volume is created by serially scanning successive sections, typically perpendicular to the vertical Z-axis. Here voxels at the same (X, Y) position in two successive registered images can be conveniently labeled as likely to be drawn from the same underlying physical object by marking their common vertical edge as active (Fig. 1). However, as the specimen could have equally well have been sectioned perpendicular to the X- or Y- axes, the same edge labeling scheme is extended to all three axes [5]. The labeling procedure is then generalized to embrace other rotationally invariant choices of edges emanating from a vertex.

An important application of the PVDS is in forming three-dimensional L-blocks, a new data structure introduced for the representation of volumetric data [19, 20]. This data structure is introduced to efficiently deal with volume data where the full volume data set can be extremely large but the data of interest within it occupies only a small percentage of the volume. An L-block is defined as a 3-dimensional isorectangular block

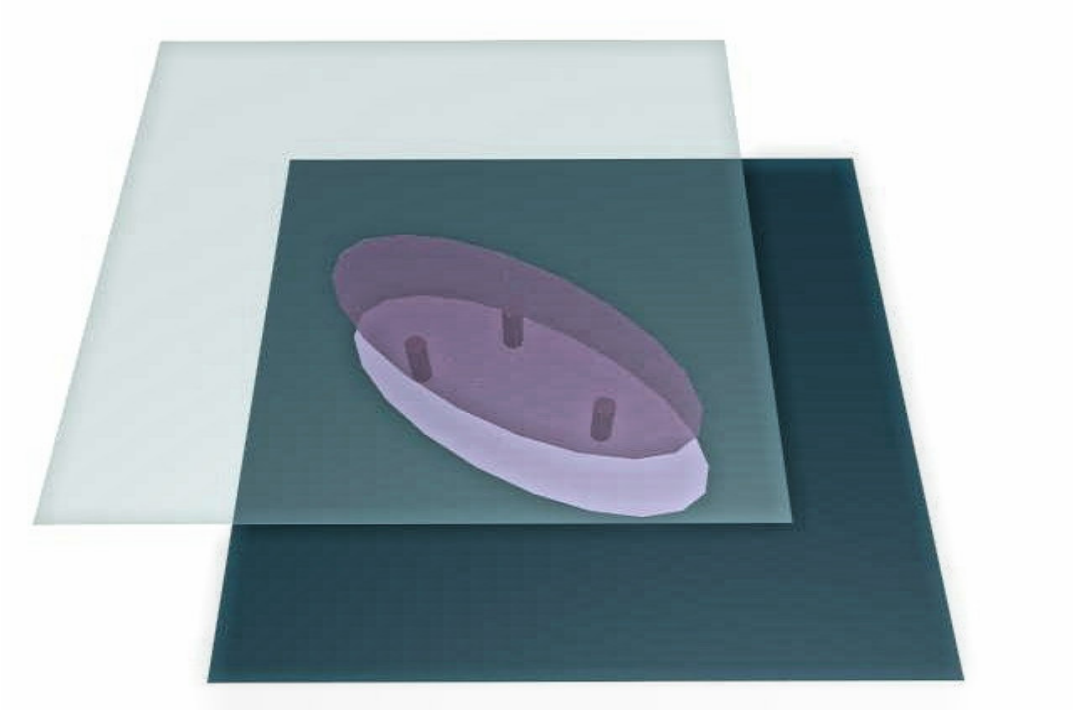


Fig. 1. Active vertical edges between successive sectional images denoting the same underlying physical object.

of enhanced vertex information. The block must be entirely contained within the uniform 3-dimensional grid of the PVDS. An (l_1, l_2, l_3) L-block refers to a block of l_1 vertices in the first dimension, l_2 in the second, etc., Each L-block is defined by: (1) its position, e.g. (i, j, k), of its least vertex, as indexed within the parent 3-dimensional uniform grid, (2) its template (l_1, l_2, l_3), and (3) its vertex array, containing the enhanced vertex information (e.g., voxel value(s) and edge labels).

As seen from the definition of L-blocks, the polymerized volume data set mainly supports the implementation of L-blocks. An initial implementation of L-block coverings was based on the simple method of using a threshold level for finding active edges (Fig. 2). Better methods of forming the PVDS increase the accuracy of labeling active edges and hence improve the correctness of 3D volume model reconstructed using L-blocks.

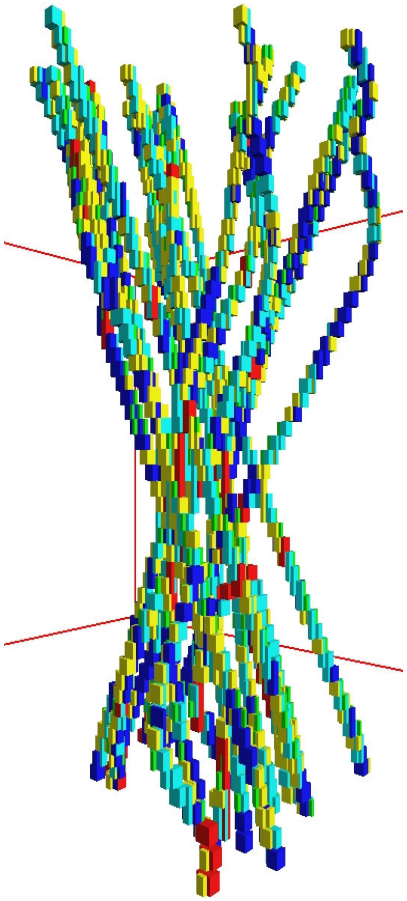


Fig. 2. Examples of L-block coverings [11].

1.3. Overview of the thesis

This thesis develops a strategy for polymerized volume data set construction. In Chapter II, the concept of a lattice defined over a regular orthogonal grid in 3D space is discussed. This helps in visualizing three types of adjacency in 3D that are 6, 8 and 12-adjacency classes. The data representation of PVDS is defined in this chapter.

In Chapter III, we derive the discriminant functions for edge-labeling, one for each of the 6, 8 and 12-adjacencies. Discriminant functions classify input vectors, drawn from neighboring gray-scale values associated with an edge, and assign each vector to a class C_k among n different classes C_1 to C_n.

In Chapter IV, we derive the discriminant functions for boundary edge-labeling. This is a special case that requires discriminant functions to be asymmetric in gray scale values of the edge termini unlike symmetric functions as in Chapter III. A boundary edge is further classified as either an ‘entering’ edge or an ‘exiting’ edge.

Chapter V discusses an automated algorithm for three-dimensional vector tracing of neurons initially presented in [1]. Construction of quasi-planar templates in a paddle-wheel configuration is presented. A variant of the tracing algorithm is achieved by using Frenet frames while tracing. This algorithm has been implemented to trace neurons from the stack of images obtained in our laboratory. These vector tracing algorithms are used in this thesis to construct training sets [6].

Chapter VI explains construction of training sets for the synthetic volume data generated to test the PVDS in ideal cases. The different steps involved in generating synthetic data, which includes creation of 3D models, sectioning into slices, correction for slice thickness, application of point spread function, and anti-aliasing filters are explained. Artificial filamentary data and block data are generated for this purpose.

Chapter VII explains the construction of training sets for neuronal data obtained from the knife-edge scanning [21] of mouse brain microstructure. A computationally-expensive base-line segmentation algorithm is used to segment a sample of the volume data. This initial segmentation is used to classify edges throughout the entire volume data set into one of the three classes: interior, exterior or boundary edges. Boundary edges are then further classified into entering and exiting edge classes.

Chapter VIII discusses the procedure for training the edge-labeling discriminant functions using the training data sets. Approaches to minimize the computational cost of edge-labeling are also discussed.

Chapter IX shows the visualization of polymerized volume data set and evaluates the polymerization-induced segmentation with respect to the base-line segmentation. A set of criteria is developed to compare both segmentation methods.

Chapter X generalizes the concept of a PVDS to Bravais lattices over the regular orthogonal grid in three-dimensional space.

Finally, Chapter XI presents conclusion and future work.

CHAPTER II

LATTICE OF THE CONNECTIVITY-ENHANCED GRID SYSTEM

2.1. Three-dimensional regular grid system

We assume that a *regular three-dimensional grid system* has been established in the volume to be scanned. A regular three-dimensional grid system can be constructed simply by setting values in a rectangular array of position vectors,

$\mathbf{r}[i\ j\ k]\quad (i = 0,1,...,I-1\quad j = 0,1,...,J-1\quad k = 0,1,...,K-1)$ and identifying the indices i, j, k with the three linear coordinates [23]. For simplicity in this thesis, we will normally bypass the linear mapping and speak of the vertex (voxel) at $(i\ j\ k)$.

2.2. Lattice of the connectivity-enhanced grid system

The grid system (Section 1 above) can be extended into a three-dimensional lattice by adding directed edges at every vertex (Fig. 3). The number of edges emanating from a vertex is constant and the vertices reached are referred to as its adjacency set [15]. Directed edges are aligned into orthogonal rays throughout the grid system. Rays are assigned a direction common to all parallel rays. Edges sharing a common vertex and ray are identified as "incoming" and "outgoing", respectively.

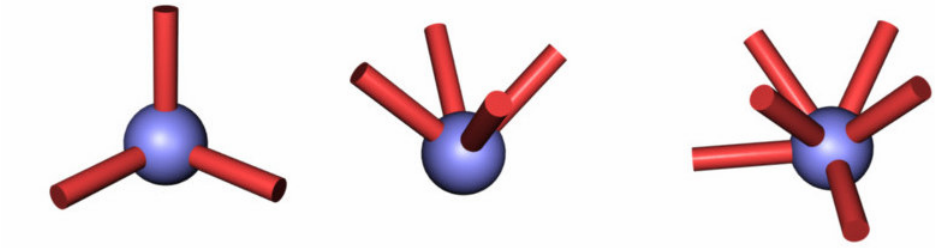


Fig. 3. Generic vertices with associated edges in 6-adjacency, 8-adjacency and 12-adjacency lattices of the grid system. The directed edges represent potentially polymerizable links to neighboring vertices (voxels). (From Melek based on [19]).

A vertex in the PVDS lattice with 6-adjacency can be thought of as an "atom" having "links" that extend to the neighboring vertices along the three coordinate axes (see Fig. 2). Thus the atom behaves somewhat like a Lego® block, with connections possible along 3 axes. In general, the lattice extension to the regular orthogonal grid system can be generated by translating the "atom" (a vertex and its emanating edges) throughout the grid system.

For example, consider an axis-aligned 2 x 2 x 2 cube centered at the vertex (i, j, k). If no edges are added to the grid system, we have 0-adjacency. Adjoining edges along three axes to neighboring faces, from vertex (i, j, k) to $(i+1, j, k)$, $(i, j+1, k)$ and $(i, j, k+1)$ respectively, gives 6-adjacency. Adjoining edges to the corners of the 2 x 2 x 2 cube gives 8-adjacency. Adjoining edges exclusively to midpoints of edges in the 2 x 2 x 2 cube gives 12-adjacency [15].

2.3. Data representation of a PVDS

The data representation of a PVDS is as follows. First we label all emergent (outgoing) edges at a vertex as "active" or "inactive", and store the Boolean vector of these values as <edge labels> at the vertex. Collectively, these edge labels enumerate all active edges within the three-dimensional lattice imposed on the orthogonal coordinate system. We store the augmented vertex information in a *vertex array*:

<*vertex array*> = array [<*vertex value*> <*edge labels*>]

For example, the data representation of a polymerized volume data set with 6-adjacency assigns a 4-tuple to each vertex of the grid: its gray-scale value followed by a three-dimensional Boolean vector of edge labels for edges emanating from the vertex in the positive i, j, k directions, respectively (Fig. 3).

2.4. Discrete rotation groups in 1D, 2D, and 3D

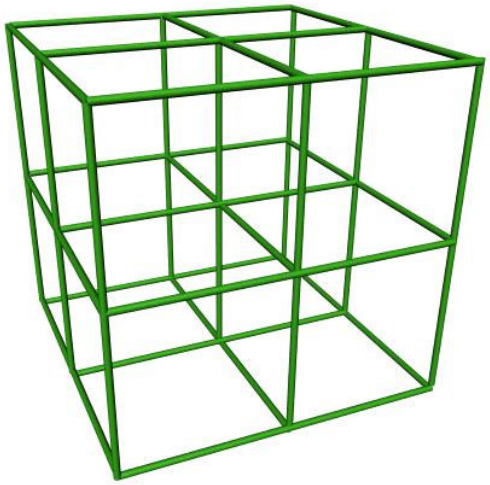
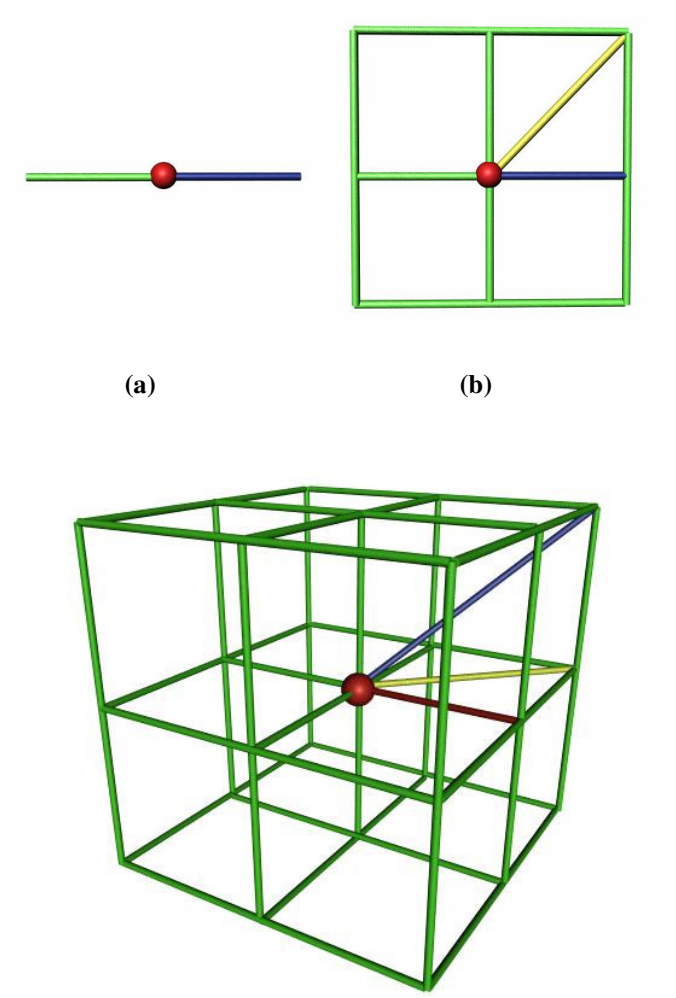


Fig. 4. A 2 x 2 x 2 cube of 8 neighboring unit cubes.

Our discussion will embrace grids in 1, 2, and 3 dimensions. Consider an interior vertex in a uniform grid system. The immediate environment of the selected vertex (here called the *root vertex*) is (1) in 1D: a linear interval of 2 neighboring unit intervals; (2) in 2D: a square of 4 neighboring unit squares; (3) in 3D: a cube of 8 neighboring unit cubes (Fig. 4); and (4), more generally, in nD: an n-cube of 2^n unit n-cubes. In this regular grid we constrain the set of undirected edges sharing the root vertex to be invariant under rotation by 90º about any axis. Once an edge (called the group generator) has been specified, all other edges of the set are generated by repeated 90º rotations. The number of distinct edges in the set is the adjacency of the vertex. Fig. 5 below exhibits generators for all rotationally invariant sets of edges about a vertex in 1, 2, and 3 dimensions. Table 1 summarizes the results. Edges generated by a common generator have terminal vertices at the same radius from the root vertex, as the 90º rotations preserve length. These radii are also given in Table 1.



(a) **(b)**

(c)

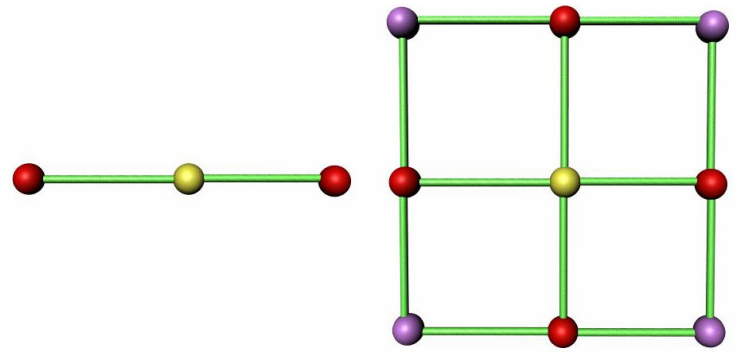
Fig. 5. Generators for rotationally invariant sets of edges about a vertex. (a) In 1-dimension (in blue) and 2-dimensions (in blue and yellow). (b) In 3-dimensions (in blue, yellow and red).

Lemma 1. For a given dimension, sets of edges invariant under 90º rotation can be built by any combination of generators. Accordingly, in 1D, edge adjacency of 0 and 2 can be constructed; and in 2D, edge adjacency of 0, 4 (axis-aligned), 4 (vertex centered), and 8 (both) can be constructed. Extending to 3D, edge adjacency of 0, 6, 8, 12, 14, 18, 20, and 26 can be constructed. In particular, in 3D, 26-adjacency implies the root vertex makes an undirected connection to every vertex in the 2 x 2 x 2 cube centered at the root vertex [4]. The adjacencies in 1D, 2D and 3D are shown in Fig. 6.

TABLE 1

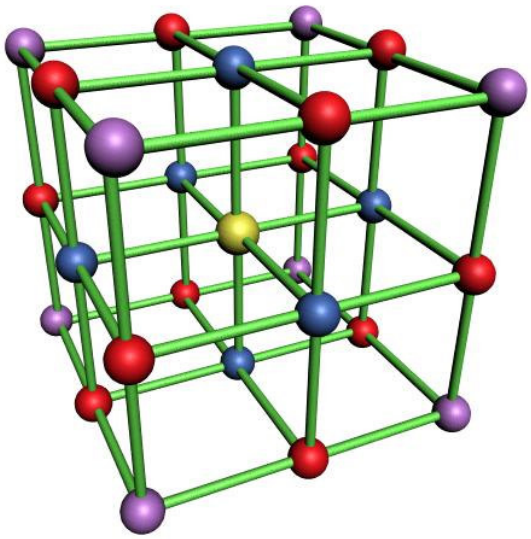
Adjacency in 1, 2 and 3 dimensions

Dimension	Adjacency	Radius
1	0-adj (no edges)	0
	2-adj (axis aligned)	1
2	0-adj (no edges)	0
	4-adj (axis aligned edges)	1
	4-adj (vertex centered)	$\sqrt{2}$
3	0-adj (no edges)	0
	6-adj (axis aligned, face centered)	1
	8-adj (vertex centered)	$\sqrt{3}$
	12-adj (edge centered)	$\sqrt{2}$



(a) **(b)**

Fig. 6. Maximal adjacency in 1, 2 and 3 dimensions (The center voxel is shown in yellow). (a) 2-adjacency in 1-dimension and 8-adjacency in 2-dimensions. (b) 26-adjacency in three dimensions. The six voxels in blue belong to 6-adjacency class, eight voxels in violet belong to 8-adjacency class and twelve voxels in red belong to 12-adjacency class.



(c)

Fig. 6. Continued.

CHAPTER III

DISCRIMINANT FUNCTIONS FOR INTERIOR/EXTERIOR EDGE-LABELING

Edges emanating from a vertex are assigned a Boolean label, {0,1}, with one label for each edge orientation, l, $(l = 1,\ldots,d)$ for the d-connected volume data set. Edge labeling uses discriminant functions, $y_l(\mathbf{g})$, one for each edge orientation l. Here $\mathbf{g}$ is a vector of gray-scale values from neighboring voxels. An edge is labeled 1 iff $y_l(\mathbf{g}) \geq 0$. The parameters of the discriminant function $y_l(\mathbf{g})$ are evaluated by supervised learning over the training sets $\mathbf{T}_l$ for a common edge orientation l. Construction of the training sets $\mathbf{T}_l, (l = 1,\ldots,d)$ is described in the Chapters VI and VII.

3.1. Construction of templates

For given d-adjacency relation, the decision to label an edge "active" (Boolean value 1) depends only on the edge and of course, on the gray-scale values of neighboring vertices. But which neighbors? Let the template $\mathbf{t}_l^d(i\,j\,k)$ designate the indexed set of neighboring vertices to be used in labeling an edge of orientation l emanating from vertex $(i\,j\,k)$. Let the edge orientation l be represented by the discrete vector (l_1, l_2, l_3), where $l_i \in \{0,1\}$ for $i = 1,2,3$. We now will impose two constraints upon $\mathbf{t}_l^d(i\,j\,k)$ such that its vertices are uniquely determined and indexed:

(1). Let $S_o(i\,j\,k)$ be the set of all terminal vertices of *outgoing* edges from vertex $(i\,j\,k)$. Let $S_I(i+l_1\ j+l_2\ k+l_3)$ be the set of all terminal vertices of *incoming* edges to vertex $(i+l_1\ j+l_2\ k+l_3)$. Then the set of neighboring vertices in our template $\mathbf{t}_l^d(ijk)$ satisfies $\mathbf{t}_l^d(i\,j\,k) = S_o(i\,j\,k) \cap S_I(i+l_1\ j+l_2\ k+l_3)$.

(2). Vertices in any one template $\mathbf{t}_l^d$ can be arbitrarily indexed. This indexing is then extended to other templates $\mathbf{t}_j^d$, $j \neq l$ by 90º rotations of $\mathbf{t}_l^d$ into $\mathbf{t}_j^d$. By convention, the tail

and head terminals of the edge with orientation *l* will be indexed as the first two vertices of $\mathbf{t}_l^d$ respectively.

The grid system of the volume data set is converted to a (directed) *lattice* $\mathbf{L}$ upon the imposition of a d-adjacency relation (Section 2 of Chapter II). If the direction of all rays in $\mathbf{L}$ is reversed, we obtain the (directed) *dual lattice* $\mathbf{L}_D$. Equivalently, outgoing edges from a vertex in $\mathbf{L}$ are incoming edges to the same vertex in $\mathbf{L}_D$. The initial choice of imposing the lattice $\mathbf{L}$ or its dual lattice $\mathbf{L}_D$ upon volume data set is essentially arbitrary, but either choice insures that an edge is examined only once. Constraint (1) above insures that the vertices associated with an edge are invariant whether we start with $\mathbf{L}$ or its dual $\mathbf{L}_D$: $\mathbf{t}_l^d(i\ j\ k)|_{\mathbf{L}} = \mathbf{t}_l^d(i+l_1\ j+l_2\ k+l_3)|_{\mathbf{L_D}}$, where we have imposed the convention that an edge in $\mathbf{L}$ and its dual in $\mathbf{L}_D$ are given the same orientation index *l*.

For 6-adjacency, the three templates $\mathbf{t}_1^3, \mathbf{t}_2^3, \mathbf{t}_3^3$, positioned at root vertex (i, j, k), consist of two vertices: the root vertex (i, j, k) and a second vertex $(i+1, j, k), (i, j+1, k)$, or $(i, j, k+1)$ respectively. For 12-adjacency, $S_o(i\ j\ k)$ and $S_l(i+l_1\ j+l_2\ k+l_3)$ overlap more, and share 6 vertices (Fig. 7).

3.2. Class-based edge labeling with ideal discriminant functions

Edges can also be labeled with a class label k, $k \in \{0,1,\ldots,K \mid K \geq 1\}$, provided appropriate labeled training sets are available. For example, we may choose to distinguish between edges that are interior, cross a boundary, or are wholly exterior (and perhaps noise-related) to objects (such as blobs or filaments) within the volume data set. Let class C_k denote a category of edges, equally applicable to edges of all orientations. For each object class C_k and edge orientation l, a discriminant function $y_{k,l}(\mathbf{g})$ must be generated such as to minimize the error of mislabeling *l*-oriented edges. Here $\mathbf{g}$ is the vector of gray-scale values of vertices in class-dependent template $\mathbf{t}_l^d$ associated with the edge being labeled.

The ideal discriminate function, $y_{l,k}(g)$, maximizing the likelihood of correct labeling, is given by

$$y_{k,l}(\mathbf{g}) = \ln p(C_k \mid \mathbf{g}, l) \\ = \ln p(\mathbf{g}, l \mid C_k) + \ln P(C_k) - \ln P(\mathbf{g}, l)$$

Here we have evoked Bayes' theorem [2]. Henceforth the term $\ln P(\mathbf{g}, l)$, the prior probability of $(\mathbf{g}, l)$, will be dropped as providing no discrimination among classes. The statistics on $p(g, l \mid C_k)$ and the prior probability $P(C_k)$ must be collected from the training set. We approximate the right-hand side of the equation by quadratic form in $\mathbf{g}$, with weights to be determined from the statistics of the training set. Discriminant functions are restricted to be no more than quadratic in $\mathbf{g}$ to minimize computational complexity, as mentioned in Chapter I.

For K = 1, let the two classes, C_k, $(k = 0,1)$ represent the class of edges respectively not contained or wholly contained in a common object in the sample volume data set. As above, we assign the edge l the Boolean label 1 if $y_{l,1}(\mathbf{g}) \geq y_{l,0}(\mathbf{g})$, else 0, where

$$y_{k,l}(\mathbf{g}) = \ln p(\mathbf{g}, l \mid C_k) + \ln P(C_k),\ k = 0,1.$$

In practice the discriminant functions, $y_l(\mathbf{g})$, may only approximate this ideal discriminant function. Introducing $y_l(\mathbf{g}) = y_{l,1}(\mathbf{g}) - y_{l,0}(\mathbf{g})$, we obtain an equivalent labeling of active edges iff $y_l(\mathbf{g}) \geq 0$.

3.3. Discriminant functions for 6-adjacency

The local information contained in a 6-connected template $\mathbf{t}_l^6(i\ j\ k)$, centered at a vertex $(i\ j\ k)$, consists of $g_0 = g(i\ j\ k)$ and $g_l \in \{g(i+1\, j\, k), g(i\ j+1\ k), g(i\ j\ k+1)\}$, as designated by $l = 1,2,3$ respectively. The general quadratic discriminant function

$y^6_{k,l}(\mathbf{g}) = y^6_{k,l}(g_0, g_l)$ for class C_k, $k = 1, \ldots, K$, edge orientation l, and symmetric in g_0, g_l, is given by:

$$\begin{aligned} y^6_{k,l}(g_0, g_l) &= w_{k0} + w_{k1}(g_0 + g_l) + w_{k2}\left(g_0^2 + g_l^2\right) \\ &\quad + w_{k3}(g_0 g_l) + w_{k4}\left|g_0 - g_l\right| + w_{k5}\left|g_0^2 - g_l^2\right| \end{aligned} \tag{1}$$

The weight vector $\mathbf{w} = (w_0, w_1, w_2, w_3, w_4, w_5)^T$ is then derived from supervised learning over the training set, $\mathbf{T}^6_{k,l}$, where the target value for an edge is 1 iff $y_{k,l}(g_0, g_l) \geq y_{j,l}(g_0, g_l)$ for all $j \neq k$. The training tries to minimize the error of misclassification.

3.4. Discriminant functions for 8-adjacency

The local information contained in a 8-connected template $\mathbf{t}^8_l(i\,j\,k)$, centered at a vertex $(i\,j\,k)$, again consists of g_0 and g_l, the gray-scale values respectively at vertex $(i\,j\,k)$ and head of its emergent *l*-designated edge.

For 8-adjacency, the general quadratic discriminant function, $y^8_{k,l}(\mathbf{g}) = y^8_{k,l}(g_0, g_l)$, restricted to this shared information, $g = (g_0, g_l)$, and symmetric in g_0 and g_l, is in functional form identical to $y^6_{k,l}(g_0, g_l)$, as developed for the 6-adjacency case (Section 3 above). The only difference is that l is allowed 8 values, rather than 6, and of course the weights will be different.

3.5. Discriminant functions for 12-adjacency

We first construct the vertices of the template $\mathbf{t}^{12}_l(i\,j\,k)$, centered at vertex $(i\,j\,k)$. As identified by geometrical construction (Fig. 7), the template vertices contain the two termini of the *l*-oriented edge and four additional vertices arrayed at the vertices of a rectangle passing through the midpoint of the *l*-edge and whose normal is the *l*-edge.

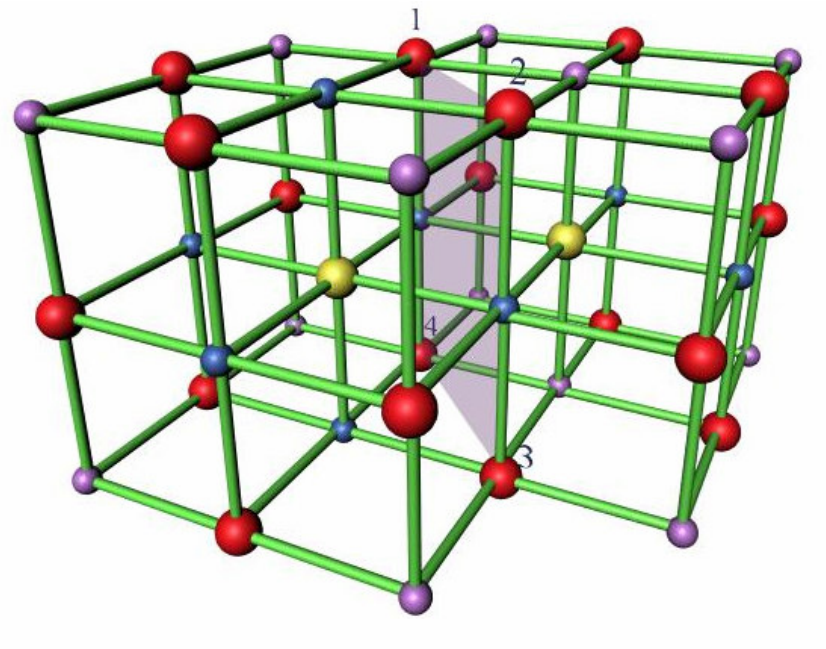


Fig. 7. Six vertices (two vertices in yellow along with the vertices 1, 2, 3 and 4) are shared between $S_O(i\,j\,k)$ and $S_I(i+l_1\;j+l_2\;k+l_3)$ for 12-adjacency.

Alternative verification comes from the listing in Table 2 of the two vertex sets: $S_o(0\,0\,0)$, the set of all terminal vertices of outward edges from vertex (000), and $S_I(1\,1\,0)$, the set of terminal vertices of incoming edges to vertex $(1\,1\,0)$. Observe that the vertices of $S_o(0\,0\,0)$ correspond to vertices of $S_I(1\,1\,0)$, apart from two exceptions: $(0,2,0)\notin S_o(0\,0\,0)$ and $(1,0,-1)\notin S_I(1\,1\,0)$. Therefore vertices of the template $\mathbf{t}_I^{12}(0\,0\,0)$ are six in number: the origin $(0\,0\,0)$ and the next five entries of the leftmost column of Table 2. Translating the template to vertex $(i\,j\,k)$, we derive the result of the geometric construction (Fig. 7).

TABLE 2

Construction of the template $\mathbf{t}_l^{12}(000)$ for $\vec{l} = (1,1,0)$

$S_o(000)$	Dual edge vectors	$S_l(110) =$ $\{(1,1,0),(1,1,0) + dualvectors\}$
$(0,0,0)$		$(1,1,0)$
$(1,0,1)$	$(-1,0,-1)$	$(0,1,-1)$
$(0,1,1)$	$(0,-1,-1)$	$(1,0,-1)$
$(1,0,-1)$	$(-1,0,1)$	$(0,1,1)$
$(0,1,-1)$	$(0,-1,1)$	$(1,0,1)$
$(1,1,0)$	$(-1,-1,0)$	$(0,0,0)$
$(1,-1,0)$	$(-1,1,0)$	$(0,2,0)$

We designate the gray-scale values of the tail and head of the *l*-edge as g_0 and g_l respectively, retaining the convention used above for 6-adjacency and 8-adjacency. The four vertices of the bisecting rectangle, cyclically labeled 1, 2, 3, and 4 (Fig. 7) have gray-scale values a_1, a_2, a_3, a_4 respectively. We impose two symmetry constraints on the form of the quadratic discriminate function, $y_l^{12}(\mathbf{g}) = y_l^{12}(g_0, g_l, a_1, \ldots, a_4)$, namely:

1. $y_l^{12}(\mathbf{g})$ is a symmetric function of g_0 and g_l, as previously evoked for 6-adjacency and 8-adjacency.

2. $y_l^{12}(\mathbf{g})$ is a cyclic invariant function of $a_1, \ldots, a_4$. The rectangle, whose vertices have gray-scale values $a_1, \ldots, a_4$, is not square; in fact the rectangle has an aspect ratio of $\sqrt{2}:2$ (Fig. 7). Nonetheless, in the interests of simplifying the form of the discriminant function we will evoke cyclic invariance.

Table 3 lists the symmetric linear and quadratic terms in g_0 and g_l, and then separately the cyclically invariant linear and quadratic terms in $a_1, \ldots, a_4$. The first, the terms in (g_0, g_l), contribute to the discriminant function a contribution identical to the discriminant function for 6-adjacency and 8-adjacency. The second, the terms in $(a_1, \ldots, a_4)$, are new. And thirdly are added cross-terms in (g_0, g_l) and $(a_1, \ldots, a_4)$.

TABLE 3

Cyclically invariant linear and quadratic terms in the discriminant function $y_l^{12}(g)$

Gray-scale values	**Cyclically invariant linear terms**	**Cyclically invariant quadratic terms**
Terms in (g_0, g_l)	$(g_0 + g_l)$	$(g_0^2 + g_l^2)$
	$\lvert g_0 - g_l \rvert$	$(g_0 g_l)$
		$\lvert g_0^2 - g_l^2 \rvert$
Terms in $(a_1, \ldots, a_4)$	$(a_1 + a_2 + a_3 + a_4)$	$(a_1^2 + a_2^2 + a_3^2 + a_4^2)$
	$\lvert a_1 - a_2 + a_3 - a_4 \rvert$	$(a_1a_2 + a_2a_3 + a_3a_4 + a_4a_1)$
		$(a_1a_3 + a_2a_4)$
		$\lvert a_1^2 - a_2^2 + a_3^2 - a_4^2 \rvert$
		$\lvert a_1a_2 - a_2a_3 + a_3a_4 - a_4a_1 \rvert$
		$\lvert a_1a_3 - a_2a_4 \rvert$

We will assume that $y_l(g)$ is independently invariant under the cyclic permutations (0,*l*), and (1, 2, 3, 4). The quadratic discriminant function $y_l^{12}(g)$, in full generality, is then given by:

$$
\begin{aligned}
y_l^{12}(g) = {} & w_0 + w_1\left(g_0 + g_l\right) + w_2\left|g_0 - g_l\right| + w_3(g_0^2 + g_l^2) \\
& + w_4\left(g_0 g_l\right) + w_5\left|g_0^2 - g_l^2\right| + \\
& w_6\left(g_0 + g_l\right)\left(a_1 + a_2 + a_3 + a_4\right) + \\
& w_7\left(g_0 + g_l\right)\left|a_1 - a_2 + a_3 - a_4\right| + \\
& w_8\left|g_0 - g_l\right|\left(a_1 + a_2 + a_3 + a_4\right) + \\
& w_9\left|g_0 - g_l\right|\left|a_1 - a_2 + a_3 - a_4\right| + \\
& w_{10}\left(a_1 + a_2 + a_3 + a_4\right) + w_{11}\left|a_1 - a_2 + a_3 - a_4\right| + \\
& w_{12}\left(a_1^2 + a_2^2 + a_3^2 + a_4^2\right) + \\
& w_{13}\left(a_1 a_2 + a_2 a_3 + a_3 a_4 + a_4 a_1\right) + \\
& w_{14}\left(a_1 a_3 + a_2 a_4\right) + w_{15}\left|a_1^2 - a_2^2 + a_3^2 - a_4^2\right| + \\
& w_{16}\left|a_1 a_2 - a_2 a_3 + a_3 a_4 - a_4 a_1\right| + w_{17}\left|a_1 a_3 - a_2 a_4\right|
\end{aligned}
$$

CHAPTER IV

DISCRIMINANT FUNCTIONS FOR BOUNDARY EDGE-LABELING

4.1. Characteristics of discriminant functions for boundary detection

Edges that cross a boundary have one terminal vertex inside and one outside an object of interest within the volume data set. This set of edges can be considered directed. Edges that go from the exterior of an object to the interior can be labeled 'entering edges'. And similarly, edges that go from the interior of an object to the exterior can be labeled 'exiting edges'. Examples of entering and exiting edges are shown in Fig. 8.

Labeling of boundary edges as either 'entering' or 'exiting' is not possible using the discriminant functions derived in Chapter III, since all those functions are symmetrical in the gray-level values of the terminal vertices of the edge. Here, asymmetrical discriminant functions are required that identify the classes of entering and exiting boundary edges. In this chapter, these asymmetrical functions are derived. Unlike discriminant functions derived in the previous chapter, which can be used to classify any edge in the volume data set, discriminant functions derived here can be used to classify only boundary edges. The recognition of boundary edges in the volume data set should be initially done using the functions derived in Chapter III. The discriminant functions for boundary-edge classification are also restricted to be quadratic in the gray-level values.

4.1.1. Discriminant functions for 6-adjacency

Using the same notation as before, the discriminant function for boundary edges classification for 6-adjacency is as follows:

$$y_l^6(g_0, g_l) = w_0 + w_1(g_0 - g_l) + w_2(g_0^2 - g_l^2)$$

Only terms that are asymmetrical in the gray-level values are considered since they serve the purpose of class-based labeling of boundary edges.

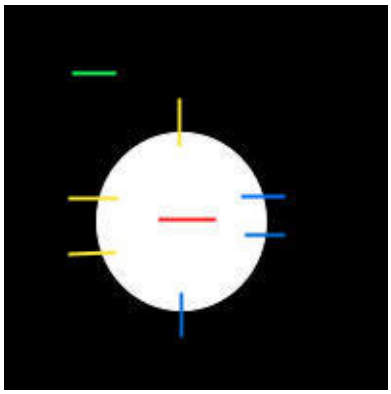


Fig. 8. Criteria for labeling edges as interior, boundary, and exterior: An interior edge is shown in red, exterior edge in green, entering and exiting edges in yellow and blue respectively.

4.1.2. Discriminant functions for 8-adjacency

The discriminant function for 8-adjacency is also dependent on only the gray-level values of the edge termini. Hence, this has the same form as the discriminant function for 6-adjacency. As mentioned in previous chapter, the difference is that l is allowed 8 values instead of 6 and g_l is determined in a different manner.

$$y_l^8(g_0, g_l) = w_0 + w_1(g_0 - g_l) + w_2(g_0^2 - g_l^2)$$

4.1.3. Discriminant functions for 12-adjacency

For 12-adjacency, four additional vertices also belong to the set of template vertices $\mathbf{t}_l^{12}(ijk)$, apart from edge termini as shown in Fig. 7 in the previous chapter. These template vertices are labeled 1, 2, 3 and 4, with gray-level values a_1, a_2, a_3, a_4 respectively. The order of these vertices change when the direction of the edge changes. Hence, the edge-labeling discriminant function is considered to be asymmetrical in the joint change of gray-level values of these four vertices (a_1, a_2, a_3, a_4) along with that of the edge termini (g_0, g_l). The function, $y_l^{12}(\mathbf{g}) = y_l^{12}(g_0, g_l, a_1, \ldots, a_4)$ is shown below.

$$y_l^{12}(\mathbf{g}) = w_0 + w_1\left(g_0 - g_l\right) + w_2\left(g_0^2 - g_l^2\right) + w_3\left(g_0 + g_l\right)\left(a_1 - a_2 + a_3 - a_4\right) +$$
$$w_4\left(g_0 - g_l\right)\left(a_1 + a_2 + a_3 + a_4\right) + w_5\left(g_0 - g_l\right)\left(a_1 - a_2 + a_3 - a_4\right) +$$
$$w_6\left(a_1 - a_2 + a_3 - a_4\right) + w_7\left(a_1^2 - a_2^2 + a_3^2 - a_4^2\right) +$$
$$w_8\left(a_1a_2 - a_2a_3 + a_3a_4 - a_4a_1\right) + w_9\left(a_1a_3 - a_2a_4\right)$$

4.1.4. The edge-labeling procedure

Boundary edges need to be labeled with a class label k, where k is either 'entering edge' or 'exiting edge'. This labeling is obtained by training discriminant functions of the above forms for the required connectivity using the appropriate labeled training sets. For each edge orientation l, a discriminant function $y_l(\mathbf{g})$ must be generated such as to minimize the error of mislabeling *l*-oriented edges. Here $\mathbf{g}$ is the vector of gray-scale values of vertices in class-dependent template $\mathbf{t}_l^d$ associated with the edge being labeled.

Let the two classes, C_k, $(k = 0,1)$ represent the class of boundary edges that are entering or exiting. As above, we assign the edge l to class C_0 if $y_l(\mathbf{g}) < 0$ and to class C_1 if $y_l(\mathbf{g}) \geq 0$. As in previous chapter, separate discriminant functions for each class k are not necessary here, as there are only two classes and a single discriminant function can classify input vectors into two classes based on thresholding.

4.2. Invariance under intensity scaling

Volume data, for which a polymerized volume data set is constructed, is typically obtained by 3D microscopy. The images obtained can show large variations in the average intensity and contrast. It is highly desirable that the polymerized volume data set constructed is largely independent of these changes. The weight parameters of discriminant functions should be independent of the average background level of the images. Otherwise, usage of the same set of discriminant functions, in spite of the variations in brightness in the set of 2D images, can lead to an incorrect PVDS.

To make the discriminant functions invariant to the low frequency spatial intensity component of the images, the images are pre-processed by using a high-pass filter, ensuring that only the very low spatial frequency component is eliminated.

CHAPTER V

VECTOR TRACING USING QUASI-PLANAR TEMPLATES

This chapter discusses an algorithm for automated vector tracing first described in [1]. The paper presents a rapid automated algorithm for three-dimensional tracing of neurons within a serial stack of images obtained by fluorescence three-dimensional microscopy. In this chapter we first explain the original algorithm, as applied to neuronal data sets obtained in our laboratory. A variant of this algorithm using Frenet frames is presented later. This algorithm has been implemented by Busse [6]. Results from this algorithm are used to construct training sets for neuronal data sets, as explained in Chapter VII.

The vector tracing algorithm starts with pre-selected seed points. Given a seed point and a hypothesized trajectory direction along the neurite, the neuronal structure is traced recursively using a method based on matched filters or templates. The tracing algorithm processes only voxels close to neurites being tracked, saving computation time. This algorithm is also helpful for quick compression and storage of data sets. Due to these advantages, we have adopted this algorithm to trace neuronal structures in our data sets, and to extract training sets for the polymerization of the neuronal volume data sets.

5.1. Construction of quasi-planar templates

Construction of three-dimensional templates for different orientations is explained in this section. Let $\boldsymbol{p}$ be any position along the neurite trajectory, where the maximum template responses along four directions (left, right, top, and bottom) will be calculated. Let $\boldsymbol{us}$ be trial unit vector along the trajectory at that position. Let $\boldsymbol{ups}$ be a unit vector representing the direction along which the template response needs to be calculated. Using RPI frames, this orientation corresponds to one of the four perpendicular shift directions for tracing, as shown in Table 4. Using Frenet frames, this orientation depends on normal and binormal vectors for tracing, as shown in Table 5.

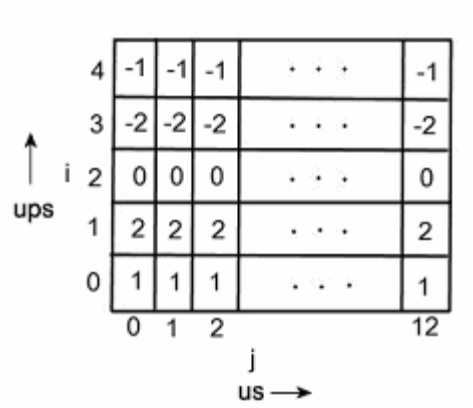


Fig. 9. 5×13 **kernel**

The basic kernel shown in Fig. 9 is used to construct the three-dimensional templates. A 5×13 kernel is used in our implementation as it worked well for our data sets. Let $v(i, j)$ represent the value of kernel at position (i, j) where $0 \le i \le 4$ and $0 \le j \le 12$. Let point (*x, y, z*) with integer coordinates denote the bottom-most voxel where the template is placed. Then, for all combinations of *i* and *j,* where $0 \le i \le 4$ and $0 \le j \le 12$, voxel positions in three-dimensional space (*tx, ty, tz*), where $v(i, j)$ occurs are given by:

$$tx = x + round(j^{*}\boldsymbol{us}.x) + round(i^{*}\boldsymbol{ups}.x)$$
$$ty = y + round(j^{*}\boldsymbol{us}.y) + round(i^{*}\boldsymbol{ups}.y)$$
$$tz = z + round(j^{*}\boldsymbol{us}.z) + round(i^{*}\boldsymbol{ups}.z)$$

Using these formulae, individual cells from of the templates are placed in a three-dimensional space at the required orientation and the integrated response of the kernel is calculated.

5.2. Two-dimensional tracing

Two-dimensional tracing will be explained first. The kernels used in 2D tracing are formed by repetitions of the basic template $[-1\ -2\ 0\ 2\ 1]^{T}$. For different trajectory

orientations, $5 \times K$ kernels are constructed for the detection of the left and right boundaries. At every centerline point while tracing along the neurite trajectory, these templates are moved perpendicular to the left and right boundaries, and the matched filter response is computed. That displacement of the template perpendicular to the hypothesized trajectory which gives the maximum correlation response defines the boundary. The next position along the trajectory is computed based on the boundary orientation identified and the previous position. Trial orientations in this algorithm can be discretized to a set of values as required, where 16 and 32 are used in [1].

The tracing process uses a predictor-corrector method. If $\boldsymbol{p}^{\boldsymbol{i}}$ is the centerline point and $\boldsymbol{u}^{\boldsymbol{i}}$ is the unit direction at that point, then the equation for finding the next position is given below:

$$\tilde{\boldsymbol{p}}^{i+1} = \boldsymbol{p}^{i} + \alpha^{i}\boldsymbol{u}^{i}$$
$$\boldsymbol{p}^{i+1} = \tilde{\boldsymbol{p}}^{i+1} + \boldsymbol{v}^{i+1}$$

Here, is $\boldsymbol{v}^{i+1}$ a correction, or fine-tuning vector, to insure a smoother trace when the local curvature is high, '~' indicates approximation, and α^{i} is the step size [9].

5.3. Three-dimensional tracing

For three-dimensional tracing, neurites are approximated by generalized cylinders, typically by cylinders with elliptical cross-sections and displaying curvature along their trajectory. In three dimensions, sampling is done in two perpendicular planes to trace the periphery of the boundary. Corresponding to these two planes, four sets of templates labeled 'top', 'bottom', 'left' and 'right' are defined. Construction of templates has been explained previously.

In 3-D space, directions are described using spherical coordinates, θ and ϕ in a right- handed coordinate system; θ describes a rotation around the Z-axis, and ϕ describes a rotation around the Y-axis obtained after being rotated by θ around the Z-axis (Fig. 10). If θ and ϕ are discretized to N values each, then we get a total number of $N \times N$ directions. A template is created for each hypothesized trajectory orientation.

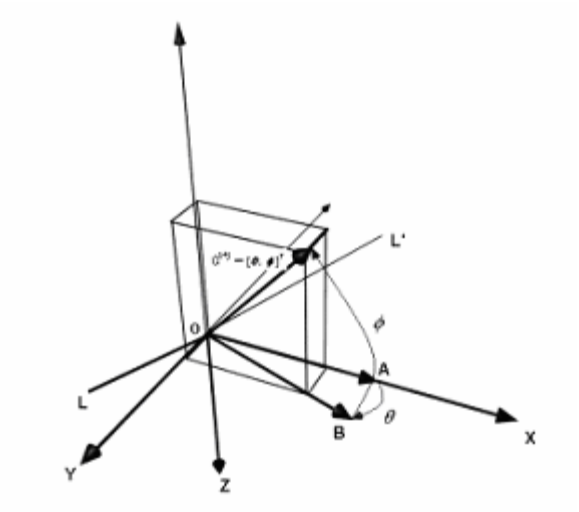


Fig. 10. Coordinate system for specifying angular directions [1].

Hence a total of $4 \times N^2$ templates are created. We follow the convention of the paper [1], where directions are represented by integer indexes s_1 and s_2, with s_1 and $s_2 \in$ {0, … , N-1}. A unit vector **u** with the orientation $[\theta = 2\pi s_1 / N, \phi = 2\pi s_2 / N]$ is expressed as $\mathbf{u} = [s_1, s_2]^T$. In our implementation, totally 16*9 = 144 directions are followed as it worked well for our data sets. During tracing, the angular θ and ϕ are constrained to a smaller number of angles ($< N \times N$), as only a span of 180° is considered for forward trajectory orientations, instead of 360°, which would include backward directions also. Horizontally the total span of 360° is considered.

As explained in the case of two-dimensional tracing, tracing in three dimensions is done by computing the correlation responses of the template kernels and finding their maximum responses. Let $R(\boldsymbol{u}_R, k, \boldsymbol{p})$ be the correlation response of a right template of length k, and direction $\boldsymbol{u}_R$, when the template is centered at the point $\boldsymbol{p}$. Similarly, let $L(\boldsymbol{u}_L, k, \boldsymbol{p}), T(\boldsymbol{u}_T, k, \boldsymbol{p})$ and $B(\boldsymbol{u}_B, k, \boldsymbol{p})$ be the responses of the left, top, and bottom templates respectively. Collectively, these are called “template responses”.

The procedure for tracing a 3-D generalized cylinder structure is shown in Fig. 11. The tracing starts at a point $\boldsymbol{p}^i$ on the centerline with an initial direction $\boldsymbol{u}^i$. Initially

we have estimates of an initial seed point positioned on or near the centerline and of its initial direction. Let these estimates be $\tilde{\boldsymbol{p}}^i$ and $\tilde{\boldsymbol{u}}^i$ for position and direction respectively. These estimates are refined by finding the maximum template responses $R(\boldsymbol{u}_T^i, k, \boldsymbol{p}_T^i)$, $R(\boldsymbol{u}_B^i, k, \boldsymbol{p}_B^i)$, $R(\boldsymbol{u}_R^i, k, \boldsymbol{p}_R^i)$, and $R(\boldsymbol{u}_L^i, k, \boldsymbol{p}_L^i)$ of the top, bottom, right and left responses respectively. These four templates are moved along their corresponding perpendicular shift directions, as shown in Table 4 for the four templates. The position where the template gives maximum response is noted. The top, bottom, left and right templates can be considered as "paddle wheels", as shown in Fig. 11.

TABLE 4

Perpendicular-shift directions for the templates [1]

Template	Direction	Perpendicular Shift Direction
Right	$\boldsymbol{u}_R = \begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$	$\boldsymbol{u}_{R\perp} = \begin{bmatrix} s_1 + \frac{N}{4} & s_2 \end{bmatrix}^T$
Left	$\boldsymbol{u}_L = \begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$	$\boldsymbol{u}_{L\perp} = \begin{bmatrix} s_1 - \frac{N}{4} & s_2 \end{bmatrix}^T$
Top	$\boldsymbol{u}_T = \begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$	$\boldsymbol{u}_{T\perp} = \begin{bmatrix} s_1 & s_2 - \frac{N}{4} \end{bmatrix}^T$
Bottom	$\boldsymbol{u}_B = \begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$	$\boldsymbol{u}_{B\perp} = \begin{bmatrix} s_1 & s_2 + \frac{N}{4} \end{bmatrix}^T$

The maximum correlation response occurs when the template coincides with the neurite boundary. Let the points on boundary that produce maximum template responses be $\{ \boldsymbol{p}_R^i, \boldsymbol{p}_L^i, \boldsymbol{p}_T^i, \boldsymbol{p}_B^i \}$. Let $\{ \boldsymbol{u}_R^i, \boldsymbol{u}_L^i, \boldsymbol{u}_T^i, \boldsymbol{u}_B^i \}$ local direction estimates be the set of template orientations that produce these maximum responses. If M is the maximum expected axon/dendrite diameter and $\sum$ is the set of all possible directions, the selection of seed

point and direction is based on the maximum response for the top template that can be written as:

$$(\boldsymbol{p}_T^i, \boldsymbol{u}_T^i) = \underset{\{(\boldsymbol{p},\boldsymbol{u}_T) | \boldsymbol{p}=\tilde{\boldsymbol{p}}^i + m\boldsymbol{u}_{T\perp},\ m=1,\ldots,M/2,\ and\ \boldsymbol{u}_T \in \Sigma\}}{\arg\max} T(\boldsymbol{u}_T, k, \boldsymbol{p})$$

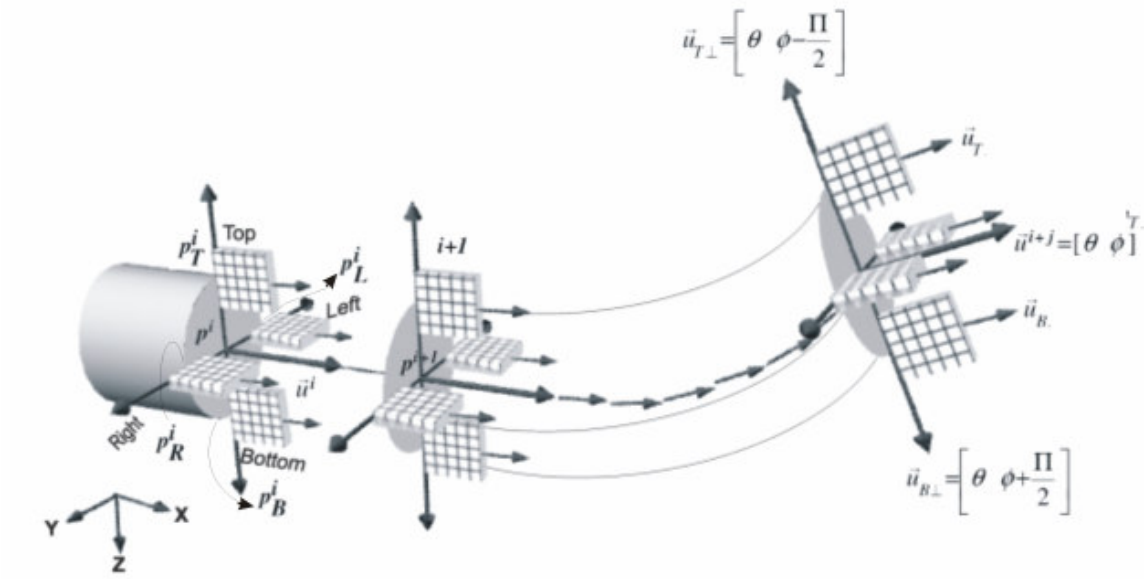


Fig. 11. Illustration of vector tracing algorithm using RPI frames (frames based on perpendicular shift directions shown in Table 4) [1].

Using these responses estimates of $\tilde{\boldsymbol{p}}^i$ and $\tilde{\boldsymbol{u}}^i$ are refined. Let $\hat{R} = R(\boldsymbol{u}_R^i, k, \boldsymbol{p}_R^i)$ denote the maximal response of the right template and a similar notation be followed for other template responses. Then the equations to refine the location and direction estimates $\tilde{\boldsymbol{p}}^i$ and $\tilde{\boldsymbol{u}}^i$ can be written as:

$$\boldsymbol{p}^i = \begin{bmatrix} x^i & y^i & z^i \end{bmatrix}^T$$

$$= \frac{\tilde{\boldsymbol{p}}^i}{2} + \begin{bmatrix} \dfrac{\hat{R}^i x_R^i + \hat{L}^i x_L^i}{2(\hat{R}^i + \hat{L}^i)} & \dfrac{\hat{R}^i y_R^i + \hat{L}^i y_L^i}{2(\hat{R}^i + \hat{L}^i)} & \dfrac{\hat{T}^i z_T^i + \hat{B}^i z_B^i}{2(\hat{T}^i + \hat{B}^i)} \end{bmatrix}^T$$

$$\boldsymbol{u}^i = \begin{bmatrix} s_1^i & s_2^i \end{bmatrix}^T$$

$$= \frac{\tilde{\boldsymbol{u}}^i}{2} + \begin{bmatrix} \dfrac{\hat{R}^i \tilde{s}_{1R}^i + \hat{L}^i \tilde{s}_{1L}^i}{2(\hat{R}^i + \hat{L}^i)} & \dfrac{\hat{T}^i \tilde{s}_{2T}^i + \hat{B}^i \tilde{s}_{2B}^i}{2(\hat{T}^i + \hat{B}^i)} \end{bmatrix}^T$$

Based on these estimates, position and direction of the next centerline point is given by:

$$\tilde{\boldsymbol{p}}^{i+1} = \boldsymbol{p}^i + \alpha^i \boldsymbol{u}^i$$

$$\tilde{\boldsymbol{u}}^{i+1} = \boldsymbol{u}^i$$

Thus, the estimates for the next successive position and orientation at that position are obtained. The process as described above is repeated to trace the structure. Thus, by recursive procedure, successive positions along the centerline, $\boldsymbol{p}^{i+1}$, $\boldsymbol{p}^{i+2}$, …, and directions $\boldsymbol{u}^{i+1}$, $\boldsymbol{u}^{i+2}$, …, at those positions are obtained. Proceeding in this way leads to tracing of the entire neurite structure. The recursive tracing process is continued until stopping criteria are met [1].

5.4. Vector tracing using Frenet frames

The previous section discussed the vector tracing algorithm. A modification to this algorithm is presented in this section. This algorithm has been implemented by Busse [6].

The original algorithm used perpendicular shift directions shown in Table 4 for calculating top, bottom, left and right template responses. The orientations of these templates are changed by using a Frenet frame of reference at every position along the generalized cylinder trajectory. The Frenet frame at any point along the trajectory curve is defined by three vectors $\boldsymbol{T}$, $\boldsymbol{N}$, and $\boldsymbol{B}$, which denote unit length tangent, normal and binormal vectors respectively (Fig. 12). These vectors are given by [25]:

Tangent, $\boldsymbol{T} = \boldsymbol{V} / |\boldsymbol{V}|$

where $\boldsymbol{V}$ is the derivative of the trajectory curve.

Normal, $\boldsymbol{N} = \boldsymbol{K} / |\boldsymbol{K}|$

where $\boldsymbol{K} = \boldsymbol{V} \times \boldsymbol{A} \times \boldsymbol{V} / |\boldsymbol{V}|^4$ and $\boldsymbol{A}$ is the second derivative of the curve.

Binormal Vector, $\boldsymbol{B} = \boldsymbol{T} \times \boldsymbol{N}$.

TABLE 5

Orientations of displacement of templates while using Frenet frames

(a) Top and bottom templates

Condition	Orientation of displacement of top template	Orientation of displacement of bottom template
$\boldsymbol{N}.z < 0$	Normal	Reverse Normal
$\boldsymbol{N}.z \geq 0$	Reverse Normal	Normal

(b) Left and right templates

Condition	Orientation of displacement of left template	Orientation of displacement of right template
$\boldsymbol{B}.y < 0$	Binormal	Reverse binormal
$\boldsymbol{B}.y \geq 0$	Reverse binormal	Binormal

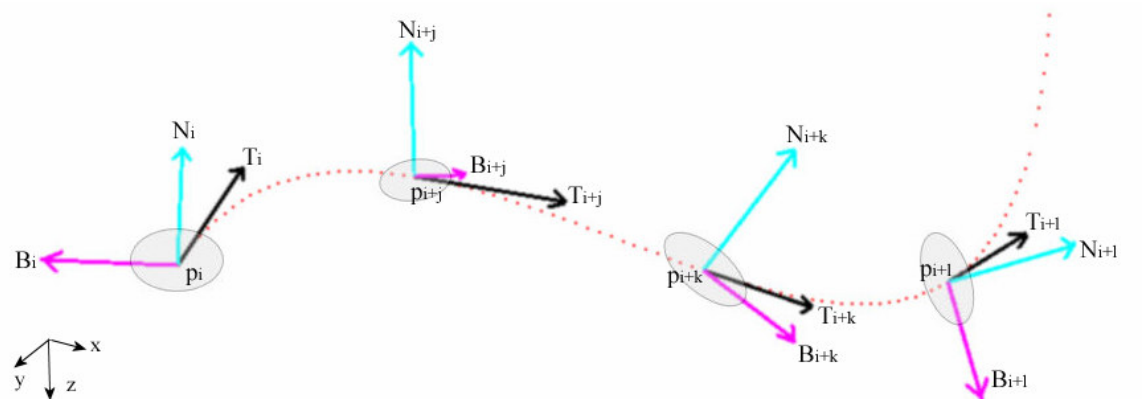


Fig. 12. Illustration of vector tracing algorithm using Frenet frames (tangent vectors are shown in black, normal vectors in blue and binormal vectors in pink).

For implementation a backward difference to the previous center point is used to approximate normal vector, ***N***. Since these vectors represent curvature of the trajectory more appropriately, using Frenet frames can improve the accuracy of tracing.

CHAPTER VI

CONSTRUCTION OF TRAINING SETS FOR SYNTHETIC VOLUME DATA

The discriminant functions developed in previous chapters must be trained and their weight vectors determined. The first step in this training process is the construction of training sets from homogeneous samples of the volume data set. These training data sets allow the classification of edges first in the sample volume data, and then, by generalization, throughout the volume data set.

The PVDS construction process is first illustrated on synthetic neuronal data constructed from graphical primitives, prior to the construction of PVDS on real-world biological data. For this purpose neuronal data that can occur in real-world brain mircostructure were generated synthetically. Neuronal data can be divided into two broad categories: filamentary data and blob data. Filamentary data is generated from fibrous neurites, like axons and dendrites. Blob data is generated from neuron cell bodies. Both types of neuronal data were synthesized and the PVDS were constructed for the synthetic volume data sets so generated.

Below, we explain the construction of this synthetic data and the design of the corresponding training sets. Construction of training sets for real-world neuronal data is deferred until the next chapter.

6.1. Synthetic data generation

The synthetic data described below resembles mouse brain microstructural data obtained from the knife-edge scanning microscope. Generation of the synthetic data can be divided into following five steps:

i. Construction of 3D hard-edge generalized cylinders
ii. Slicing of each cylinder into successive 2D sections
iii. Correction for section thickness
iv. Approximation of diffraction-limited imaging by convolving with a Point Spread Function (PSF)

v. Anti-aliasing (to compensate for the finite size of sensor pixels)

i & ii. Creation of 2D images using 3D models

Generalized cylinders representing neurites, or blobs representing cell bodies, were modeled in the software Maya. These three-dimensional models were then sectioned into successive two- dimensional images using the same software. The three-dimensional models are sliced into very thin sections to allow later correction for slice thickness. Three-dimensional models for fibers with branching and spheres were obtained from Doddapaneni [11].

iii. Correction for section thickness

Neuronal volume data is scanned from serial (physical or optical) sections. In the KESM, this section thickness is typically 0.5 μm. Comparable synthetic images can be composited from a number of ultra-thin sections (each bit-mapped as black/white) and averaging these subsectional images to form each simulated image. The variation in gray-level values obtained when these successive ultra-thin subsectional images are composited is shown graphically in Fig. 13.

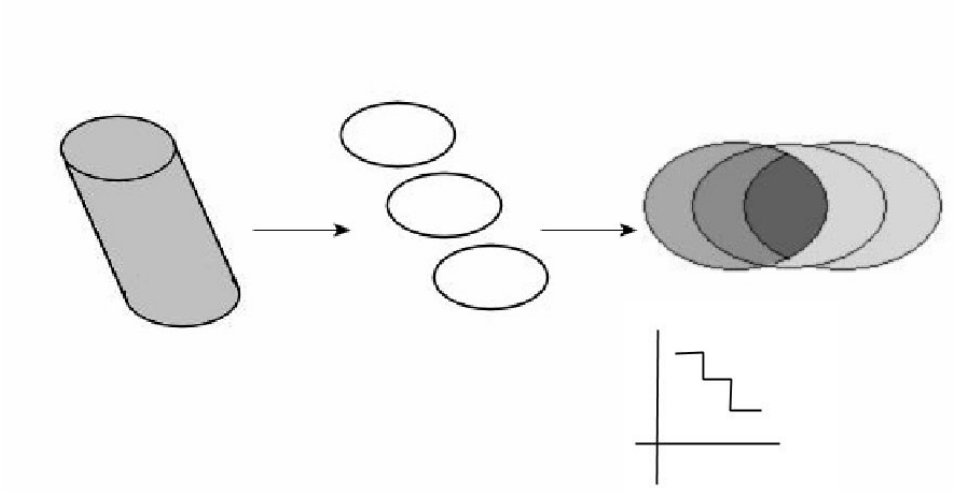


Fig. 13. Correction for section thickness by compositing subsectional hard-edged images.

iv. Point spread function

The biological neuronal data used in this thesis has been obtained from the knife-edge scanning microscope developed in our laboratory [21]. Images are captured by a scanline camera under diffraction-limited optics. Accordingly, diffraction-induced blur occurs in the images. This blur can be modeled by a two-dimensional point spread function. To ensure that synthetic data resembles the real-world data, the synthetic volume data sets were convolved with a point spread function.

The point spread function used is given by

$$y = (2 * J_1(x) / x)^2 ,$$

where $J_1(x)$ is the Bessel function of x of the first kind of order 1. This distribution is shown in Fig. 14. According to the Rayleigh criterion, two points of light in the image plane can be distinguished when in the focal plane the second point lies in the first diffraction minimum of the other. Thus, for Nyquist sampling, the spatial sampling

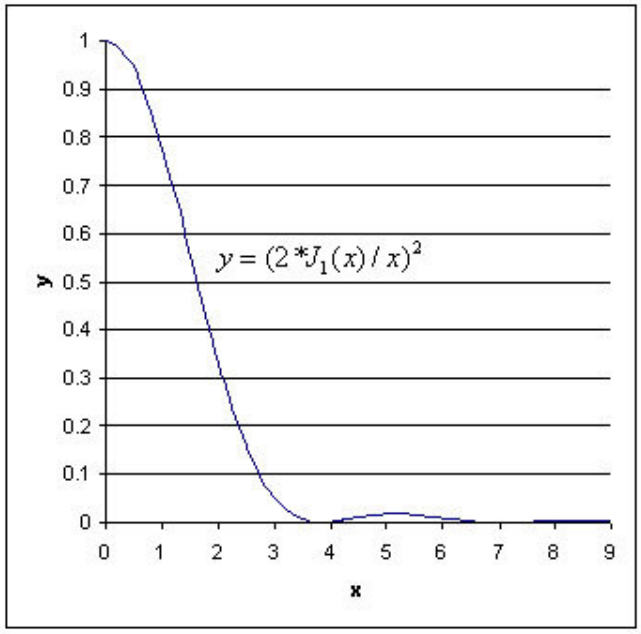


Fig. 14. Fraunhofer diffraction at a circular aperture - 2D point spread function [3].

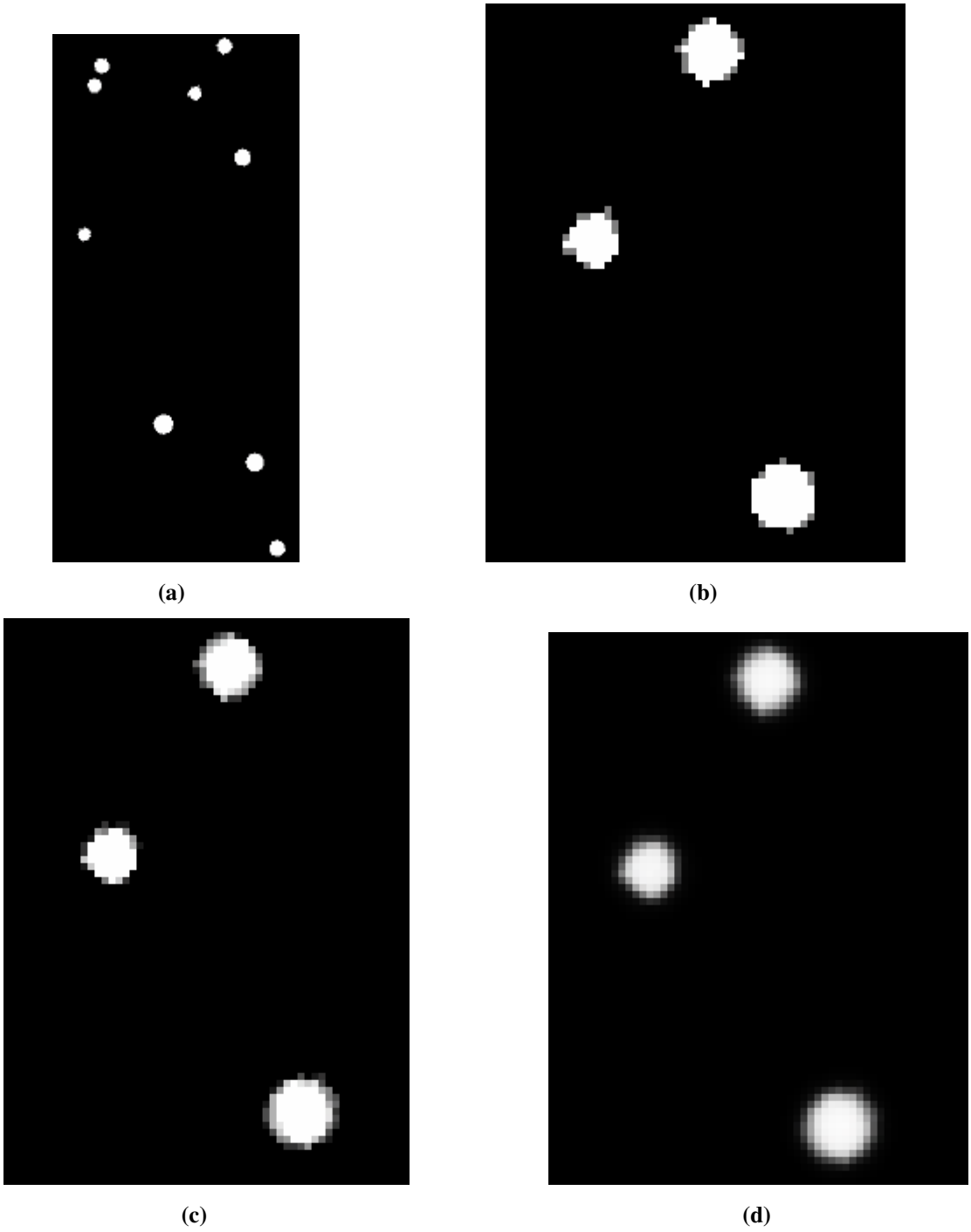
interval should be half the distance between the central maximum and the first zero of the diffraction pattern, as shown in Fig. 14. This first zero (y = 0) of the circular symmetric Fraunhofer diffraction pattern occurs at x = 3.83, which corresponds to two pixel widths from the central maximum. The PSF is convolved across the 2D synthetic images according to this criterion for resolution.

v. Anti-aliasing

The images thus created using hard-edged 3D models suffer from aliasing, though the PSF filtering helps in removing aliasing. Aliasing in images is eliminated before using this synthetic data set. Anti-aliasing is done using a 3×3 box filter, which is considered to be sufficient for this purpose. Application of this technique leads to our final synthetic data set.

6.2. Construction of training sets

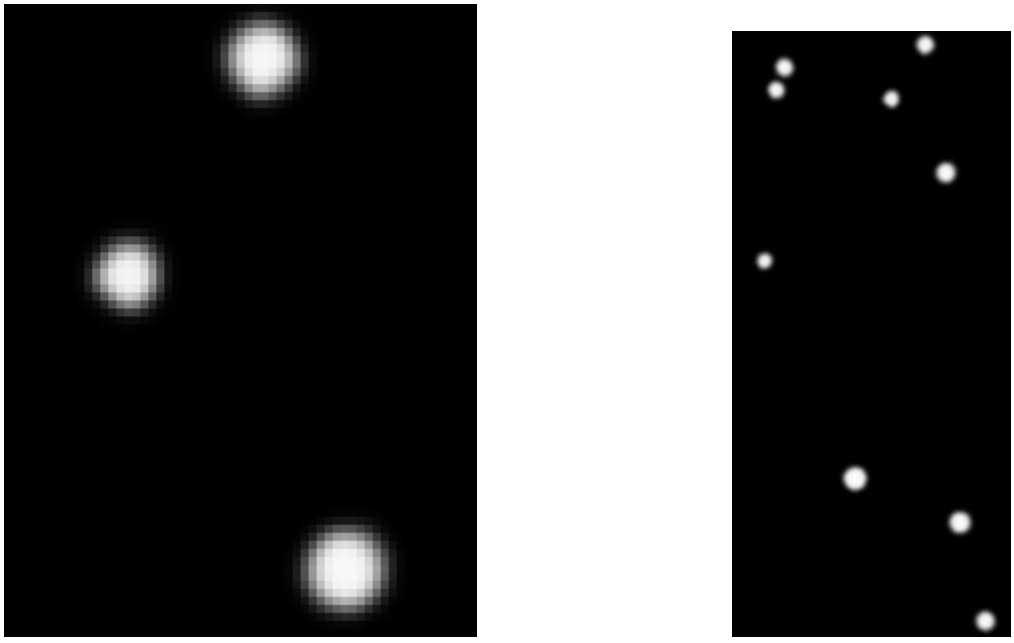
The process of training set construction from the synthetic images is explained in this section. The final synthetic volume data set (image stack) is obtained after averaging a number of ultra-thin sub-sections to simulate the gray level distribution in sections (typically 0.5 μm thick) used in the KESM; blurring using PSF; and anti-aliasing, as explained above. Edges in these final images are classified based on the corresponding initial image. Since the simulated image of a tissue section is obtained from a set of 2D images, we select the central image in that set as the reference image (ground truth) for classification of edges as interior, boundary or exterior. As the initial images are very thin, the displacement of the synthetic fibers/cell bodies from image to image is slight. Hence, using a center image in the initial set as a reference image creates a good classification of edges for the simulated image. For example, if a set of five images is used to obtain each simulated image of a tissue section, then the classification of edges in the final image is based on the third image in the set of five ultra-thin images. Since the reference image has a sharp gray-level difference between the interior and the exterior, edges in this image can be classified based on simple thresholding. An illustration of these images in successive stages is shown in Fig. 15.



(a) **(b)**

(c) **(d)**

Fig. 15. Simulated filamentary data set of fibers with branching. (a) A sub-sectional slice. (b) Blow-up of a part of image shown in Fig. 15.a. (c) Blow-up of a part of image obtained after averaging five successive ultra-thin images (with image shown in Fig. 15.a. as the center one) . (d) Image obtained after applying 2D point spread function to the blown-up part shown in Fig. 15.c. (e) Same part of the image after applying anti-aliasing filter. (f) Final synthetic image. Training samples from this final image are classified in accordance with the image shown in Fig. 15.a.



(e) **(f)**

Fig. 15. Continued.

6.3. Filamentary data

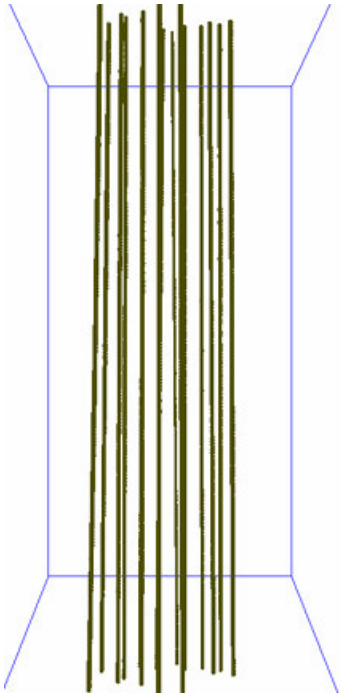
This type of data is generated by neurites like dendrites and axons, especially as seen in Golgi-stained tissue. Data sets resembling different kinds of fibers were constructed artificially as illustrated in Fig. 16 to Fig. 19. The volume data sets are displayed based on classification of edges in training sets.

6.4. Blob data

This type of data is generated by neuronal cell bodies, especially as seen in Nissl-stained tissue. Data sets resembling spheres, elliposoids and cylinders were constructed artificially as illustrated in Fig. 20. Some local views of these training sets are shown in Fig. 21.

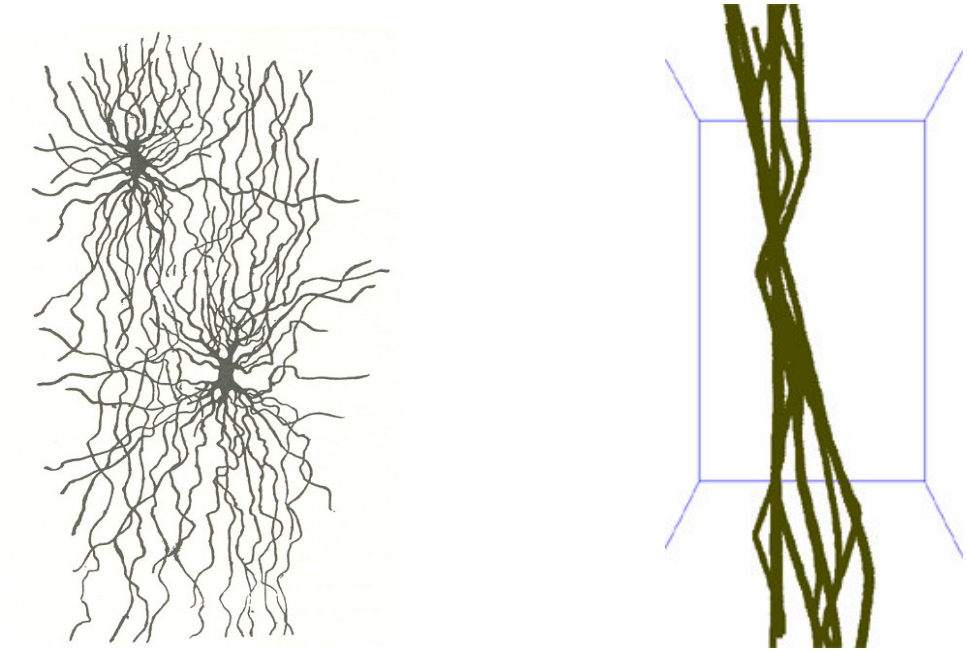
Fig. 16. Fine fibers: This data models thin dendrites and axons that occur in microscopic data of the mouse brain. Dendrites and axons are specialized extensions that project from the neuronal cell body.

(a)



(b)

Fig. 17. Parallel fibers: Parallel fibers occur when fine fibers extend such that they are all approximately parallel to each other. (a) Part of a sagittal preparation through the corpus striatum of a several-day-old rabbit. [Figure 326 in *Histology of the Nervous System* [8]; reproduced with permission of the publisher]. (b) Corresponding synthetic data.

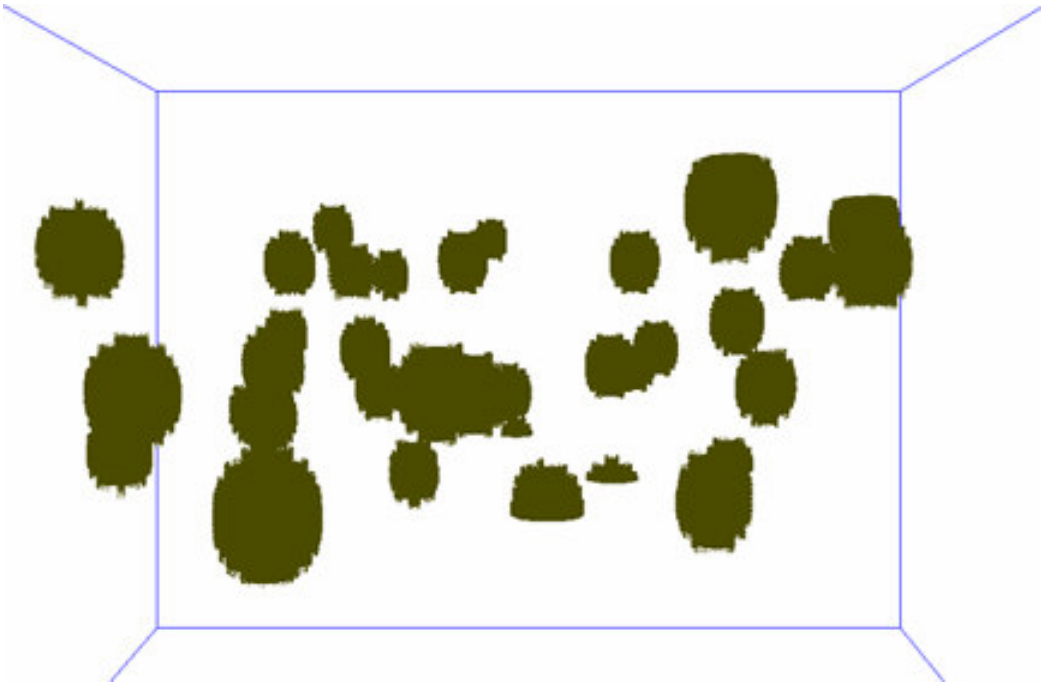


(a) **(b)**

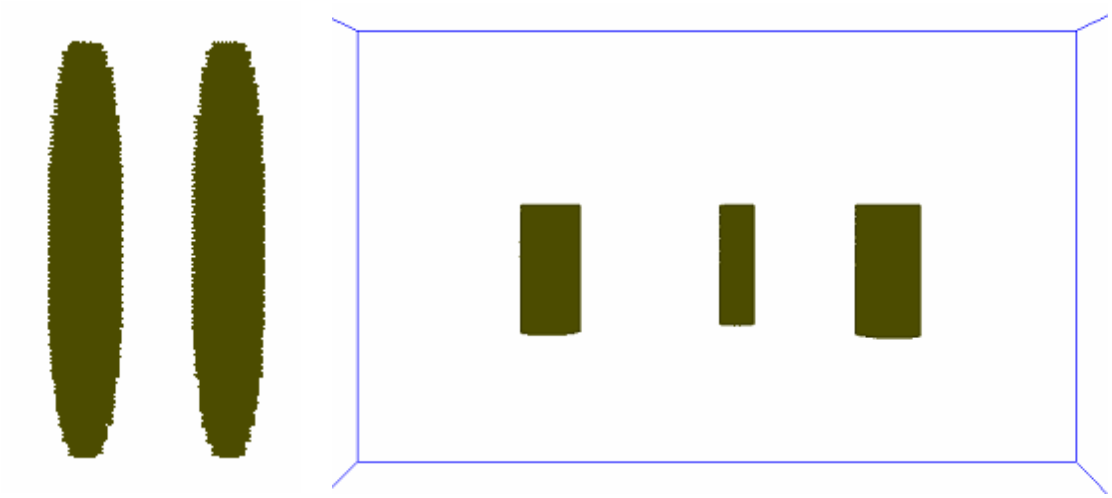
Fig. 18. Fibers with branching: This data set models junctions, evoking points where a dendrite or axon branches. (a)) An image of Glial cells in white matter of the adult human brain [Figure 78 in *Histology of the Nervous System* [7]; reproduced with permission of the publisher]. (b) Corresponding synthetic data.

(a) **(b)**

Fig. 19. Neuropil-like mats of fine fibers: Neuropil is a mesh of fine axonal fibers commonly seen in Golgi-stained brain tissue. (a) The hypoglossal nucleaus in a near-term rabbit fetus, generated by Golgi method. [Figure 293 in Histology of the Nervous System [7]; reproduced with permission of the publisher] (b) Corresponding synthetic data.

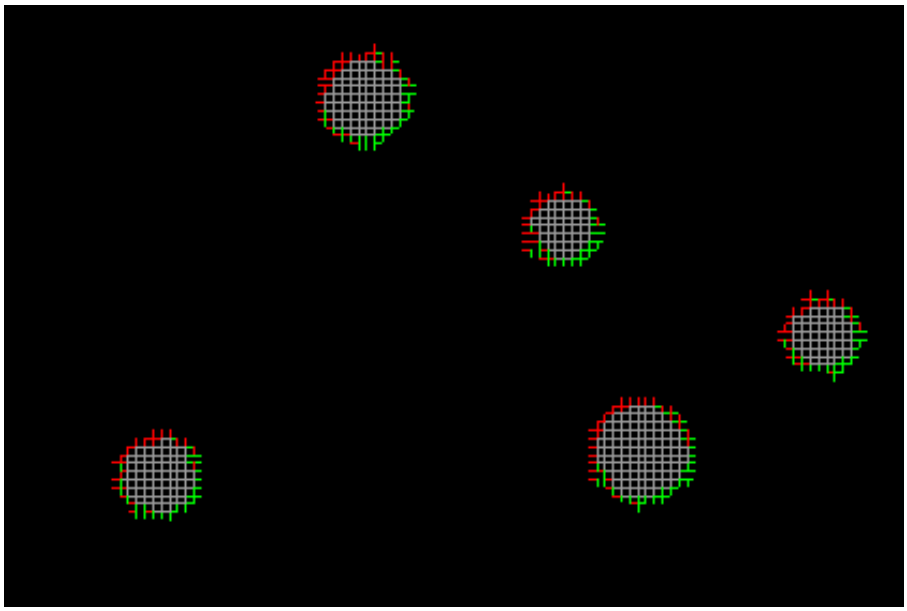


(a)

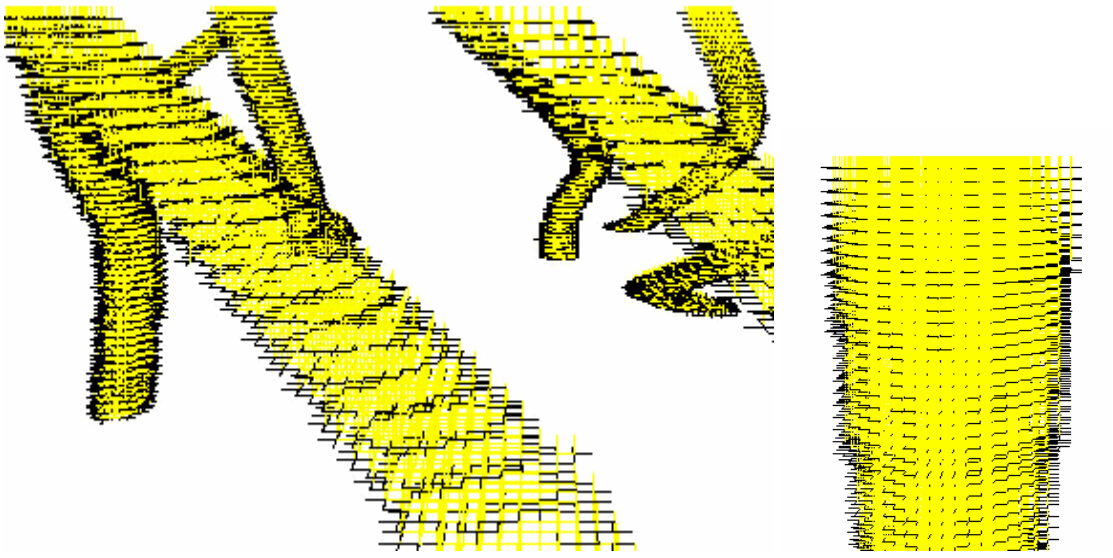


(b) **(c)**

Fig. 20. Synthetic data sets for blobular data. (a) Spheres. (b) Ellipsoids. (c) Cylinders.



(a)



(b) **(c)**

Fig. 21. Local views of training sets. (a) 2D view: Interior edges in gray, entering edges in red and exiting edges in green. (b) 3D view of filamentary data: Interior edges in yellow and boundary edges in black. (c) 3D view of blobular data: Interior edges in yellow and boundary edges in black.

CHAPTER VII

CONSTRUCTION OF TRAINING SETS FOR THREE-DIMENSIONAL MICROSCOPY DATA OF MOUSE BRAIN MICROSTRUCTURE

A goal of this thesis is to demonstrate the construction of polymerized volume data sets for mouse brain microstructure at a neuronal level of detail. Using a base-line segmentation algorithm, training sets were constructed from the neuronal volume data. The methods used generalize to volume data obtained from all known forms of 3D microscopy.

Volume data of mouse brain microstructure used in this thesis were obtained using a knife-edge scanning microscope, as explained in Section 1 below. Construction of training sets for Nissl-stained tissue is then presented. The vector tracing algorithm, presented in Chapter V, is used to generate training sets for the real-world biological data of mouse brain microstructure.

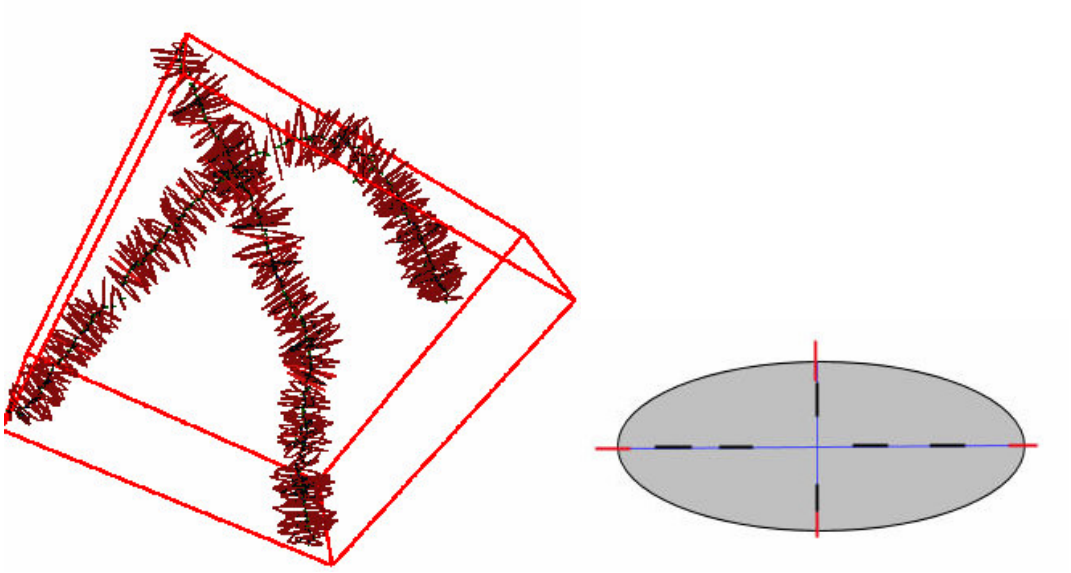
7.1. Acquisition of neuronal data

The mouse brain tissue is stained with Nissl stain[17]. This staining methods are explained in successive sections below. The stained tissue is then embedded in a hard polymer for sectioning. This embedded tissue is serially sectioned and concurrently scanned using the knife-edge scanning microscope (KESM), a new instrument that makes possible three-dimensional microscopy of large biological specimens [21]. A stack of successive two-dimensional images obtained by scanning the mouse brain tissue constitutes the neuronal volume data set.

7.2. Construction of training sets

Vector tracing, described in Chapter V, is used to construct training sets for the neuronal volume data. The output of vector tracing algorithm is a set of ellipses (Fig. 22a). Each ellipse is defined by a center point, and lengths and orientations of its major and minor axes. Orientations of the major and minor axes correspond to normal and

binormal vectors respectively at that position, while the tangent is aligned along the neurite trajectory. From this output, training sets are extracted in the following way.



(a) **(b)**

Fig. 22. (a) Example of output of vector tracing algorithm. (b) Criteria for labeling edges in an ellipse whose major and minor axes are shown in blue. Interior edges are shown in black and boundary edges in red.

Given the center point of an ellipse, and lengths of its major and minor axes whose orientations are known, we proceed from the center point along each one of those axes towards boundary in two directions, axis direction and the direction opposite to it (Fig. 22b). Then, all edges along a major/minor axis of the ellipse that lie inside the boundary are classified as interior edges and are placed in respective adjacency classes. Boundary edges that have one vertex inside and one vertex outside are also identified. Exterior edges are obtained by selecting some blocks of the sample volume data that do not belong to any ellipse in the set of ellipses obtained by vector tracing. Hence, we obtain training set that has edges belonging to interior, boundary, and exterior edges.

7.3. Nissl-stained tissue

Nissl stains only neuronal cell bodies and their nuclei in the brain tissue. Dendrites and axons are not stained. The whole brain specimen is dyed with thionine, staining the cytoplasmic RNA of all neurons, as well as the DNA in all cell bodies [17].

A sample two-dimensional image from the set of Nissl-stained images is shown in Fig. 23. These images are obtained by knife-edge scanning microscope using a 40x objective. Each pixel in these images is $0.3\mu m \times 0.3\mu m$. We can observe that fibrous structures are absent in these images, as only cell bodies are stained. This data falls under the category of blobular data, as described in the previous chapter.

Before using these images for PVDS construction, selected parts of these images were compressed and preprocessed (Fig. 24). A high-pass filter is applied on these images, as explained in Section 2 of Chapter IV and the contrast of the images was enhanced. This preprocessing used Adobe Photoshop. Training sets were constructed from the preprocessed images using vector tracing, as described above.

7.3.1. Nucleus-cytoplasm separation in Nissl-stained tissue

In Nissl-stained tissue, all cell bodies are visible. In the images obtained from Nissl-stained tissue, we observed that the nucleus of the cell body is darker than cell cytoplasm. To achieve separation between nucleus and cytoplasm, we introduce further classification of interior edges. Since no boundary edges need to be identified, terms with absolute value of difference between gray-level values are not included in the

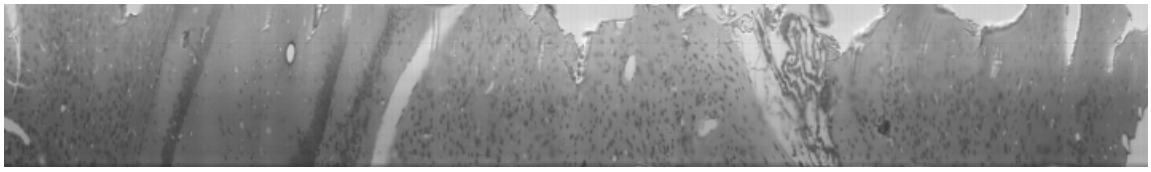


Fig. 23. A sample 2D image of Nissl data obtained at 40x objective.

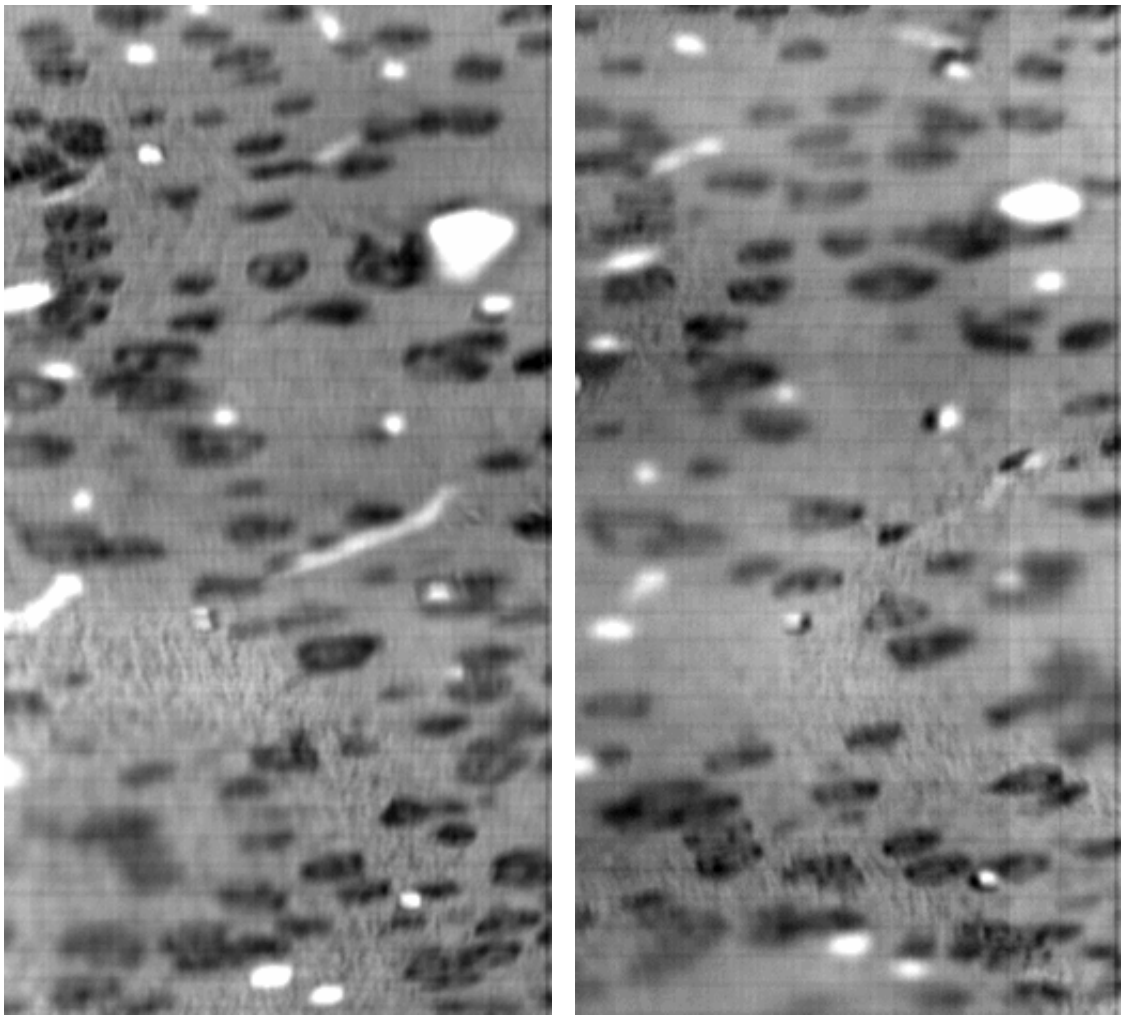


Fig. 24. Sample 2D images of Nissl data after preprocessing.

discriminant functions for interior edge sub-classification. If the gray-level values of interior edge termini used in these functions are normalized by the average gray-level value of the corresponding cell body, then we achieve a better identification of nucleus in each cell. The average gray-level value of each cell can be found by L-block coverings, which are presented in Chapter I. Normalization of gray-level values is left to future work. Discriminant functions for interior edge sub-classification into nucleus and cytoplasm edges are given below for 6-, 8- and 12- adjacencies:

$$y_l^6(g_0, g_l) = w_0 + w_1(g_0 + g_l) + w_2\left(g_0^2 + g_l^2\right) + w_3(g_0 g_l)$$

$$y_l^8(g_0, g_l) = w_0 + w_1(g_0 + g_l) + w_2\left(g_0^2 + g_l^2\right) + w_3(g_0 g_l)$$

$$\begin{aligned} y_l^{12}(g) = {} & w_0 + w_1\left(g_0 + g_l\right) + w_2(g_0^2 + g_l^2) \\ & + w_3\left(g_0 g_l\right) + \\ & w_4\left(g_0 + g_l\right)\left(a_1 + a_2 + a_3 + a_4\right) + \\ & w_5\left(g_0 + g_l\right)\left|a_1 - a_2 + a_3 - a_4\right| + \\ & w_6\left(a_1 + a_2 + a_3 + a_4\right) + w_7\left|a_1 - a_2 + a_3 - a_4\right| + \\ & w_8\left(a_1^2 + a_2^2 + a_3^2 + a_4^2\right) + \\ & w_9\left(a_1 a_2 + a_2 a_3 + a_3 a_4 + a_4 a_1\right) + \\ & w_{10}\left(a_1 a_3 + a_2 a_4\right) + w_{11}\left|a_1^2 - a_2^2 + a_3^2 - a_4^2\right| + \\ & w_{12}\left|a_1 a_2 - a_2 a_3 + a_3 a_4 - a_4 a_1\right| + w_{13}\left|a_1 a_3 - a_2 a_4\right| \end{aligned}$$

Construction of training sets for this classification is done manually at present. More automated methods are left to future work.

CHAPTER VIII

TRAINING OF THE EDGE-LABELING DISCRIMINANT FUNCTIONS

The previous two chapters have discussed the construction of training sets for synthetic and neuronal volume data sets respectively. These training sets can be used to train discriminant functions to label edges as interior, exterior or boundary. Also, edges belonging to the boundary class can further be subdivided into 'entering' and 'exiting' edges. The procedure followed for training these functions is explained in this chapter.

The discriminant functions formulated in this thesis are single-layer perceptrons, and are trained by the perceptron learning algorithm [13]. This algorithm is reviewed in the initial part of this chapter. Training procedures for perceptrons used for two-class classification, as well as multi-class classification, are discussed. Omission of terms with negligible weights from the discriminant functions serves to reduce the computational cost of edge labeling. This procedure is discussed in the final part of this chapter.

8.1. Single-layer perceptron training

As mentioned above, discriminant functions formulated for edge labeling are single-layer perceptrons. A perceptron is a neural network model proposed by Rosenblatt [22]. It consists of a single neuron with adjustable synaptic weights and bias. The signal-flow graph of a general perceptron is shown in Fig. 25 [13]. In the signal-flow graph shown, the 'induced local field' of a neuron is the sum of bias and all signals that enter the neuron weighted by their corresponding synaptic weights. The 'induced local field', v, is given by:

$$v = \sum_{i=1}^{m} w_i x_i + b \, .$$

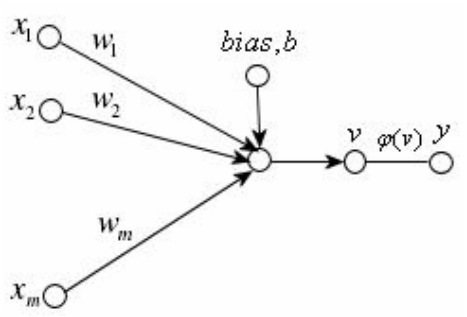


Fig. 25. Signal-flow graph of a single-layer perceptron.

We consider *b* as the synaptic weight w_0 for a link from an input node, x_0, to the neuron, where $w_0 = b$ and $x_0 = 1$. The induced local field of the neuron can be written as:

$$v = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w}^T \boldsymbol{x} .$$

Here $\boldsymbol{w}^T \boldsymbol{x}$ represents the inner product of weight and input vectors, viewed in the m+1-dimensional real space, R^{m+1}. The activation function, $\varphi(v)$, is a threshold function in our case. This perceptron is a two-class classifier, which assigns an input vector $\boldsymbol{x} = [x_0, x_1, ..., x_m]^T$ into class 1 or class 2, based on the value of the threshold function of the induced local field. That is, $\boldsymbol{x}$ is assigned to class 1 if v < 0 or to class 2 if $v \geq 0$.

This two-class perceptron model applies to the discriminant function, as derived in Chapter IV, used to classify boundary edges into entering and exiting edge classes. The corresponding signal-flow graph is shown in Fig. 26. Discriminant functions used to classify edges into interior, exterior and boundary classes, which are derived in Chapter III, are multi-class perceptrons, as they classify edges into more than two classes. The perceptron learning algorithms used to train a two-class perceptron and a multi-class perceptron are presented below.

Let $\mathbf{T}_1$ be the set of vectors $\{ \boldsymbol{x}^1(1), \boldsymbol{x}^1(2), ... \}$ belonging to class C_1 and $\mathbf{T}_2$ be the set consisting of vectors $\{ \boldsymbol{x}^2(1), \boldsymbol{x}^2(2), ... \}$ belonging to class C_2. The union of $\mathbf{T}_1$ and $\mathbf{T}_2$ forms the training set.

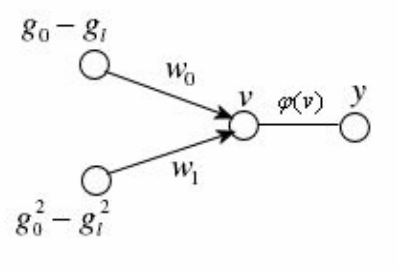


Fig. 26. Signal-flow graph of perceptron corresponding to the discriminant function that classifies boundary edges.

Let $\boldsymbol{x}(n)$ be an input vector used to train the weight vector at the nth step. Let the weight vector at the n[th] step be represented by $\boldsymbol{w}(n)$. Let $\boldsymbol{d}(n)$ denote the desired response of the neuron for a given input, $\boldsymbol{x}(n)$ and, let $\boldsymbol{y}(n)$ be the actual response of the neuron. An error signal, $\boldsymbol{e}(n)$ at the output of the neuron at iteration n is defined by [13]:

$$\boldsymbol{e}(n) = d(n) - y(n)$$

A cost function, $E(n)$, defined in terms of this error signal is as follows:

$$E(n) = \frac{1}{2} e_k^2(n)$$

In our case, the goal of training is to minimize the average error energy over all input vectors in the training set. Let N denote the total number of input vectors present in the training set. Then, average squared error energy is given as follows [13]:

$$E_{avg}(n) = \frac{1}{N} \sum_{n=1}^{N} E(n)$$

The above function is used as an index of performance while training the perceptron.

The algorithm for training the two-class perceptron is explained here [13]. Let us start with a weight vector initialized to zero vector. Then, the weight vector is trained according to the following rules:

i. Weight vector is unchanged if it correctly classifies the current input vector. That is,

$\boldsymbol{w}(n+1) = \boldsymbol{w}(n)$ if $\boldsymbol{w}(n)^T \boldsymbol{x}(n) > 0$ and $\boldsymbol{x}(n)$ belongs to class C_1

$\boldsymbol{w}(n+1) = \boldsymbol{w}(n)$ if $\boldsymbol{w}(n)^T \boldsymbol{x}(n) \leq 0$ and $\boldsymbol{x}(n)$ belongs to class C_2

ii. If the input vector is incorrectly classified, then the weight vector is updated as given below:

$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta(n)\,\boldsymbol{x}(n)$ if $\boldsymbol{w}(n)^T\,\boldsymbol{x}(n) > 0$ and $\boldsymbol{x}(n)$ belongs to class C_2

$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta(n)\,\boldsymbol{x}(n)$ if $\boldsymbol{w}(n)^T\,\boldsymbol{x}(n) \leq 0$ and $\boldsymbol{x}(n)$ belongs to class C_1

Here, $\eta(n)$ is a learning-rate parameter at iteration n and its value gradually decreases till a small value (≈ 0.1) with increasing number of iterations while training the discriminant function. A variable learning-rate is employed to ensure that weight vector converges in the mean, since the input vectors are not perfectly linearly separable in our case. The training is continued till the weight values converge and reach a steady state. Using the algorithm given above, the discriminant function can be trained to classify boundary edges into 'entering' and 'exiting' edge classes.

8.2. Multi-class perceptron learning

As the discriminant functions derived in Chapter III classify edges into three classes, we require a multi-class perceptron model [16]. Multiple perceptrons are employed in parallel as shown in Fig. 27. Each perceptron is similar to the single-layer general perceptron discussed above. The number of such perceptrons required in a multi-output perceptron is equal to the total number of classes into which vectors in the training set are divided. Input vectors are classified based on the output of all these perceptrons. In our implementation, a function that finds the maximum of all output values is used. An edge is classified into the class that corresponds to the perceptron that gives the maximum output value (winner-takes-all, WTA). Hence, the multi-output perceptron is trained such that when an input vector of gray-level values corresponding to an edge is presented, the edge is assigned to the class of the perceptron giving maximum response.

The set of perceptrons used for multiple-class classification is trained by reinforced and anti-reinforced learning rules [16]. Let $C_1, C_2, \ldots, C_m$ be a set of m classes

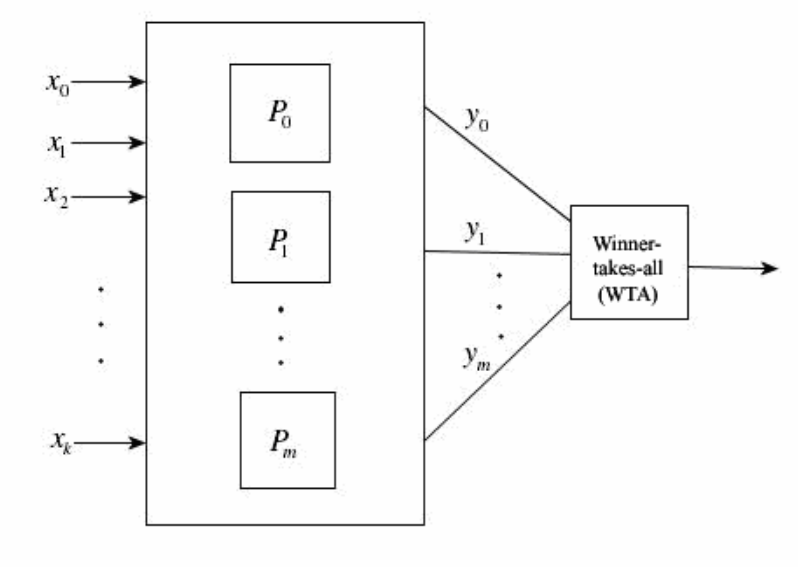


Fig. 27. Multi-output perceptron used for multiple class classification.

and $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ..., $\boldsymbol{w}_m$ be weight vectors of the perceptrons corresponding to these m classes respectively. If $\boldsymbol{x}$ represents an input vector that belongs to class C_i, then the goal of training is to produce weight vectors such that $\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}$ for all $\mathrm{i} \neq \mathrm{j}$. But, as mentioned in the case of two-class perceptron, since input vectors are in general not linearly separable, finding a weight vector that classifies every input vector in the training set without error is not possible. Hence, we train the functions such that the average error energy, as defined above, is minimized.

Rules of the training are given below. Same notation is followed for input vectors and weight vectors at nth step as in the case of two-class perceptron.

i. Weight vectors are unchanged if they correctly classify a given input vector $\boldsymbol{x}(\mathrm{n})$.

ii. If $\boldsymbol{x}(\mathrm{n})$ belongs to class C_i, and $\boldsymbol{w}_j(\mathrm{n})^T \boldsymbol{x}(\mathrm{n}) > \boldsymbol{w}_i(\mathrm{n})^T \boldsymbol{x}(\mathrm{n})$ for all $\mathrm{j} \neq \mathrm{i}$, which means that input is misclassified into class C_j, then the following updates are performed:

Reinforced learning: $\boldsymbol{w}_i(n+1) = \boldsymbol{w}_i(n) + \eta(n)\,\boldsymbol{x}(\mathrm{n})$

Anti-reinforced learning: $\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) - \eta(n)\boldsymbol{x}(n)$

The other weight vectors $\boldsymbol{w}_k(n)$ for k = 1, 2, …, m and $k \neq i$ and $k \neq j$ are unchanged. As mentioned before, $\eta(n)$ is a variable learning-rate parameter.

A signal-flow graph of a perceptron representing the discriminant function for 6-adjacency is shown in Fig. 28. Only a single perceptron is shown in the figure. Three such perceptrons are trained and used together to classify edges into interior, exterior and boundary classes. Similar signal-flow graphs can be obtained for discriminant functions for 8- and 12- adjacencies.

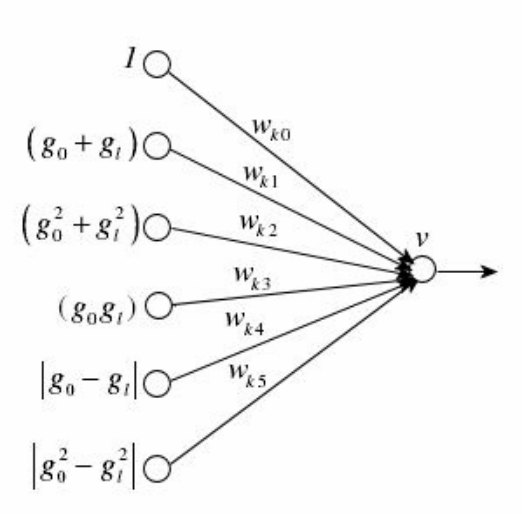


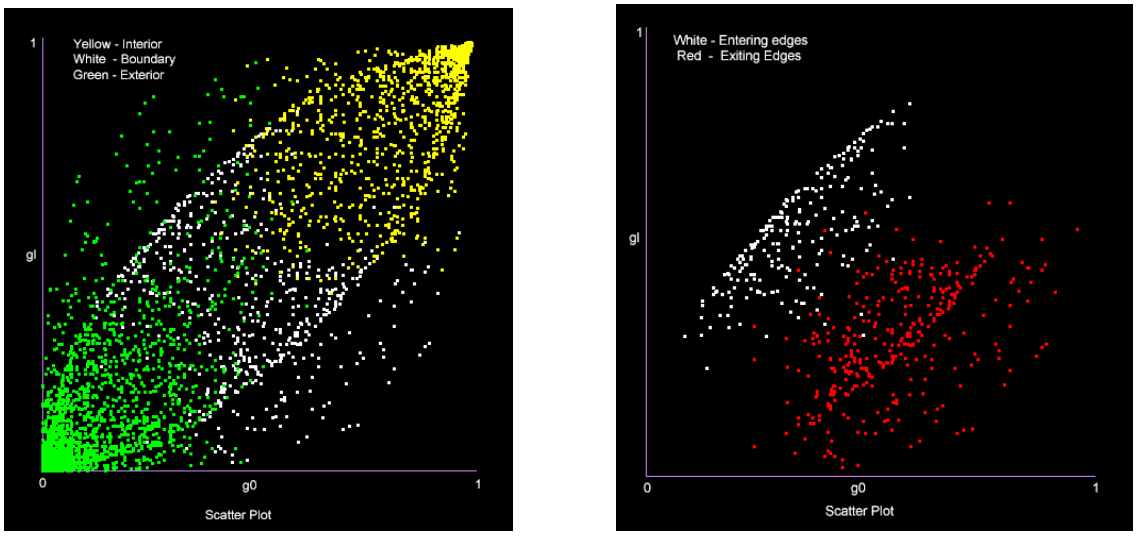
Fig. 28. Signal-flow graph model of perceptron for 6-adjacency discriminant function.

8.3. Training the discriminant functions

Two-dimensional scatter plots of the terminal vertices of an edge, g0 and gl, are shown in Fig. 29 and Fig. 30. Fig. 29a and Fig. 29b correspond to training sets for synthetic volume data. Fig. 30 corresponds to a training set for Nissl data. These plots show the distribution of gray-level values based solely on their base-line segmentation. We can observe that these classes are not linearly separable and a quadratic function might help in classifying the edges.

In Fig. 30, we observe there is no perfect separation between the interior and exterior, due to presence of cell bodies in Nissl images that have high average gray-level values. Those cell bodies are light-gray in color and close to the gray-level of exterior edges. Also, it can be observed that some of the boundary edges have less difference between g_0 and g_l. This occurs because for some cells gray-level values from cell interior to exterior varies smoothly over several pixel displacements.

The graph shown in Fig. 31 proves the convergence of weight parameters of discriminant functions during training. Three terms in the discriminant function for 6-adjacency are selected and their convergence is displayed graphically. Results presented in Chapter IX prove that weight values have converged such that classification due to discriminant functions is similar to the classification in training sets.



(a) **(b)**

Fig. 29. Image showing normalized gray-level values of the edge termini from the training set for synthetic data. The X-axis represents 'g0' with range 0-1 (corresponding to 0-255) and Y-axis represents 'gl' with the same range. (a) Points corresponding to edges belonging to three different classes are shown in three different colors: interior edges in yellow, boundary edges in white and exterior edges in green. (b) Points corresponding to boundary edges belonging to two different classes: entering edges in white and exiting edges in red.

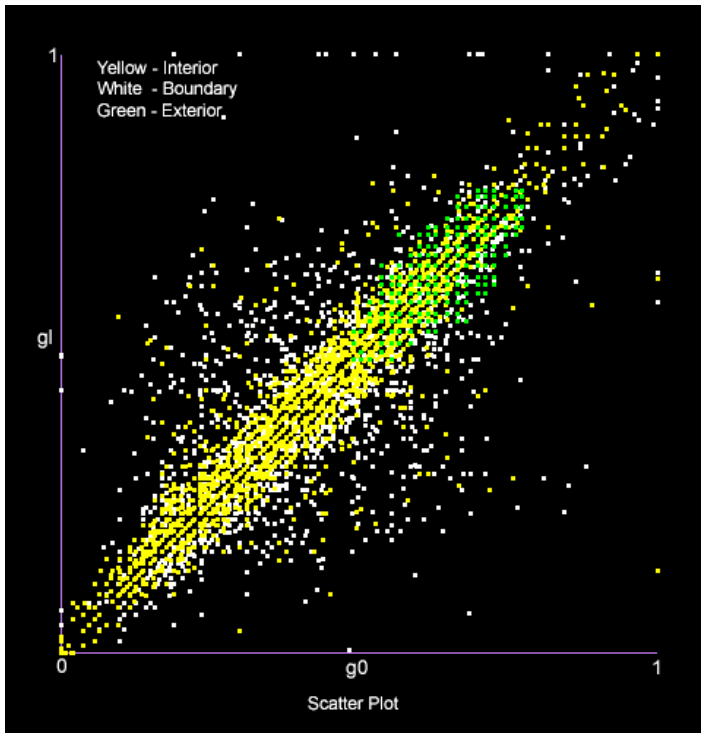
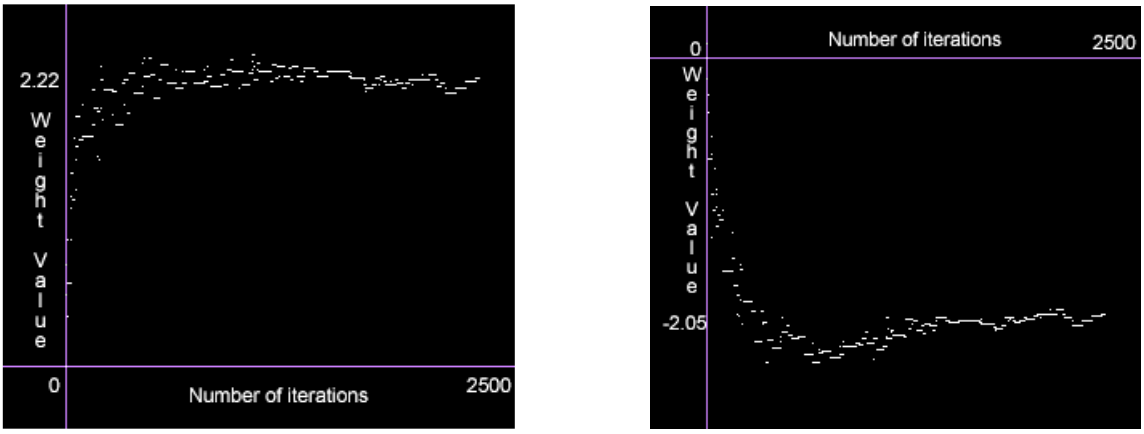


Fig. 30. Image showing normalized gray-level values of the edge termini from the training set for Nissl data. The X-axis represents 'g0' with range 0-1 (corresponding to 0-255) and Y-axis represents 'gl' with the same range. Points corresponding to edges belonging to three different classes are shown in three different colors: interior edges in yellow, boundary edges in white and exterior edges in green.



(a) **(b)**

Fig. 31. Graph showing the convergence in mean of weights of selected terms in the discriminant function for 6-adjacency for synthetic data. (a) and (b) Terms $g_0 g_1$ and $\left| g_0^2 - g_1^2 \right|$ in the function corresponding to interior edge class.

8.4. Validating feature selection in the multi-term discriminant functions and minimizing edge-labeling cost

Discriminant functions derived for interior/exterior edge-labeling consist of multiple terms. The discriminant functions for 6- and 8- adjacencies derived in Chapter III have five terms apart from bias. The discriminant function for 12-adjacency has seventeen terms apart from bias. These functions are initially formulated to include all terms that are symmetrical in the input gray-level values.

In our implementation, gray-level values are normalized to lie in between 0 and 1, before using them in discriminant functions. Hence, insignificant terms in the trained discriminant functions can be identified by examining their weight values. The terms whose weights have negligible values are then deleted from the discriminant functions. As these terms do not play a major role in classification of edges, deletion of these terms helps in minimizing the cost of edge-labeling without affecting the results.

Weight values of discriminant functions for 6-adjacency trained on Nissl data are shown in Table 6. From this table, we observe that the discriminant function for boundary class has comparatively high weights for the terms with gray-level value difference, which is as expected. The term with absolute difference of gray-level values

TABLE 6

Final weights of discriminant functions for 6-adjacency trained on Nissl data

Weights → / Class ↓	w_{k0}	w_{k1} $(g_0 + g_l)$	w_{k2} $\left(g_0^2 + g_l^2\right)$	w_{k3} $(g_0 g_l)$	w_{k4} $\lvert g_0 - g_l \rvert$	w_{k5} $\left\lvert g_0^2 - g_l^2 \right\rvert$
Interior (k = 0)	1.448	-1.815	-0.689	-0.389	0.087	0.246
Exterior (k = 1)	-3.723	2.801	1.294	0.784	-2.419	-2.337
Boundary (k = 2)	0.608	-0.986	-0.605	-0.395	2.332	2.091

is less significant in the function for interior class when compared to other terms. In the function for exterior class, almost all the terms look significant.

CHAPTER IX

VISUALIZATION AND VALIDATION OF PVDS

Construction of training sets for synthetic and real-world neuronal data sets have been presented in Chapters VI and VII respectively. Algorithms for training the discriminant functions for edge-labeling were presented in Chapter IX. This chapter summarizes the results obtained using the trained discriminant functions. PVDS construction for different volume data sets presented in previous chapter is displayed in the sections below. Finally criteria for evaluating the PVDS-induced segmentation against a base-line segmentation are discussed.

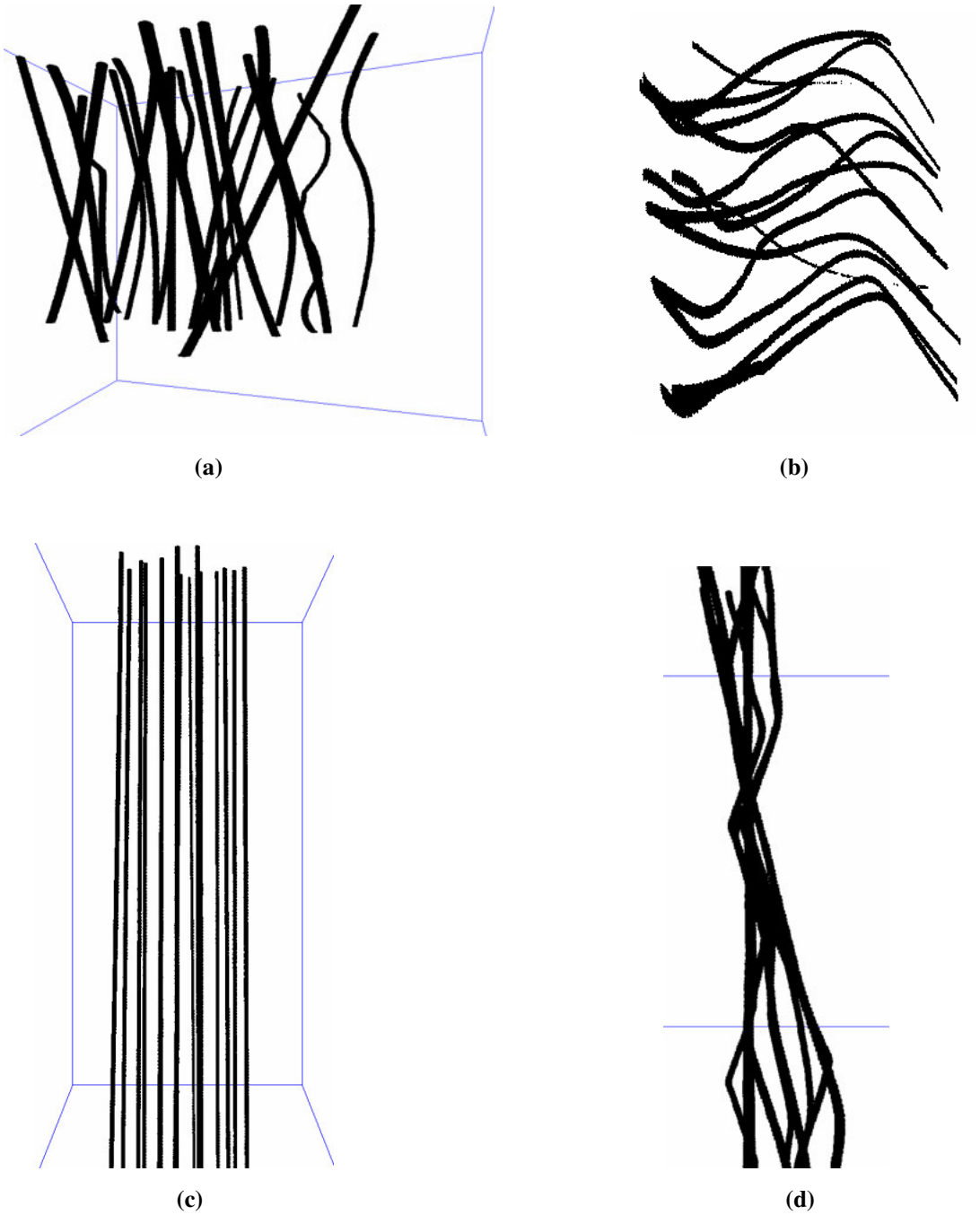
9.1. Global and local views of the segmentation

Global views of the PVDS obtained for filamentary and blobular synthetic data sets are shown in Fig. 32 and Fig. 33 respectively. Two-dimensional and three-dimensional local views of selected parts of PVDS are shown in Fig. 35 respectively. These images were obtained using OpenGL [26].

Global and local views of PVDS obtained for Nissl data are shown in Fig. 35. In the results for Nissl data, further classification of interior edges is shown. In the figures, blue colored edges represent nuclei. As gray-level values are not normalized with respect to average gray-level value of the corresponding cell body, identification of nuclei in different cells is not similar, as discussed in Chapter VII.

9.2. Scatter plots for PVDS

The scatter plot showing the classification of edges by PVDS for one of the synthetic volume data sets is shown in Fig. 36. We observe that the classification of edges by the trained discriminant functions is nearly as good as the classification in the training set shown in Fig. 29.

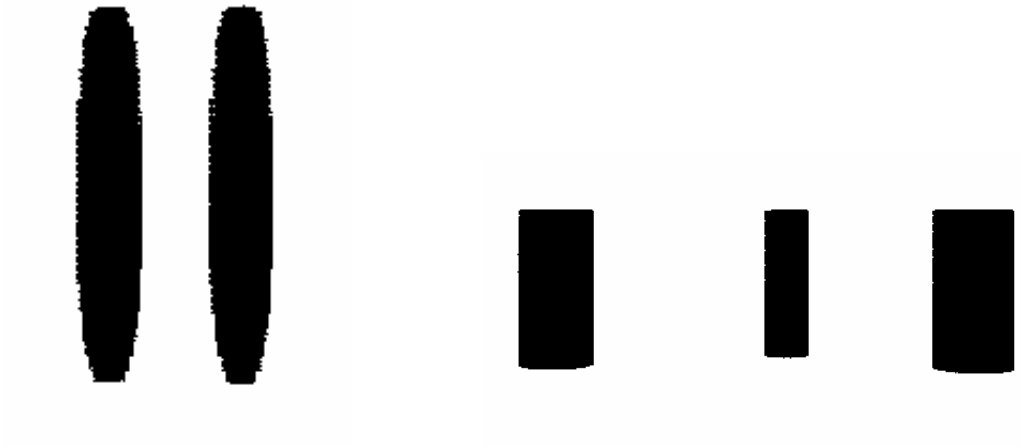


(a) **(b)**

(c) **(d)**

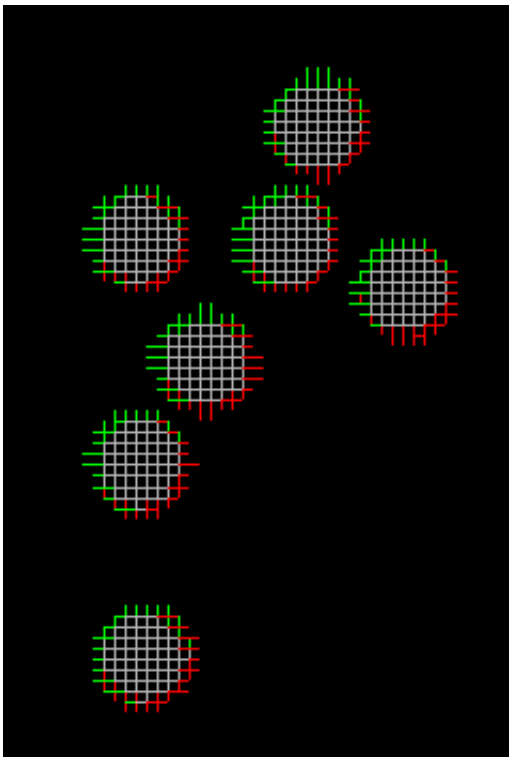
Fig. 32. Global view of PVDS for filamentary synthetic data sets. (a) Fine fibers. (b) A neuropil-like mat of fine fibers. (c) Parallel fibers. (d) Fibers with branching.

(a)

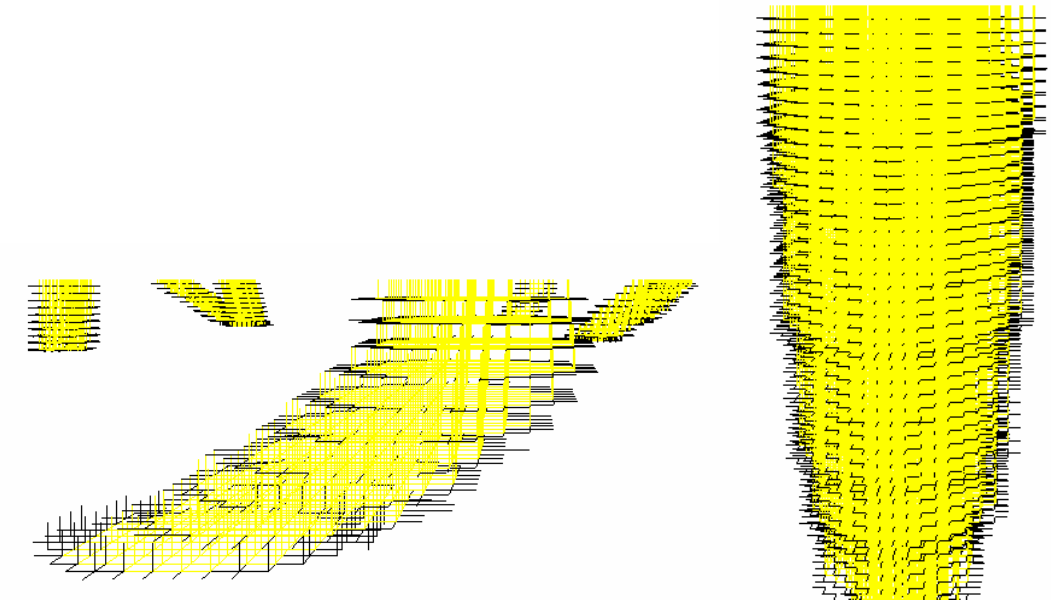


(b) **(c)**

Fig. 33. Global view of PVDS for synthetic data sets simulating blob data. (a) Spheres. (b) Ellipsoids. (c) Cylinders.

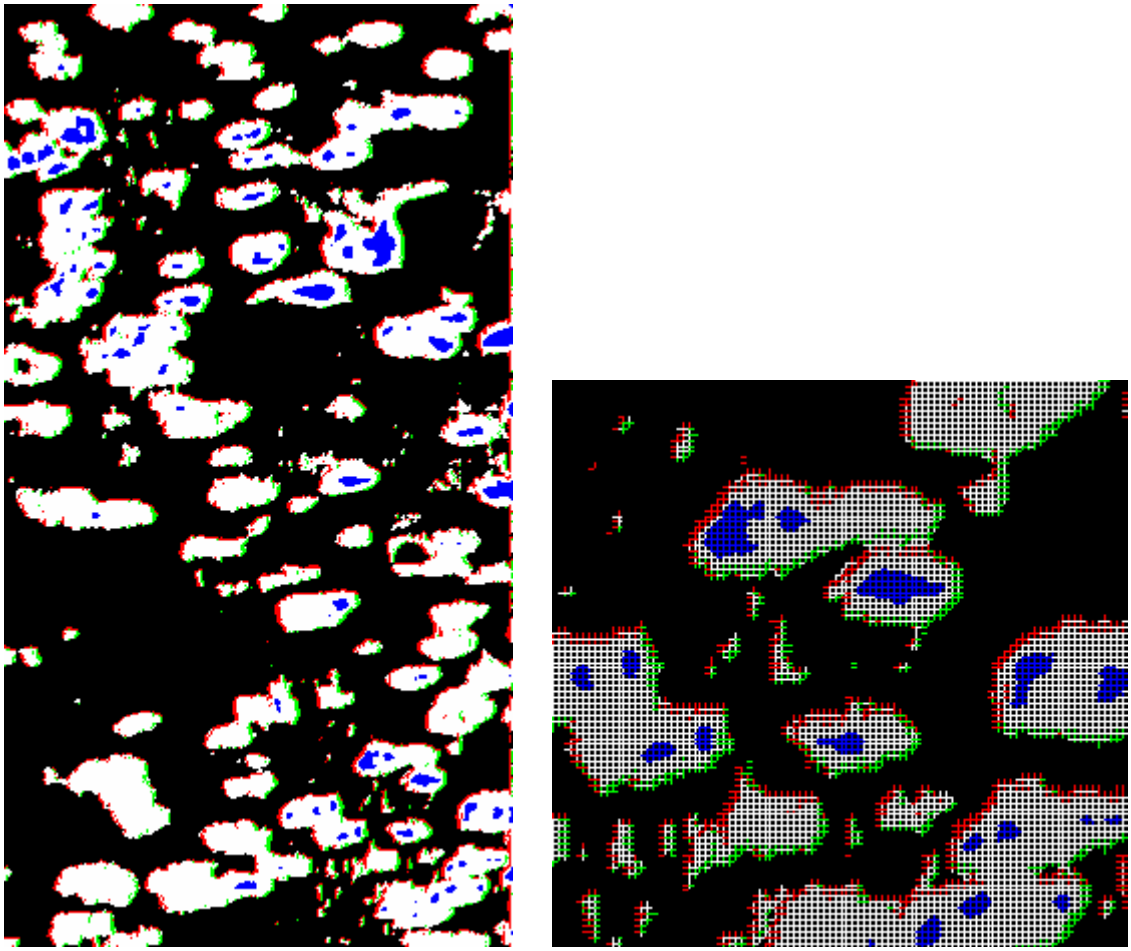


(a)



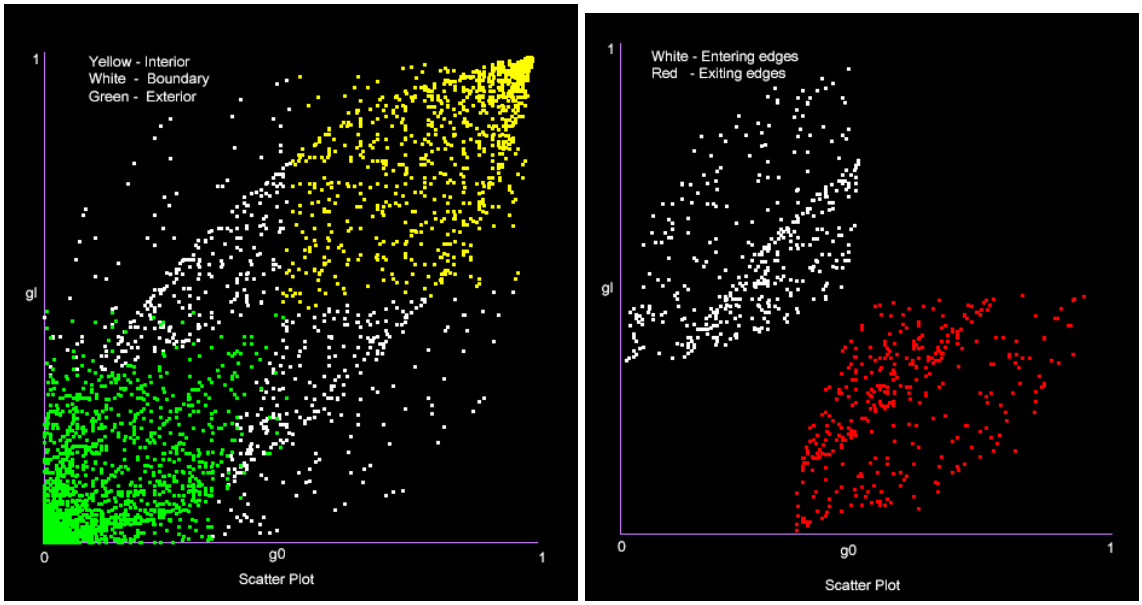
(b) **(c)**

Fig. 34. Local views of PVDS for a synthetic data set. (a) 2D view: Interior edges in gray, entering edges in red and exiting edges in green. (b) 3D view of filamentary data: Interior edges in yellow and boundary edges in black. (c) 3D view of blobular data: Interior edges in yellow and boundary edges in black.



(a) (b)

Fig. 35. 2D views of PVDS for Nissl-stained data. Interior edges are shown in white and blue with blue representing nuclei of cell bodies, entering edges in red and exiting edges in green. (a) Global view of PVDS for an image from Nissl-stained data (This result corresponds to the image shown on the left side in Fig. 24). (b) A local view of PVDS.



(a) **(b)**

Fig. 36. Image showing gray-level values of the edge termini from the PVDS. The X-axis represents 'g0' with range 0-1 (corresponding to 0-255) and Y-axis represents 'gl' with the same range. (a) Points corresponding to edges belonging to three different classes are shown in three different colors: Interior edges in yellow, boundary edges in white and exterior edges in green. (b) Points corresponding to boundary edges belonging to two different classes: Entering edges in white and exiting edges in red.

Similar scatter plot for PVDS obtained from Nissl data is shown in Fig. 37. We observe that the classification of edges between interior and exterior classes resembles the classification shown in training set in Fig. 30, though the training set has more overlap of interior edges with exterior ones. There is a clear separation between edges belonging to different classes in the result shown, as the discriminant functions are limited to be quadratic, as explained in Chapter III. Boundary edge classification is observed to be good.

9.3. Criteria for evaluation of polymerization-induced segmentation

Except for synthetic volume data sets, there exists no absolute criteria for real-world biological data for what constitutes a "good segmentation". We can resort only to relative criteria that evaluate how the obtained segmentation compares to a given base-

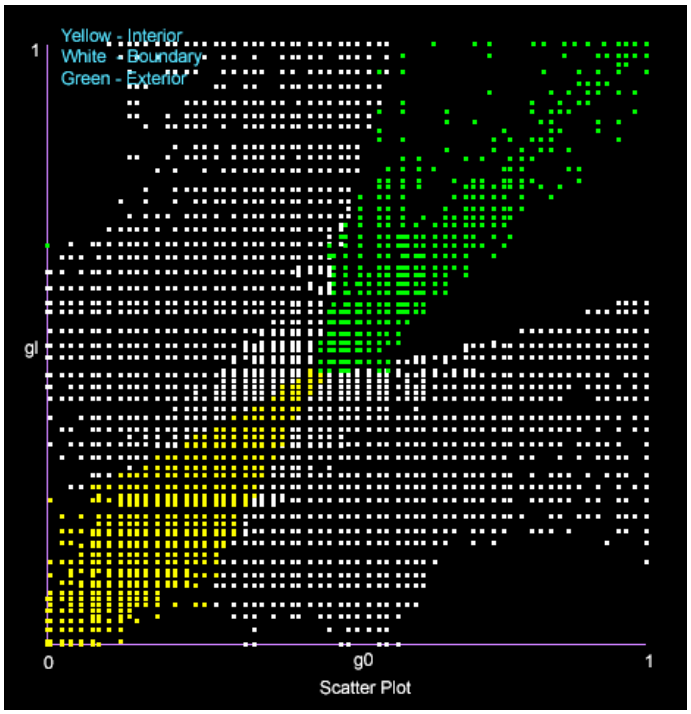


Fig. 37. Image showing normalized gray-level values of the edge termini from the training set for Nissl data. The X-axis represents 'g0' with range 0-1 (corresponding to 0-255) and Y-axis represents 'gl' with the same range. Points corresponding to edges belonging to three different classes are shown in three different colors: Interior edges in yellow, boundary edges in white and exterior edges in green.

line segmentation. A measure of mutual information between base-line and PVDS-induced segmentations can be used as criterion for evaluation.

Additional criteria specific to neuronal data that can be used to evaluate polymerized-induced segmentation Vs. base-line segmentation are given below:

- Ability to separate juxtaposed neurons
- Tracing of faint and gap-ridden axonal tracing
- Coping with neurite branching

9.4. User interface for viewing agreement/disagreement between base-line and polymerization-induced segmentations

Based on experience gained in preparing this thesis, I believe the following user interfaces will be helpful for PVDS construction:

- Display of the scatter plot for the training set and the scatter plot for the PVDS labeled set side-by-side and an interface that flags misclassified points.
- Display of overlap of scatter plots of training sets and PVDS labeled sets with one of them rendered semi-transparent. This helps in identifying misclassified points and those points can be flagged for editing.
- Display of overlap of original image and PVDS image and/or overlap of training image and PVDS image with one of them rendered semi-transparent. This helps in identifying segmentations of the images that do not match.
- Successive display of a set of PVDS images, one after another, to visualize the continuity of data between successive images. Successive image would be displayed after each click by the user.
- Visualization of two-dimensional and three-dimensional views of PVDS with features like zoom-in, zoom-out, and selection of various image parts for added attention.

The implementation of these user interfaces will be left to future work.

CHAPTER X

PVDS CONSTRUCTION GENERALIZED TO BRAVAIS LATTICES

In Chapter I the volume data set was defined over a regular cubic three-dimensional grid, with voxels centered on the vertices of the grid. Then in Chapter II the grid was transformed into a lattice by introducing edges from any vertex to the vertices of its adjacency set. Explicitly, 6-, 8-, and 12-adjacencies were considered. Here we recognize that alternative lattices, called *Bravais lattices*, can be superimposed on the regular cubic lattice. These lattices permit indexing the periodic array using lattice translation vectors with integer coefficients, generalizing integer indexing within the cubic lattice. Furthermore, these Bravais lattices support 6-, 8-, and 12-adjacencies, that is, these sets of edges are readily identified in a Bravais lattice. Accordingly edges within the Bravais lattice can be labeled "active" or "inactive" using discriminant functions, generalizing the analysis of previous chapters of this thesis. In summary, a Bravais lattice allows a different visualization of the same volume data set, but one emphasizing a different local connectivity.

We proceed below by first defining Bravais lattices and Bravais nets, followed by defining the 6-, 8-, and 12-adjacency sets of a vertex in a Bravais lattice. Next we enumerate all Bravais nets, and then Bravais lattices. Finally, as an illustration of the potency of generalizing PVDS construction to Bravais lattices and nets, we consider the construction and enumeration of perfectly planar templates, all of whose vertices (voxels) lie precisely in a common plane. Further applications of generalizing PVDS construction is deferred to future work.

10.1. Bravais lattices and Bravais nets

Bravais lattice: Let us choose three vectors **a**, **b** and **c** in such a way that: |**a**| is the shortest period in the lattice (or one of several equal ones), **b** is the shortest vector not parallel to **a**, and **c** is the shortest one not coplanar to **a** and **b**. If a periodic array has exactly the same arrangement and orientation at positions r and r' such that, r' = r+l**a**+m**b**+n**c** where l,m,n are integers, then the set of points r' defines a lattice (Fig. 38).

The vectors **a**, **b** and **c** are called 'lattice translation vectors'. This lattice is called the *Bravais lattice* of the crystal [24], or in our case, the underlying cubic lattice, to which it belongs.

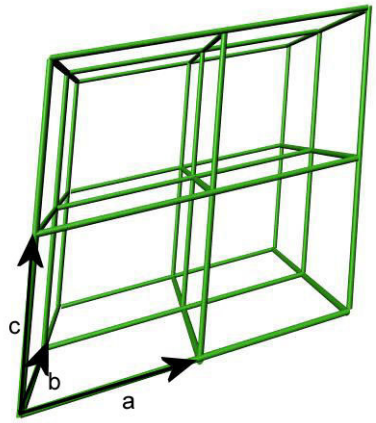


Fig. 38. Bravais lattice with lattice translation vectors a, b and c [24].

Bravais net: A Bravais net is two-dimensional planar lattice with the smallest repeatable unit being the unit cell defined by vectors **a** and **b**, chosen as above for a Bravais lattice. Hence a *Bravais net* is a two-dimensional planar lattice over the cubic grid, which can be expanded into a Bravais lattice by introducing a third non-coplanar lattice vector. Such planar arrays are a natural substrate for the construction of planar templates for kernel functions defined over the volume data set. And example is developed below, where extended planar templates are developed, each class embedded in its own distinct Bravais net.

10.2. Adjacency sets in a Bravais lattice

Given the Bravais lattice vectors **a**, **b**, and **c**, we form the 6-, 8-, and 12-adjacency sets of a vertex r as follows:

6-adjacency: r ± {**a**, **b**, **c**}; face-centered.

8-adjacency: r ± **a** ± **b** ± **c;** vertex-centered**.**

12-adjacency: r ± {**a** ± **b**, **b** ± **c, c** ± **a**}; edge-centered.

For the special case where the Bravais lattice is the simple cubic lattice, these adjacency definitions reduce to the face-, vertex-, and edge-centered definitions respectively of Chapter II.

10.3. Classification of Bravais nets

Bravais nets in a 2*2*2 cube can be classified into five distinct groups. These groups are analogous to the five unique and distinct Bravais nets found in the literature [10]. Other forms of Bravais nets are obtained by various symmetry operations that map any given lattice onto itself. In this case, reflection and rotation operations constitute these symmetry operations. We define each Bravais net in a 2*2*2 cube in terms of a "signature line". This line, along with the center vertex of the cube, defines a planar Bravais net. The five groups of Bravais nets are described below. The magnitudes of two unit vectors forming the plane are defined by |**a**| and |**b**|, and angle between these vectors is given by $\boldsymbol{\gamma}$.

Square net [|a| = |b| and $\gamma = 90°$]: Bravais nets in this group are defined by a signature line formed by joining two edge centered points such that the plane defined by the line along with the center vertex gives rise to a plane parallel to one of the XY-, YZ- or ZX-planes (Fig. 39a).

Rectangular net [|a| ≠ |b| and $\gamma = 90°$]: This group includes diagonal planes. The Bravais nets in this group can be defined by a signature line that forms a corner edge of the 2*2*2 cube (Fig. 39b).

Hexagonal net [|a| = |b| and $\gamma = 60°$]: This group is defined by signature lines that connect two edge-centered vertices that lie on the same face of 2*2*2 cube (Fig. 39c).

Oblique net [|a| ≠ |b| and $\gamma \neq 90°$]: Oblique Bravais nets are obtained by a plane passing through center vertex and the signature line connecting a vertex-centered vertex with an edge-centered vertex (Fig. 39d).

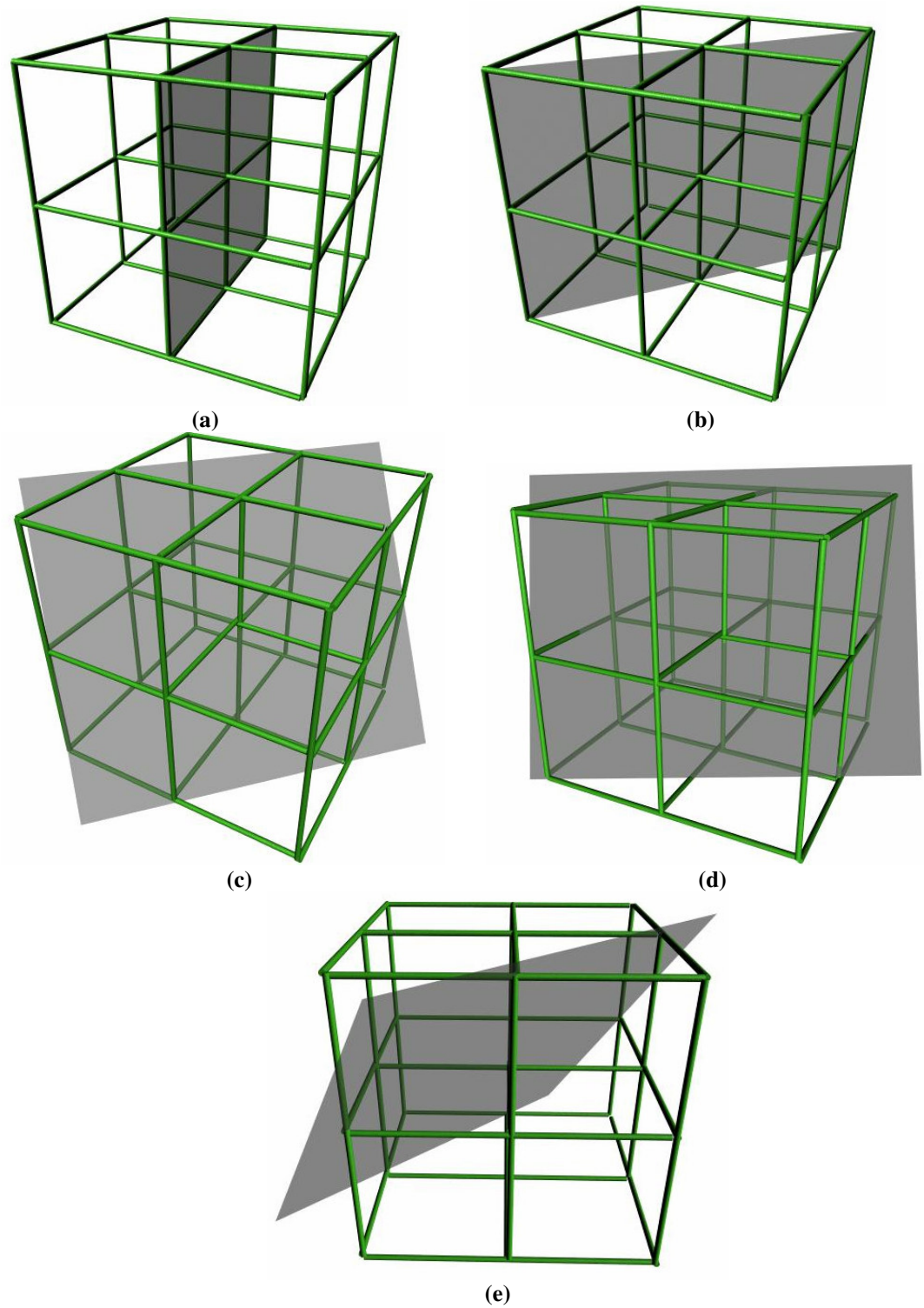


Fig. 39. Examples of Bravais nets. (a) Square net. (b) Rectangular net. (c) Hexagonal net. (d) Oblique net. (e) Centered rectangular net.

Centered rectangular net [|a| ≠ |b| and cos γ=|a|/2|b| (or diamond net) with |a'| = |b'| and γ≠ 90, 60, 120°]: This group is defined by signature lines formed by connecting two edge-centered vertices that do not lie in the same face (Fig. 39e).

10.4. Enumeration of canonical forms of Bravais nets

With the five distinct groups of Bravais nets defined, let us enumerate the total number of canonical Bravais nets that can be obtained. For the sake of enumeration, let us consider the 2*2*2 cube figure shown in Fig. 40. The edges of back faces are not shown for clarity. Among the 26 neighboring vertices of the center vertex, we can see 19 vertices. But, 13 of the 26 vertices can be considered as mirror images of the other 13.

Considering all of them to count the number of Bravais nets will give rise to duplicate planes. Hence, six vertices of the 19 vertices that can be seen, whose mirror images are also included in the set of vertices seen, are not considered in the enumeration. These six vertices are shown in violet.

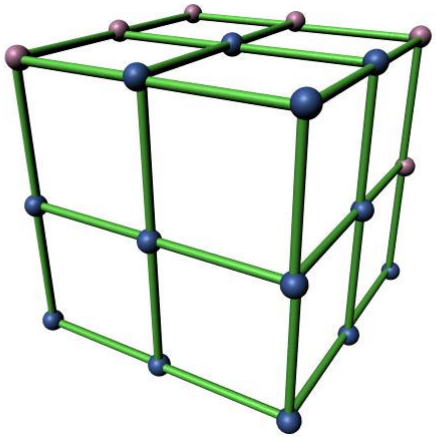


Fig. 40. 2*2*2 cube with only front faces seen. Six vertices which are mirror images of other six seen in the image are shown in violet. Rest of the vertices is shown in blue.

Total number of two-dimensional plane lattices obtained in each group is given below:

Square net: In this group, 3 planes can be obtained, corresponding to XY-, YZ- and ZX-planes. Signature lines of these planes are shown in Fig. 41a.

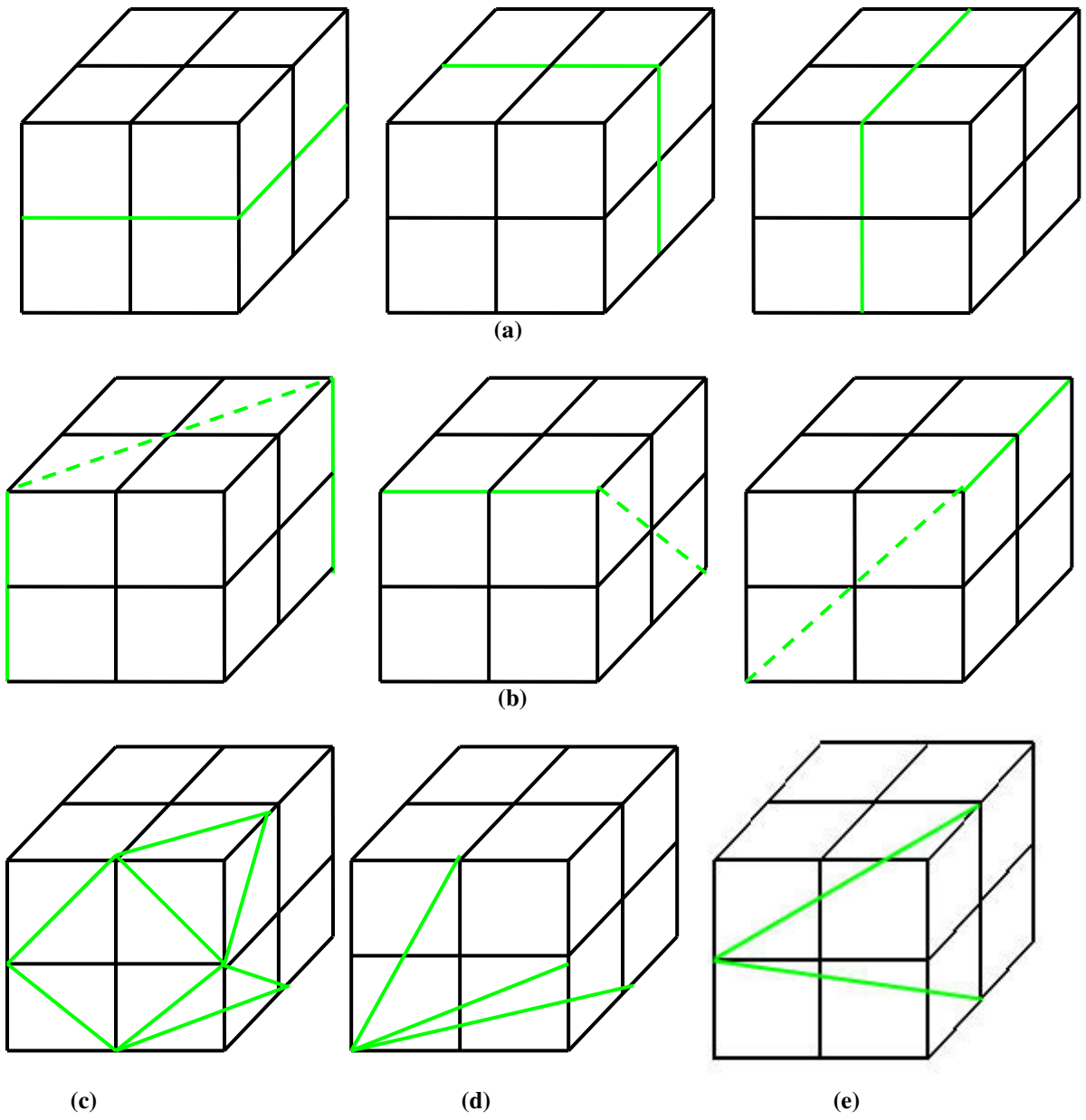


Fig. 41. Canonical forms of Bravais nets. Signature lines are shown in green.

Rectangular net: 6 diagonal Bravais nets are obtained. Three of these are shown in Fig. 41b.

Hexagonal net: 8 corner planes whose signature lines are shown in Fig. 41c are obtained.

Oblique net: 12 vertex oblique planes are present. Signatures for three of these are seen in Fig. 41d.

Centered rectangular net: 4 planes are obtained. Signatures of two of these are seen in Fig. 41e.

Thus, a total of 33 canonical forms of Bravais nets are obtained.

10.4.1. Proof that all possible planes are considered

Here, let us prove that with the five groups defined constitute all possible Bravais nets in a 2*2*2 cube.

With 13 vertices, we can define the "signature" of a plane in 78 (13*12/2) ways. We shall prove that the five groups considered include all these possibilities as shown in Table 7.

TABLE 7

Number of planes and Bravais nets in five different groups

Group	Number of Bravais nets	Possible ways of obtaining	Total number of planes that can be obtained
Square net	3	6	18
Rectangular net	6	6	36
Hexagonal net	8	1	8
Oblique planes	12	1	12
Centered rectangular net	4	1	4

Thus, total number of planes in these five groups sum to 78, which is equal to the number of Bravais nets possible. This proves that no Bravais net is omitted in the above enumeration.

10.5. Enumeration of Bravais lattices

A Bravais lattice is defined by a tetrahedron defined by the center plus 3 admissible surface points such that the four points do not lie in a common Bravais net, as seen in Fig. 42. Bravais lattices are enumerated by considering the number of such sets of three vertices or triplets on the 2*2*2 cube. Each such tetrahedron gives rise to a Bravais lattice and is unique. This is because the plane containing these three points is unique.

The number of distinct triplets that can be obtained from 13 vertices is 13*12*11 / 3*2 = 286. But, some of these triplets do not give rise to a tetrahedron as they might be coplanar with the center vertex. Hence, eliminating collinear triplets from the total number of possible three vertices set, we get the total number of Bravais lattices.

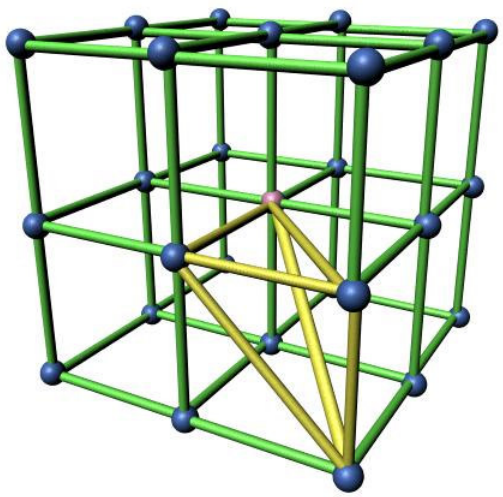


Fig. 42. Bravais lattice defined by a tetrahedron containing the center vertex and three non-collinear vertices on the 2*2*2 cube.

Collinear triplets are counted for each group of Bravais net described above. For square net group, there are 12 such triplets as there are 4 triplets in three ways. For rectangular net group, since there are 6 sets of 4 collinear triplets, we obtain 24 collinear triplets. Thus, eliminating these 36 collinear triplets, we obtain 250 Bravais lattices.

10.6. Extended templates for edge-labeling

Segmentation of the volume data set has been based in Chapters II-IV on classifying individual edges as members of the interior, exterior, and boundary classes. We considered one edge at a time to see if it is active or not (for each class). In this section, the templates that aid us in segmentation are extended to include a larger number of edges. These extended templates can then tiled to form matched filters for boundary detection, analogous to those discussed in Chapter V [1].

A two-edge pattern consists of two vectors, **c** and **d**, which are restricted to be non-parallel and form an acute angle: **c.d** > 0. Each two-edge pattern can be iterated to form a 'ray', which can be thought of as a jagged line. Each ray can be associated with a direction, called the 'orientation of the ray'. For this purpose, the centers of the fixed edges of the repeated patterns are joined to form a line and the direction of that line is considered as the ray orientation. Below, a set of basic two-edged patterns are developed, and the spatial orientation of these rays is exhibited.

Vectors **c** and **d** define a plane, and by translation in this plane, a two-dimensional net. This net, in general, is not a Bravais net, but rather is a sublattice of a Bravais net.

A segment of a ray can be translated in the plane. By this means, extended templates can formed by iterations of two-edge patterns as explained below. These extended templates can then form the basis for matched filters used in vector tracing of neurites.

10.6.1. Two-edge patterns

As mentioned earlier, two-edge patterns are used to form extended templates. One of these two edges can be labeled the 'fixed edge' and the other one 'winged edge'. We develop the patterns by retaining the fixed edge in one constant position and changing the positions of the winged edge to all valid positions. In the three-dimensional grid as described in Chapter II, the length of the fixed edge can be $1, \sqrt{2}$, or $\sqrt{3}$. The positions of the winged edge are explored for fixed edges of each of these lengths.

When fixed edge is of length 1, it can be oriented in three directions, along the X-, Y- or Z-axes, respectively. Let us consider the case where it is oriented along X-axis, as shown in Fig. 43. To the right vertex in the Fig. 43, we can add winged edges in 25 ways. However we shall consider only winged-edge positions which give rise to two-edged patterns that approximate a smooth boundary in real-world volume data set. Using this basis, the 8 neighbors that are in the left perpendicular plane to the fixed edge are not considered. Similarly, the neighbors in the plane on which right vertex lies and that are perpendicular to fixed edge are eliminated for the same reason. The remaining nine neighboring vertices give rise to nine winged-edge positions as shown in Fig. 43.

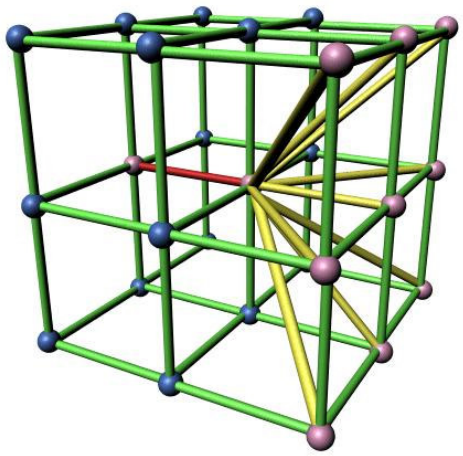


Fig. 43. Fixed edge of length 1 (red color) and nine corresponding positions of the winged edge (yellow color).

Similarly, fixed edge of length 1 aligned along Y and Z axes give rise to 18 more patterns. In total, 27 patterns are obtained for a fixed edge of length 1.

When fixed edge is of length $\sqrt{2}$, the positions of the winged edge are shown in Fig. 44. We see that nine positions are obtained for the fixed edge lying in XY-plane. The other positions are eliminated due to same argument as before. Thus, considering fixed edges lying in YZ- and ZX- planes also, we totally obtain 27 winged edge positions in this case.

Similarly, when fixed edge is of length $\sqrt{3}$, we obtain nine positions for winged edge as shown in Fig. 45. This case gives rise to 27 patterns. Thus totally we obtain 81 two-edge patterns.

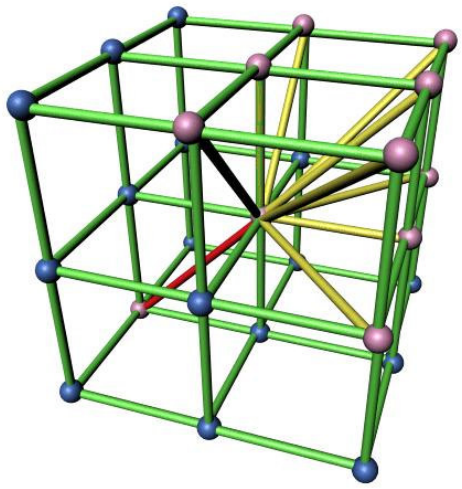


Fig. 44. Fixed edge of length $\sqrt{2}$ (red color) and nine corresponding positions of the winged edge (yellow color).

10.6.2. Forming planar templates from two-edge patterns

When two-edge patterns are given, these can be iterated to form extended planar templates, as shown in Fig. 46. For the template shown in Fig. 46, the orientation of the ray is shown in red color.

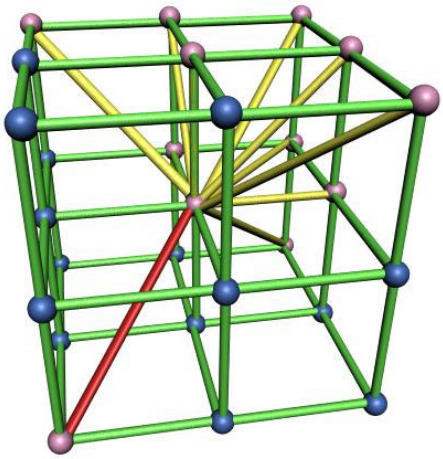


Fig. 45. Fixed edge of length $\sqrt{3}$ (red color) and nine corresponding positions of the winged edge (yellow color).

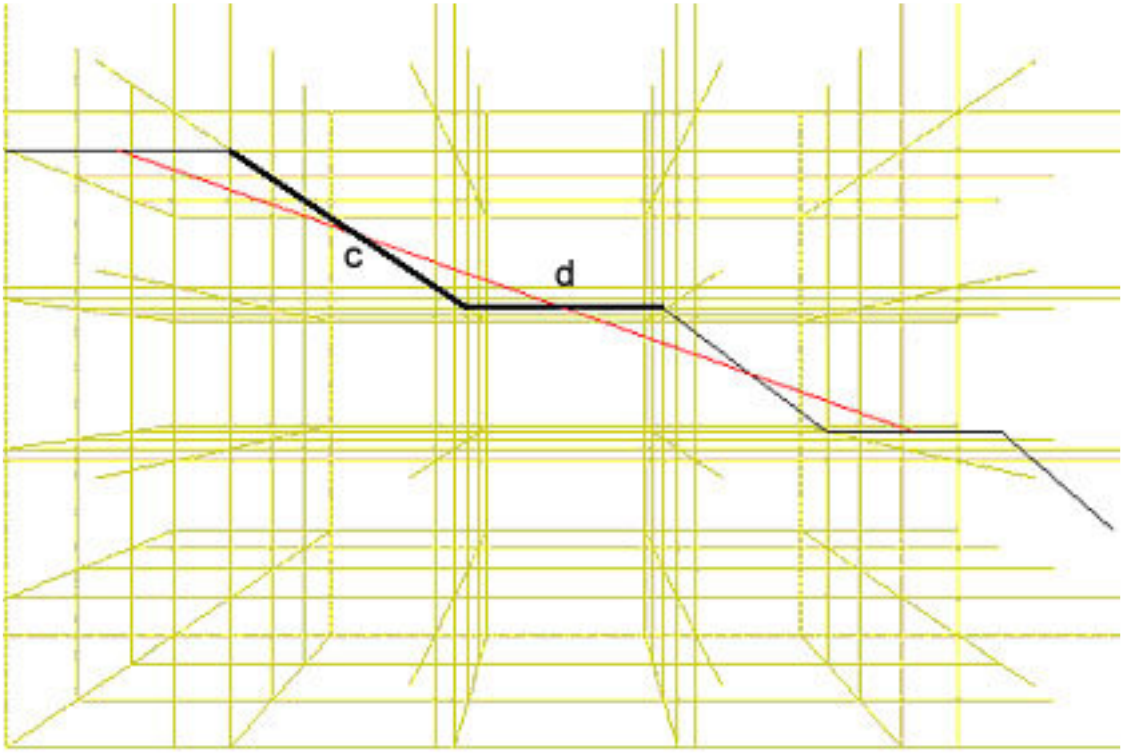


Fig. 46. Example of an extended planar template (thin black lines) formed by iteration of a two-edge pattern(thick black lines). Orientation of ray is shown in red color.

CHAPTER XI

CONCLUSION AND FUTURE WORK

11.1. Summary

A strategy for constructing Polymerized Volume Data Sets (PVDS) using discriminant functions for edge labeling has been presented. Different adjacency classes in a regular three-dimensional grid are described. Discriminant functions for edge-labeling for 6-, 8- and 12-adjacency classes are derived. These discriminant functions label an edge in a volume data set as interior, exterior, or boundary. A discriminant function to further classify boundary edges into entering and exiting edges is also derived. Construction of training sets from a homogeneous sample of volume data and algorithms for training the discriminant functions are discussed.

This strategy is first illustrated on synthetic volume data and then on biological volume data. The process of generating synthetic data that resembles real-world biological data obtained from three-dimensional microscopy is explained. Synthetic data was constructed for filamentary as well as blobular data. Training sets were constructed and discriminant functions for edge-labeling were trained. The strategy worked very well with synthetic data and the results obtained were good.

The strategy was then illustrated on neuronal data. Here, an image stack of 40x Nissl-stained tissue obtained by a knife-edge scanning microscopy was used. Training sets for this volume data were obtained using vector tracing algorithm [1]. Segmentation identified by PVDS are compared with the initial segmentation defined by vector tracing and the results are discussed.

Finally, construction of PVDS is generalized to Bravais lattices. Bravais nets and Bravais lattices are enumerated. Extended templates, an application of Bravais nets, are presented.

11.2. Future work

Future work in this area should include, in approximate order of priority:

- Testing the strategy on volume data sets obtained from other forms of three-dimensional microscopy, such as confocal and two-photon microscopy and from other brain tissue stains, both fluorescent and absorbing.
- Training the discriminant functions using Support Vector Machines (SVMs), a neural network procedure that finds optimal hyperplanes of separation between different classes.
- Deriving discriminant functions for edge-labeling for a volume data set whose voxels are assigned a vector of values (e.g., output of a color camera).
- Using multi-layer perceptrons for edge-labeling and evaluating their efficiency.
- Modifying discriminant functions to exhibit more dependence on shape. One way this can be done is by including more global information, as with the use of extended templates. These larger templates give better noise suppression.
- Developing a good user-interface that helps in visualizing, evaluating, and editing polymerized volume data sets.
- Studying further applications of generalizing PVDS construction to Bravais lattices.

REFERENCES

1. K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner and B. Roysam, "Rapid automated three-dimensional tracing of neurons from confocal image stacks," *IEEE Trans. Inf. Technol. Biomed.*, vol. 6, pp. 171–187, 2002.
2. C.M. Bishop, *Neural Networks for Pattern Recognition*, New York: Oxford University Press, pp. 1-161, 1995.
3. M. Born and E. Wolf, *Principles of Optics*, New York: The Macmillan Company, 1959.
4. U. Braga-Neto, "Connectivity in image processing and analysis: Theory, multiscale extensions and applications", Ph. D. Dissertation, The Johns Hopkins University, Baltimore, 2002.
5. B.P. Burton, B.H. McCormick, R. Torp and J.H. Fallon, "Three-dimensional reconstruction of neuronal forests", *Neurocomputing*, vol. 38-40, pp. 1643-1650, 2001.
6. B. Busse, Personal Communication, Department of Computer Science, Texas A&M University, 2004.
7. S.R. Cajal, N. Swanson and L. W. Swanson, *Histology of the Nervous System*, vol. I, New York: Oxford University Press, 1995.
8. S.R. Cajal, N. Swanson and L. W. Swanson, *Histology of the Nervous System*, vol. II, New York: Oxford University Press, 1995.
9. A. Can, H. Shen, J.N. Turner, H.L. Tanenbaum and B. Roysam, "Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms," *IEEE Trans. Inform.Technol. Biomed.*, vol. 3, pp. 125–138, June 1999.
10. Crystals, symmetry, and diffraction, http://em-outreach.ucsd.edu/web-course/Sec-III.C.1-C.5/Sec-III.C.1-C.5.html, September 2004.

11. P. Doddapaneni, “Segmentation strategies for polymerized volume data sets”, Manuscript, Department of Computer Science, Texas A&M University, College Station.
12. J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice*, (2nd ed.), Reading: Addison-Wesley, 1996.
13. S. Haykin, *Neural Networks, A Comprehensive Foundation*, (2nd ed.), Upper Saddle River: Prentice Hall, 1999.
14. A. Kaufman, *Volume Visualization*, Los Alamitos: IEEE Computer Society Press, 1991.
15. R. Klette and A. Rosenfeld, *Digital Geometry*, San Francisco: Morgan Kaufmann Publishers, 2004.
16. Linear Perceptron Networks, http://www.gc.ssr.upm.es/inves/neural/ann1/supmodel/linperne.htm, September 2004.
17. D.M. Mayerich, “Acquisition and reconstruction of brain tissue using knife-edge scanning microscopy”, M. S. thesis, Texas A&M University, College Station, December 2003.
18. B. H. McCormick and P. Aragonda, “Volume data set polymerization: discriminant functions for edge labeling”, Submitted to VolVis ’04.
19. B.H. McCormick, B. Busse, P. Doddapaneni, Z. Melek and J. Keyser, “Compression, segmentation, and modeling of filamentary volumetric data,” *Proceedings of 9th ACM Symposium on Solid Modeling and Applications*, pp. 333-338, June 2004.
20. B.H. McCormick, B. Busse, Z. Melek and J. Keyser, “Polymerization strategy for the compression, segmentation, and modeling of volumetric data”, Technical Report 2002-12-1, Department of Computer Science, Texas A&M University, College Station, December 2002.

21. B.H. McCormick and D.M. Mayerich, “Three-dimensional imaging using knife-edge scanning microscopy”, *Microsc. Microanal. 10*, Suppl. 2, pp.1466-67, 2004.
22. F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review*, vol. 65, pp. 386-408, 1958.
23. J.F. Thompson, B.K. Soni and N.P. Weatherill, *Handbook of Grid Generation*, Boca Raton: CRC Press, pp. 1-4, 1999.
24. G.H. Wannier, *Elements of Solid State Theory*, Cambridge: Cambridge at the University Press, pp. 1-38, 1959.
25. A. Watt, *3D Computer Graphics*, (3rd ed.), Reading: Addison-Wesley, 2000.
26. W. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide*, Reading: Addison-Wesley, 1999.

VITA

Prathyusha Aragonda received Bachelor of Technology in computer science and engineering from Sri Venkateswara University, Tirupati, India in 2002, where she graduated in first position in her class. She has been working towards her M. S. degree from Fall 2002 at Texas A&M University. She is planning to start her professional career soon after her graduation in December 2004.

Following her recent marriage, she is relocating to Boise, Idaho. She can be reached via email at prathyusha_a@yahoo.com or at this permanent address: D. No. 20-544, A. M. Street, Mittoor, Chittoor – 517001, Andhra Pradesh, India.