SIERRA : an octree-based spatial data system for neural data
by Louis Glassy

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
Neuroinformatics requires the manipulation and querying of data in spatial, visual, and relational forms. SIERRA, the Spatial Interface for Efficient Relational Retrieval and Analysis, is a software system developed for storing and querying spatial representations of neural data within a relational data framework. This system uses a hybrid of proven spatial data structures and Data Base Management System (DBMS) technology, in the form of linear octrees and the Oracle commercial relational DBMS (RDBMS), to meet the data retrieval and visualization requirements of neuroscience research in a simple yet elegant way.

SIERRA: An Octree-based Spatial Data System for Neural Data

by

Louis Glassy

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

Montana State University
Bozeman, Montana

1 September 1998

APPROVAL

of a thesis submitted by

Louis Glassy

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

August 28, 1998
Date

Chairperson, Graduate Committee

Approved for the Major Department

August 28th 1998
Date

Head, Major Department

Approved for the College of Graduate Studies

9/8/98
Date

Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature Louis Glassy

Date 1 September 1998

Contents

List of Figures

Abstract

Neuroinformatics requires the manipulation and querying of data in spatial, visual, and relational forms. SIERRA, the Spatial Interface for Efficient Relational Retrieval and Analysis, is a software system developed for storing and querying spatial representations of neural data within a relational data framework. This system uses a hybrid of proven spatial data structures and Data Base Management System (DBMS) technology, in the form of linear octrees and the Oracle commercial relational DBMS (RDBMS), to meet the data retrieval and visualization requirements of neuroscience research in a simple yet elegant way.

Chapter 1

Introduction

In neuroscience, there is a growing awareness of the importance of spatial relationships of neurons in biological information-processing systems [12, 22].

Nervous systems represent and process internal models of external stimuli and environments with *neural maps*. Neural maps are basic computational units in nearly all nervous systems, and consist of arrays of neurons across which there are systematic variations in the values of perceived or computed parameters. In the neural map of the cercal sensory system in the common house cricket *Acheta domesticus*, the directions and velocities of air current stimuli to the cerci are encoded in the cricket's terminal ganglion, a mass of intersecting neurons located in the rear of the cricket's abdomen (Figure 1.1). Each cercus is covered with approximately 1000 filiform hairs (Figure 1.2), each of which is connected to a sensory neuron, or *afferent*, the axons of which project in an orderly array into the terminal ganglion to form a functional representation of air current velocity and direction.

To understand the underlying algorithms performing neural computation in a neural map, we need to know the details of the anatomy of the neurons from

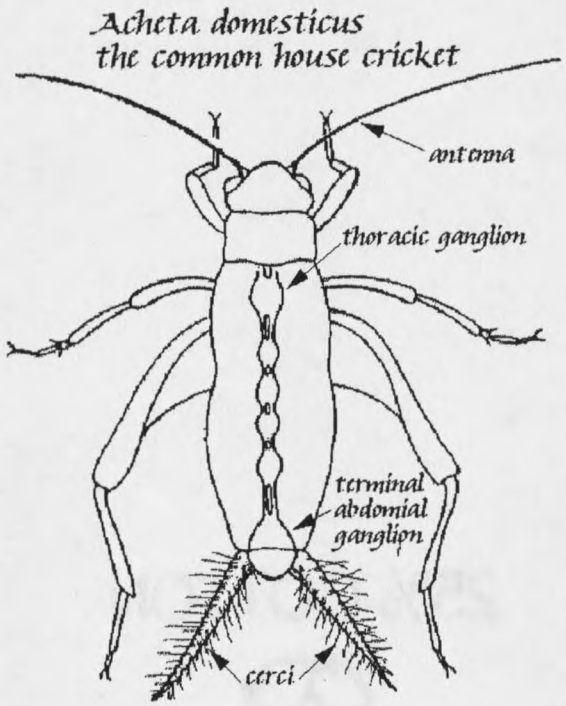


Figure 1.1: Cricket cercal system, from [12]

Figure 1.2: Microphotograph of cricket cercus, modified from [12]

which the map is constructed. Anatomy in this context refers to the topological and spatial relationships of the neurons in the terminal ganglion.

The terminal ganglion acts as a multipurpose neural connection and processing center for approximately two dozen interneurons and two thousand afferents. To understand how sensory information is processed in the terminal ganglion, it is necessary to observe the communication and interaction of entire groups of neurons, or *neural ensembles*, as opposed to the activities of single neurons in isolation. As the anatomical relationships of neural ensembles determine the computational properties of these ensembles, a clear understanding of neural map topologies will give us new insights into the structure, function, and algorithmics of information processing networks.

It is this requirement for ensemble-based analysis that spurred the devel-

opment of SIERRA[1], a component of the NEUROSYS[2] neural visualization system. To find out which sets of neurons intercommunicate, and thus form larger processing units within the terminal ganglion, we must be able to identify and visualize the set of neurons that are physically in contact with or near a given neuron of interest.

To frame this requirement in its simplest form, a spatial data system for neurons must support intersection queries of spatial objects stored in a persistent database. This captures the functional objective of SIERRA — to provide an acceptably efficient and persistent spatial data system which makes spatial intersection queries possible and convenient for nonspecialists to use in the context of biological research. Since the host system NEUROSYS already used an SQL-based relational database system (Oracle) for managing experimental data, it was decided for engineering reasons to develop SIERRA within the context of an RDBMS. This required us to solve the problem of representing spatial-topological data within a relational data framework.

Normally, a relational data model cannot represent spatial data in an efficient and useful form. The reason for this is that a relational system returns only quantitative data, such as lists of tuples, and not qualitative data, such as topological connectedness, proximity, or overlap [6].

Our solution to this apparent mismatch between the topological nature of spatial data and the relational form was to encode the spatial data in a form which permitted both storage in relational tables and manipulation via relational operators. We call this representational sleight-of-hand *linearization*, and

[1] Spatial Interface for Efficient Relational Retrieval and Analysis, see [8].
[2] http://nervana.montana.edu/projects/neurosys.

applied it to the creation of linear octrees which were treated as relational data by the underlying RDBMS. The details and significance of linearization will be discussed in Section 3.2.

Chapter 2

Earlier Work

We examine the earlier work on spatial databases for bioinformatics from two viewpoints: (a) early attempts at spatial datasystems in neurobiology, and (b) nonbiological spatial datasystems which use concepts and data structures similar to those employed by SIERRA.

2.1 Spatial data in bioinformatics

The principal advance of SIERRA is that it provides, to our knowledge, the first example of a true spatial database in the field of neuroscience. The earliest and still most common approach taken in the past in constructing spatial databases for neuroscience was to (a) section the object under study, (b) photograph or digitize the sections, and (c) identify manually features of interest on each section so obtained.

The results, while useful for descriptive and qualitative purposes, lacked a quantitative basis from which deeper patterns of structure and function could be derived. A recent and fine example of this "flip book" type of bioinformatics altas is the *Atlas of the Human Brain*, by Jennes, Traurig, and Conn [13].

A later refinement of the "flip book" type of neuroinformatics database is the attributed image model exemplified by the FLYBRAIN project [24]. In this type of database, an axial series of 2-D images were collected of the 3-D biological specimen[1], and textual descriptions and biological attribute data were associated with each 2-D image collected.

The key to making this type of database queryable by other than purely graphic means is to provide a multitude of textual and numeric references (or fields) in the attribute data connected to or associated with each image. In this way, the user can use any of the attribute fields provided to examine sets of image data. While this avenue of providing aspatial attribute-oriented queries looks workable, in practice it is cumbersome and awkward to use, since the number of attributes necessary to capture a specific pattern or systemic behavior may not be bounded by any usefully small set of constraints.

The true problem with this type of spatial database is that like the "flip book" database, the data in an image-oriented database are not rigorously aligned or registered according to a common 3-D reference frame, and thus cannot usefully be inspected or analyzed by quantitative methods, such as spatial statistics. This problem of unregistered data is perhaps one of the most vexing difficulties facing modern neuroinformatics, since the objects of investigation show gross morphological and functional identity but lack precise spatial referents in the sense that inorganic spatial domains have them, such as a pattern of roads in a geographic information system. In biological systems, we cannot define a canonical spatial reference frame which is reasonable in the face of inter- and intraorganismic variations of structural spatial distribution.

[1] In the case of FLYBRAIN, the test animal is the common fruitfly *Drosophila melanogaster*.

In both the "flip book" and image-based biological databases, the problem of registration is largely ignored, since a database of these types restricts itself to image sections taken from a single biological specimen, and since there is no effort made to perform quantitative analysis on the spatial structures represented in the image data.

SIERRA, in contrast, enables a fully quantifiable approach to be taken in analyzing and exploring biological structures, since we have in fact performed a statistical spatial normalization on the 3-D neural data used by NEUROSYS. In other words, we have positioned the full body of our data within a consistent and uniform spatial reference frame, while accounting for variations between test animals from which the data were collected. Because of this, a user of SIERRA can perform analyses of data based on true anatomical and spatial structures and relationships. These analyses are quantifiable and reproducible to an extent never before achieved in bioinformatics.

2.2 Spatial databases

Hierarchical data structures and relational databases have been used in various configurations before. The hierarchical spatial data structure used in SIERRA is the octree [19], but other choices are conceivable, depending on what specific operations one which wishes to perform on the data.

The SB+ tree [11] uses a generalization of B+ trees with minimum bounding rectangles. This data structure permits efficient region-based queries,[2] but does not permit spatial queries[3] in which the search space is defined by actual objects

[2]For example, "What objects are in a specified region of space?"
[3]For example, "Which neurons are close to neuron X?"

in the database. This latter ability[4] is a requirement of NEUROSYS, and thus is provided by SIERRA.

The quadtree is the 2-dimensional analog of the octree, and has seen use in commercial GIS's. SPANS [27], MapInfo [25], and CARIS [28] are proprietary, commercial GIS's which use quadtrees to manage spatial data [17, 9, 21]. In addition, SPANS and MapInfo claim to have a degree of interoperability with commercial RDBMS's such as Oracle and Informix.

It is not clear from the publically-available information to what extent these GIS's use a relational data substrate to store and manipulate spatial data, although MapInfo claims to support a "spatial SQL" and thus would appear to use relational structures throughout its underlying thematic[5] data layers.

An early example of a quadtree-based spatial data system was QUILT [20], a GIS used primarily for cartographic purposes. A fundamental difference between QUILT and SIERRA is that QUILT was principally used for managing the visual display characteristics of spatial data, while NEUROSYS handles visualization for SIERRA.

An early spatial data system that bears some resemblance to SIERRA, at least conceptually, is the GEOVIEW GIS [14, p. 531] [30], an RDBMS-oriented GIS developed at the University of Edinburgh. GEOVIEW pioneered the use of relational methods for spatial data, and showed that a wide range of spatial data types could be stored and retrieved efficiently in a relational database provided that certain conditions were met, namely: (a) that the spatial data could

[4]Entity-based spatial queries.

[5]In GIS terminology, *thematic data* are data which are related in a functional or "theme-like" sense. For example, the roads in an urban planning GIS are all entities which share a single nominal classification, namely, that of being a road.

be accurately mapped to RDBMS-native datatypes, and (b) that the spatial data could be represented with a unified relational table structure. This latter constraint permits the use of efficient relational operations (UNION, INTERSECTION, JOIN) on the spatial data.

A key distinction between GEOVIEW and SIERRA is that the former used quadtrees primarily as a space-saving mechanism for bulky 2-D grid data; while it was possible to perform intersection or GIS overlay operations on quadtree grids, the means for doing so depended on a complex "enumerate-and-intersect" algorithm, rather than the conceptually simpler "sort-and-count" strategy used by SIERRA.

Chapter 3

Methods

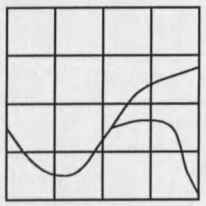
3.1 Spatial data structures

Many spatial data structures exist [4] which could have been selected for use in SIERRA The spatial data structure chosen was the octree [19], a hierarchical spatial data structure suitable for storing 3-D data.

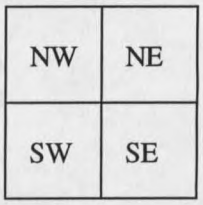
Subsequently, other spatial data structures such as k-d trees, segment trees, and BSP-trees [4] were considered but rejected as unimplementable within the time constraints of the project. These data structures offer different time-space tradeoffs relative to the octree, and may prove useful in later development of SIERRA.

For expository purposes, we will discuss the 2-D analog of the octree, the *quadtree*. A quadtree is a quaternary tree in which the ordering of the leaf nodes represents a recursive square tiling of 2-space. As long as the spatial ordering is used consistently, the precise ordering of the child nodes of each interior node is in practice immaterial, although some spatial orderings have group-theoretic symmetry properties of purely mathematical interest.[1]

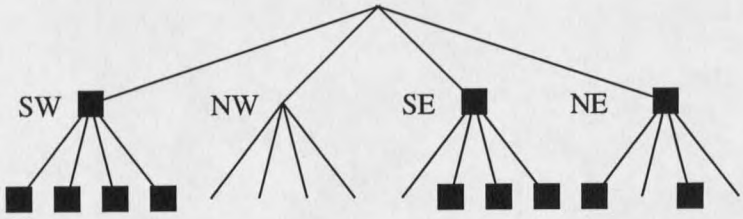
[1] In a separate investigation, we found that the 4-permutations that represent the Peano Z-ordering, the Liu-Schrack U-ordering [16], and the third possible ordering of $[4^4]$ space, the



(a)



(b)



(c)

Figure 3.1: Quadtree representation of an object (a) an object in space (b) assignment of spatial quadrants (c) quadtree representation of object

In SIERRA, we use a variant of the Peano or Z-ordering [14, pp. 162–167]. In the two-dimensional quadtree case, this means the 4 children of any interior node of a quadree are ordered SW, NW, SE, and NE, consecutively, as shown in part (b) of Figure 3.1. In the working implementation of SIERRA, octrees are used; these use octenary (8-way) trees (hence, *octrees*) to model a recursive cubic decomposition of Euclidean 3-space.

Figure 3.1 illustrates a neuron, the SW-NW-SE-NE or Peano ordering of space, and the structure of the corresponding quadtree. The square nodes in

α-ordering, are all cosets of the alternating group A_4.

the tree represent object incidence in the corresponding quadrant or subquadrant of space. Each level in the tree represents a spatial approximation of the represented object, with the leaves denoting the highest-resolution or best spatial approximation of the object.

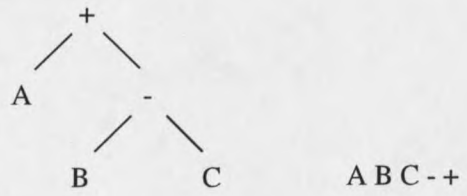
3.2 Linearization

Quadtrees or octrees represented in the "natural" or dendritic form are simple to visualize, but cannot be stored as such in a conventional database system with persistent backing store. Thus, we need to convert the tree-structure of an octree into a persistent but semantically equivalent form for disk storage. We do this by *linearizing* the tree, i.e., by mapping the recursively-structured N-dimensional data in the tree to linear sequences which can be stored and manipulated in a relational database. To be useful in a spatial database, this mapping must be invertible.[2]

Linearization, while little discussed as a technique in its own right, emerges from a common theme in computer science — data in an inconvenient form can be translated into a form more suitable for achieving particular ends. For example, in compiler theory, it is common to discuss *expression trees* as a way of representing and thinking about arithmetic expressions. A postorder traversal of such a tree yields a simple 1-dimensional linearization, namely, a *postfix expression* semantically equivalent to the original expression tree. Figure 3.2 illustrates this.

Other tree linearizations exist which do not correspond to simple traversals.

[2] If the mapping were non-invertible, multiple spatial features in the original object would be mashed into the same linear code. The linear code would not be able to distinguish between its progenitors, which implies that the spatial-to-linear transformation would be lossy.



(a) (b)

Figure 3.2: Tree Linearization (a) expression tree (b) linearized postfix form

If we take the *paths* from the root of the spatial data tree down to the leaves, we have yet another linearization of the tree. In quadtrees, the branching factor at each level is 4; in octrees, it is 8. Thus, there are a fixed number of distinct values required to encode the "directions" or turns one would take in going from the root of the tree down to each leaf node. For a leaf node at depth n, n symbols are required to fully specify the root-to-leaf path, where each symbol can assume one of b distinct values, b being the fixed branching factor of the tree. If the symbol sequence $c_1 c_2 ... c_n$ represents the path from the root to a given leaf node c, then the left prefix of this string $c_1 c_2 ... c_k$ is a resolution-k approximation of node c. In spatial terms, this denotes a region of space which approximates the entity c; this entity may be interpreted as a point, a line segment, an area, volume or hypervolume, depending on the physical significance or meaning of the data.

Linear octrees [19] were selected as a suitable data structure offering efficient query and storage characteristics for SIERRA. A linear octree stores spatial data as paths from the root to the leaves of the octree, as described above. These paths can be stored in a conventional relational database table as strings

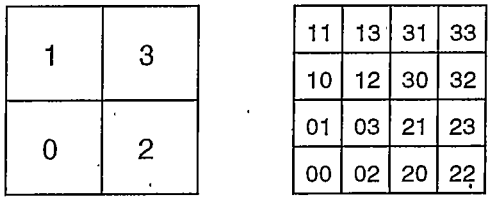


Figure 3.3: Quadtree spatial ordering at two levels of resolution

of characters.[3] These tables consist of columns of octree location codes, with each column representing a different level of spatial resolution for the neuronal data points. Each row in a neuron's table represents the spatial and attribute keys for a single defining point.

Figure 3.3 illustrates the 2-D projection of the ordering of space used in SIERRA, at the two lowest levels of spatial resolution.[4]

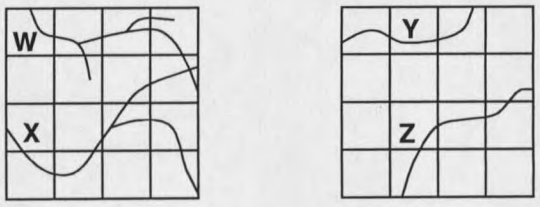
Figure 3.4 depicts 4 sample neurons and the corresponding location codes (LCs) that would be stored in relational database tables for these neurons. If a neuron is present in a specified cubic voxel of space, we store the LC for that voxel in a row of the neuron's spatial data table, along with an identifier for the neuron.

3.3 How spatial queries work

Spatial queries in SIERRA enable the user to query a neural database for spatial overlaps between a reference neuron and a set of other neurons in the database. These queries take four input parameters:

[3] If space consumption were critical, we could encode each octree path symbol as an octal digit of 3 bits, since $2^3 = 8$.

[4] While SIERRA uses octrees with a 3-D ordering of space, we use 2-D quadtrees for explanatory purposes in this thesis. The 2-D variants of these data structures are simpler to illustrate than the 3-D versions, and are effective for illustrating the basic ideas involved.



(a)

ID	*LC*	*ID*	*LC*	*ID*	*LC*	*ID*	*LC*
W	11	X	01	Y	11	Z	02
W	13	X	00	Y	13	Z	03
W	12	X	02	Y	31	Z	21
W	31	X	03			Z	23
W	33	X	21			Z	32
W	32	X	23				
		X	22				
		X	30				
		X	32				

(b)

Figure 3.4: Sample neurons (a) in space (b) as stored in linear octree form

1. the name of reference (or *base*) neuron

2. the set of names of query neurons

3. the spatial resolution of the query

4. the overlap threshhold percentage

The last two of the four input parameters, spatial resolution and overlap percentage, have default values which are pre-wired to 30μm and 60%, respectively, to enable a simpler form of query input for the novice user. These defaults were established by examination of the statistical properties of a nominal sample of interneurons and afferents from the neural database.

The ability to vary the spatial resolution of a query enables the user to change the degree of spatial proximity or "closeness" between neurons, based on the current task. In practice, the use of the highest spatial resolution is not always optimal, since queries with an artifactually high resolution can result in a nonadjacency result for neurons with known synaptic connectivity.

Provision of the variable overlap control will enable us to validate SIERRA's answers against other methods of overlap calculation, such as overlap by edge-list intersection [4, pp. 33–42] and the OVERLAP neural modelling program developed by Jacobs and Theunissen [12].

Taken together, these four query inputs confer considerable flexibility in selecting a precise set of spatial relationships for the input set of neurons.

To perform a spatial query in SIERRA, the user selects an initial or *base* neuron, and a set of one or more *query* neurons to intersect with the base

neuron.[5] An intersection at resolution level r exists between the base neuron b and a given query neuron q when these neurons overlap in space by at least a minimum threshhold percentage t.

Let M_r be the set of distinct LCs for neuron M at spatial resolution r. Let $|M_r|$ be the size of this set. When the selected resolution r is less than the maximum resolution of the LCs stored for M, the LCs of M_r are formed by taking left-anchored LC substrings of length r. For example, $Z_2 = \{02, 03, 21, 23, 32\}$, while $Z_1 = \{0, 2, 3\}$. Given a base neuron b, a set of query neurons Q, a threshhold percentage t, and a level of spatial resolution r, the output of a spatial query in SIERRA is the set of neurons described by

$$\{q \in Q : |b_r \cap q_r| \geq |q_r| \cdot t\}.$$

Figure 3.5 contains example queries based on the neurons presented in Figure 3.4, with each step of the evaluation process and the final query outputs. The mathematical underpinnings of these queries are based on the *Peano tuple algebra*, a succinct treatment of which may be found in [14, pp. 520–533]. The relational join operation from the tuple algebra is described in [14, p. 531], and [3, pp. 62–65].

A spatial query may be spatially-oriented [6] or entity-oriented, in which the space of interest is defined by an entity in that space. SIERRA's queries are entity-oriented, with a default resolution of approximately 30 microns, and a default minimal overlap threshhold of 60%.

[5] In practice, the base neuron is most frequently an interneuron, and the query neurons are generally *afferents*, or sensory neurons.

[6] In GIS terminology, these would be called *region queries*.

B	Q	r	t	*evaluation*
W	$\{X\}$	2	0.1	$W_2 = \{11, 13, 12, 31, 33, 32\}$ $X_2 = \{01, 00, 02, 03, 21, 23,$ $22, 30, 32\}$ $W_2 \cap X_2 = \{32\}$ $\lvert W_2 \cap X_2 \rvert = 1$ $\lvert X_2 \rvert \cdot t = 9 \cdot 0.1 = 0.9$ output: $\{X\}$, as $1 \geq 0.9$.

(a) $W \cap \{X\}, r = 2, t = 0.1$

B	Q	r	t	*evaluation*
Y	$\{Z\}$	1	0.5	$Y_1 = \{1; 3\}$ $Z_1 = \{0, 2, 3\}$ $Y_1 \cap Z_1 = \{3\}$ $\lvert Y_1 \cap Z_1 \rvert = 1$ $\lvert Z_1 \rvert \cdot t = 3 \cdot 0.5 = 1.5$ output: $\emptyset$, as $1 \not\geq 1.5$.

(b) $Y \cap \{Z\}, r = 1, t = 0.5$

B	Q	r	t	*evaluation*
X	$\{W, Z\}$	2	0.8	$X_2 = \{01, 00, 02, 03, 21,$ $23, 22, 30, 32\}$ $W_2 = \{11, 13, 12, 31, 33,$ $32\}$ $X_2 \cap W_2 = \{32\}$ $\lvert X_2 \cap W_2 \rvert = 1$ $\lvert W_2 \rvert \cdot t = 6 \cdot 0.8 = 4.8$ $Z_2 = \{02, 03, 21, 23, 32\}$ $X_2 \cap Z_2 = \{02, 03, 21,$ $23, 32\}$ $\lvert X_2 \cap Z_2 \rvert = 5$ $\lvert Z_2 \rvert \cdot t = 5 \cdot 0.8 = 4.0$ output: $\{Z\}$, as $1 \not\geq 4.8$ and $5 \geq 4.0$

(c) $X \cap \{W, Z\}, r = 2, t = 0.8$

Figure 3.5: Spatial queries: (a) $W \cap \{X\}, r = 2, t = 0.1$ (b) $Y \cap \{Z\}, r = 1, t = 0.5$ (c) $X \cap \{W, Z\}, r = 2, t = 0.8$

3.4 Implementation details

The two versions of SIERRA implemented so far have stored linear octree data in different forms. The initial `perl` implementation of SIERRA stored octree location codes (LCs) as strings of ASCII characters, and used substring matching to create spatially-approximate codes for reduced-resolution queries.

The current embedded-SQL implementation of SIERRA stores LCs as columns of machine integers, since this data type is the most time-efficient of those supported by Oracle, the host RDBMS. Reduced-resolution queries are mechanized by choosing the appropriate k-resolution column in an SQL SELECT statement.

Pseudocode for a spatial intersection query in the current version of SIERRA is provided in Appendix A.

Chapter 4

Conclusions

4.1 The role of SIERRA within NEUROSYS

SIERRA forms the spatial database component of NEUROSYS, a comprehensive neuroinformatics exploration and research tool. In normal operation, a NEUROSYS user does not explicitly invoke SIERRA as a standalone program, but instead specifies via a graphical interface a query which may contain both relational and spatial constraints. Purely relational queries are performed by embedded SQL transactions on tables in the Oracle RDBMS used by NEUROSYS. Queries which contain spatial constraints trigger the execution of SIERRA. Results of spatial queries may be displayed either textually or graphically, using the OpenInventor-based graphic interface of NEUROSYS. Textual results may be presented either as a list of identified neurons matching a given set of input query criteria, or graphically rendered in real-time with different colors assigned to different neurons.

A key element to the success of NEUROSYS is the ability to perform locational queries between sets of spatially-registered neurons. For these queries to be visualized with acceptable real-time response, algorithms and data struc-

tures supporting fast spatial computation were essential. In NEUROSYS, spatial queries may serve as "preprocessors" for other programs, such as the neural activity pattern animator, NAPA [10].

Neurons as naturally-occurring objects have an inherently dendritic (tree-like) structure. A common way to store lines in spatial data systems is as tables of line-segments [14]. Statistical analysis of the raw neuronal spatial coordinate data indicated that a point-oriented approximation of the neurons would provide adequate spatial connectivity for the projected use of the database — spatial intersection queries — since the mean length of the segments which define the neurons' branching structure is on the same order of magnitude as the mean diameters of these branches (4 microns), while the resolution of a typical spatial intersection query is 4 to 7 times this size. The point-oriented approximation permitted a simple and efficient encoding of neurons as linear octrees.

Neurons in the database are spatially represented as sets of points located in a uniformly-registered 3-space. Thus, the tree-like spatial structure of a neuron is actually stored as a cloud of discrete points. These point sets are encoded as linear point-volume octrees, a 3-D generalization of PR-quadtrees [19]. Since the Oracle RDBMS forms the relational data substrate of NEUROSYS, the ability of SIERRA to use RDBMS technology further simplifies the overall software architecture of NEUROSYS.

The relational octree scheme used in SIERRA can handle additional data types in the form of areal (2-D) or volumetric (3-D) objects without changing the internal representation of the octrees. By assigning a different interpretation to data-incidence within a given 2-D or 3-D coordinate region, we can re-use

existing code to store and access new types of biological data, e.g., 3-D reconstructions of neurons and ensembles from confocal imaging, and microtomized 2-D images of cellular structures. The ability of octrees to spatially approximate data is insensitive to the true thematic nature of the data; a voxel can as easily represent a 3-D coordinate as a linear or areal spatial entity.

By using SQL and Oracle RDBMS to store and access the neuronal octree data, SIERRA leverages existing RDBMS functionality and capability within a distributed client-server environment.

Information about SIERRA's implementations and `perl` code for the original SIERRA prototype may be found at `http://nervana.montana.edu/sierra`.

4.2 Performance considerations

The octree representation of neurons in SIERRA is typically precomputed and stored as relational tables. The algorithm used to encode $\langle x, y, z \rangle$ coordinate triples of neuronal data as location codes (LCs) runs in $O(m \cdot n)$ time, where m is the maximum depth of the octree in resolution levels, and n is the number of $\langle x, y, z \rangle$ points used to define a neuron. The number of defining points ranges from a few hundred, for sensory neurons, to several thousand for interneurons. Since in practice m is fixed to a small integer (8, in the current database), the process of generating octree tables is effectively linear in time complexity.

An intersection query in SIERRA runs in $O(n \log n)$ time, based on a simple catenating merge of neural data ($O(n)$ with respect to the total number of defining points in all neurons participating in the query), followed by an internal sort by the database ($O(n \log n)$), followed by a counting pass over the sorted data ($O(n)$ with respect to the total number of data points). In practice, the

performance bottleneck in SIERRA's intersection queries stems from the speed of I/O in the underlying RDBMS.

At a practical level, the original `perl` prototype performed worst-case spatial queries (one base neuron, 200 query neurons) in 10 to 12 seconds on a 200 MHz Pentium Pro system running RedHat Linux 4.2. On a 180 MHz R4000-based SGI O2 workstation running Irix 6.3, worst-case queries with the `perl` implementation took 45 to 50 seconds. The large discrepancy in performance between the Linux runs and the Irix runs can be attributed to the Irix I/O kernel subsystem, which appeared to be less tuned for speed than the Linux system. Both systems on which the `perl` version of SIERRA was run (Linux and SGI Irix) had 64 MB of RAM, and the performance tests were run on effectively idle systems, so the culprit cannot be excessive interaction with the virtual memory system.

The first SQL reimplementation of SIERRA used embedded SQL in connection with the commercial Oracle DBMS. On an SGI O2, worst-case query response time increased to 6 minutes. This was too slow for interactive use. Gross inefficiency was pinpointed in the structure of the embedded SQL query, which used nested SQL SELECTs, each with UNIQUE command. While the answers returned were correct[1], the UNIQUEs in Oracle SQL proved to be very slow.

In the second SQL reimplementation of SIERRA, we traded time for space, and restructured the spatial data from one table of nine columns (1 column for neuron ID, 8 columns for LCs at the 8 supported spatial resolutions) to 8 tables

[1]The original embedded SQL query code was based on the pseudocode in Appendix A, and gave answers that agreed to within machine precision of the `perl` implementation.

of 2 columns each (1 column for neuron ID, 1 column for LCs). Each of the new tables contained the spatial data for a single spatial resolution, and had all redundant LCs removed at table load time.

This implementation of SIERRA is used with the current (1998) version of NEUROSYS, and performs a worst-case spatial query at approximately the same speed as the original `perl` prototype. The advantage of SQL-based query support is that all the spatial data now resides in DBMS tables, while the `perl` version had to duplicate, store, and query all the data in a separate form based on flat files. The flat-file spatial database proved to be a data management nightmare, since the precise structure of the flat files and neuron naming conventions changed periodically to accommodate the evolving requirements of NEUROSYS; the changes in the source data necessitated ongoing modifications to the `perl` prototype.

4.3 Further work

SIERRA was developed under stringent time constraints to solve a real problem in bioinformatics, namely, that of identifying spatially proximate neurons using a preexisting relational database. As a proof-of-concept system, SIERRA works, and may be considered a success. The current system meets minimal performance and useability requirements, and will be used in inquiry-based lab activities which will give neurobiology undergraduates their first real experience with the intricacies and function of neural ensembles [26].

Creators of practical and robust spatial data systems could use the concepts pioneered in SIERRA in a more evolved form to build general-purpose spatial databases for bioinformatics. Future developers will want to carefully review and

analyze some of the short-term design tradeoffs we resolved in the development of SIERRA. These include:

- *Choice of spatial data structure.* The octree spatial data structure was chosen because it was well-understood by the developers; given the time constraints of the project, minimizing development time took precedence to maximizing system performance.

  The GIS and computational geometry communities have identified a number of other spatial data structures, one or more of which may prove a better fit for the types of questions bioinformaticists would like to ask. These data structures include bintrees, strip trees [19]; k-d trees, segment trees, and BSP-trees [4]; an extension of the SB+ tree [11]; adaptive recursive tesselations, a generalization of the quadtree [23]; and natural trees [29]. Each of these data structures have particular strengths and weaknesses, as does the octree.[2] SIERRA's simple octree-based design opens the doors to exploration of more sophisticated data structures which may provide a better semantic fit to spatial operators not employed in the current version of NEUROSYS.

- *Prototype vs. production code.* SIERRA was initially implemented in `perl` over a short period. The `perl` implementation was originally intended to be only a proof-of-concept model of SIERRA, but nevertheless was extended and pressed into student lab use as a web-based CGI script even after the more-refined SQL version was deployed for workstation users,

[2] A weakness of the octree is that it consumes considerable space if used on data that is not highly "clumpy." Geostatistics provides us with a number of measures of spatial variability [5, pp. 129–163], or "clumpiness," which can give us a good idea whether or not an octree is a good choice of data structure for a given set or type of biological data.

due to the higher memory requirements of the SQL version when run with many (15) simultaneous users.

The development of a production-quality bioinformatics spatial database would greatly benefit from a clear and static specification for data format, performance requirements, and user interfaces.

- *Interface issues.* A key growth-area in bioinformatics is the task of tranferring information from the computer to the user's mind. Clearly, the user interface presented by the computer and the software system plays a role in this information transfer. Lee and Chin [15] developed an iconic interface for accessing and querying spatial databases. This interface lets the user graphically specify spatial relationships and query constraints with icons selected from a toolbar. In conjunction with the topological data model defined by Egenhofer [7], a development of the Lee-Chin approach could give us a convenient, general-purpose and simple-to-use interface for bioinformatics research.

- *Storage efficiency.* SIERRA in its current form pays little attention to issues of space consumption. To handle gigabyte-sized data sets, better on-disk and in-core strategies for managing memory will be essential. Chang and Liou [2] present a method for creating and managing compact linear quadtrees (as small as 1/3 the number of bits per datum of conventional linearization methods). For voluminous 3-D atlases of complex biological structures, such as the brain, Chang's data model may be worth investigating as a refinement to SIERRA.

- *Registration.* In a spatial database, the utility and value of the data is only as good as the quality of spatial registration used. As discussed in section 2.1 previously, the Achilles heel of earlier spatial databases in bioinformatics was the poor or nonexistent registration used for aligning the data.

  The challenge that registration poses in a biological context cannot be understated. Other (nonbiological) spatial data systems such as GISs model spatial phenomena that are clearly defined in an extensional sense: an intersection between two roads, for example, does not have any "fuzziness" associated with it. In contrast, the spatial entities we wish to model from biological systems have locations that vary from animal to animal; this spatial variability reflects the unique developmental trajectory of the individual organism.

  How then, can we treat a given identified neuron as a single spatial entity, when the data collected for locations of this neuron vary from one animal to another? The answer in our work is to locate the neuron in reference to known landmarks or *fiducials* which show little interspecimen variation. In *Acheta domesticus*, these fiducials are reference points established at junctions of major biological structures in the terminal ganglion. The residual variation in location between specimens not removed by referencing neurons with respect to fiducials is removed by taking the simple arithmetic mean of individual neural defining points across animals. Early analysis revealed that the level of error remaining in a neuron's location determined by this method is less than 5%.

In the current version of NEUROSYS, spatial data from individual specimens are aligned manually with a data visualization tool. For large volumes of data, an automated and unsupervised alignment process would be highly desireable. Automated registration of biological imagery is very much an open research issue. The placement strategy of spatial reference points from fiducials and of inferring spatial data coordinate locations from statistical models of fiducials must be done with great care, as some common least-squares fit methods (e.g. Gauss-Jordan matrix inversion on regression covariance matrices), can fail spectacularly when the covariance matrices are ill-conditioned [18, pp. 550–551].

- *Mathematical foundations.* Ideally, one could approach the task of building a spatial data system base with a solid grasp of computational geometry and GIS theory. A understanding and application of the mathematics of spatial data will prove useful in extending SIERRA's capabilities. Barr and Barnsley [1] give a graph-theoretic treatment of spatial query operations which is surprisingly simple and elegant. The aim of using advanced mathematical models of data (instead of simpler hierarchical models such as the octree) would be to enable SIERRA to perform more spatial analysis functions than just intersection queries, and over time evolve into a full-fledged 3-D spatial information system for bioinformatics. For example, SIERRA already extends the relational data model to handle one topological operator, that of spatial intersection. A comprehensive and rigorous topological model of spatial data [7] may simplify the addition other spatial operations to SIERRA while retaining the practicality and

relative portability of the relational data model as a low-level implementation substrate.

Appendix A

SQL pseudocode

```
-- LC is column of location codes at resolution N
base_count = SELECT COUNT(UNIQUE(LC))
             FROM table1
             WHERE id = base_object

FOR other_object IN Q LOOP     -- Q is the set of other objects.

    other_count = SELECT COUNT(UNIQUE(LC))
                  FROM table1
                  WHERE id = other_object

    intersection_count = SELECT COUNT(UNIQUE(LC))
                         FROM table1 t1, table1 t2
                         WHERE t1.id = base_object
                         AND   t2.id = other_object
                         AND   t1.LC = t2.LC

    IF intersection_count >= (threshhold * other_count) THEN
        RETURN ("base object and ", other object, "intersect")
    ELSE
        RETURN ("base object and ", other object, "do not intersect")
    END IF

END LOOP
```

Bibliography

[1] S. Barr and M. Barnsley. A region-based, graph-theoretic data model for the inference of second-order thematic information from remotely-sensed images. *International Journal of Geographic Information Science (IJGIS)*, 11(6):555–576, 1997.

[2] H. Chang and C. Liou. A constant key length coding scheme for linear quadtrees. *International Journal of Geographic Information Science (IJGIS)*, 10(2):205–218, 1996.

[3] C. J. Date. *An Introduction to Database Systems.* Addison-Wesley, 2nd edition, 1977.

[4] M. De Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 1997.

[5] D. Ebdon. *Statistics in Geography.* Blackwell Publishers, 1985.

[6] M. J. Egenhofer. Why not SQL! *International Journal of Geographic Information Science (IJGIS)*, 6(2):71–85, 1992.

[7] M. J. Egenhofer and R. D. Franzosa. On the equivalence of topological relations. *International Journal of Geographic Information Science (IJGIS)*, 9(2):133–152, 1995.

[8] L. Glassy, G. A. Jacobs, and J. D. Starkey. SIERRA: a spatial data system for neuronal data. In *Proceedings of the IASTED International Conference on Computer Graphics and Imaging (CGIM), Halifax, Nova Scotia, Canada*, pages 41–44, June 1–4 1998.

[9] D. J. Hagberg. Personal email communication on MapInfo, 6 July 1998.

[10] K.M. Hodge, G.A. Jacobs, and J.D. Starkey. NAPA: The neural activity pattern animator. In *Proceedings of the IASTED International Conference on Computer Graphics and Imaging (CGIM), Halifax, Nova Scotia, Canada*, pages 15–18, June 1–4 1998.

[11] A. Ibrahim, F. Fotouhi, and S. Hasan. The SB+ tree: an efficient index structure for joining spatial relations. *International Journal of Geographic Information Science (IJGIS)*, 11(2):163–182, 1997.

[12] G.A. Jacobs and F.E. Theunissen. Functional organization of a neural map in the cricket cercal sensory system. *J. Neuroscience*, 16:769–784, 1996.

[13] L. Jennes, H. H. Traurig, and P. M. Conn. *Atlas of the Human Brain.* J. P. Lippincott Company, 1995.

[14]. R. Laurini and D. Thompson. *Fundamentals of spatial information systems.* Academic Press, 1992.

[15] Y. C. Lee and F. L. Chin. An iconic query language for topological relationships in GIS. *International Journal of Geographic Information Science (IJGIS)*, 9(1):25–46, 1995.

[16] X. Liu and G. F. Schrack. A new ordering strategy applied to spatial data processing. *International Journal of Geographic Information Science (IJGIS)*, 12(1):3–22, 1998.

[17] D. Mark. Personal email communication about CARIS, 6 July 1998.

[18] P. M. Mather. Map-image registration accuracy using least-square polynomials. *International Journal of Geographic Information Science (IJGIS)*, 9(5):543–554, 1995.

[19] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260, 1984.

[20] C. A. Shaffer, H. Samet, and R. C. Nelson. QUILT: a geographic information system based on quadtrees. *International Journal of Geographic Information Science (IJGIS)*, 4(2):103–131, 1990.

[21] A. Sorokine. Personal email communication on SPANS, 6 July 1998.

[22] T.W. Troyer, J.E. Levin, and G.A. Jacobs. Construction and analysis of a data base representing a neural map. *Microscopy Research and Techniques*, 29:329–343, 1994.

[23] P. H. Y. Tsui and A. J. Brimicombe. Adaptive recursive tesselation (ART) for GIS. *International Journal of Geographic Information Science (IJGIS)*, 11(3):247–263, 1997.

[24] WWW URL. `http://flybrain.neurobio.arizona.edu/`. FLYBRAIN web site at University of Arizona.

[25] WWW URL. `http://www.mapinfo.com/`. MapInfo GIS home page.

[26] WWW URL. `http://www.nervana.montana.edu/neuroscape/`. NEUROSCAPE web site at the Center for Computational Biology, MSU—Bozeman, Montana.

[27] WWW URL. `http://www.tydac.com/`. SPANS GIS web site at TYDAC Research.

[28] WWW URL. `http://www.universal.ca/`. CARIS GIS web site at Universal Systems Ltd.

[29] D. F. Watson and A. I. Mees. Natural trees — neighbor location in a nutshell. *International Journal of Geographic Information Science (IJGIS)*, 10(5):563–572, 1996.

[30] T. C. Waugh and R. G. Healey. The GEOVIEW design: a relational database approach to geographic data handling. *International Journal of Geographic Information Science (IJGIS)*, 1(1):101–112, 1987.