Neuron PRM: A Framework for Constructing Cortical Networks

Jyh-Ming Lien neilien@cs.tamu.edu
Marco Morales marco@cs.tamu.edu
Nancy M. Amato amato@cs.tamu.edu

Technical Report TR01-002
PARASOL LAB
Department of Computer Science
Texas A&M University
October 3, 2001

Abstract

The brain has extraordinary computational power to represent and interpret complex natural environments. These natural computations are essentially determined by the topology and geometry of the brain's architectures. We present a framework to construct a 3D model of a cortical network using probabilistic roadmap methods. Although not the usual motion planning problem, our objective of building a network that encodes the pathways of the cortical network is analogous to the PRM objective of constructing roadmaps that contain a representative sample of feasible paths. We represent the network as a large-scale directed graph, and use L-systems and statistics data to 'grow' neurons that are morphologically indistinguishable from real neurons. We detect connections (synapses) between neurons using geometric proximity tests.

*This research supported in part by NSF CAREER Award CCR-9624315, NSF Grants IIS-9619850, ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACI-0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by the Texas Higher Education Coordinating Board grant ARP-036327-017.

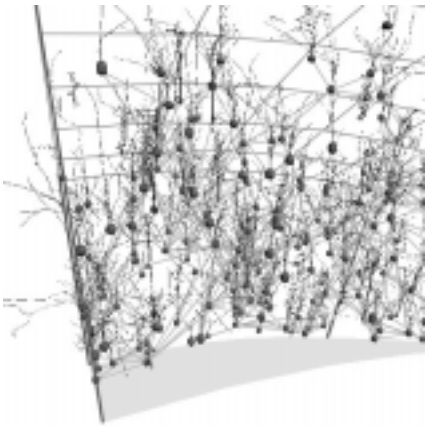


Figure 1: Synthetic neurons generated and connected in the cortex.

1 Introduction

The brain has extraordinary computational power which is determined in large part by the topology and geometry of its structures. A unique instrument developed at Texas A&M University, the Brain Tissue Scanner (BTS), will enable an entire mouse brain to be imaged and reconstructed at the neuronal level of detail. In terms of 3D visualization, this project is the microscopic counterpart of the Visible Human Project at a vastly expanded scale. Moreover, by enabling studies of the topology of cortical networks, it could provide insight into one of the least understood biological processes — neural computation.

Using destructive sectioning and cross-sectional imaging, the BTS can scan an entire transgenic GFP/XFP-stained mouse brain in approximately one month. The data produced by the BTS is used to reconstruct the three-dimensional structure–the neural forests and fibers–of the scanned tissue. However, only a small percentage (usually 2-3%) of neurons will be stained, so the neurons reconstructed from BTS data will be augmented with synthetic neurons that are grown based on the measured biological neurons. Next, their interconnections (synapses) will be generated. The geometry and connectivity of the resulting cortical network will be studied using theoretical techniques and simulation on massively parallel machines. Our central goal is to map and understand cortical network connectivity and geometry.

The work presented here is part of a larger project. Here, we start with geometrical models of reconstructed neurons, and concentrate on the generation and connection of anatomically realistic synthetic neurons needed to complete the cortical network. A simple prototype system has been implemented which borrows from probabilistic roadmap methods (PRM) [10]. Neurons are generated from L-system-based neuron models, whose rules are computed from a statistical analysis of reconstructed actual neurons. We propose techniques for identifying geometrically and anatomically realistic synapses (the connection phase).

2 Preliminaries

Organization of the cerebral cortex. The cortex is a planar and convoluted sheet of tissue with a homogeneous appearance located in the anterior part of the brain. It is rather difficult to set it apart from other parts of the brain, though this can be done with an analysis of impractical connections [6]. In the human, the cortex's surface area is about 2,600 cm^2 and it is 3-4 mm thick. and contains about 28×10^9 neurons. The number of connections or synapses is uncertain but it has been estimated to be about 10^{12}-10^{13}. These are the targets of the millions of nerve fibers that are connected to sensors arranged throughout the body and the origin of stimuli to move muscles [13].

The neurons in the cortex are arranged in several horizontal layers forming local circuits arranged in vertical groups called columns with heavy interconnections in the vertical direction that pass through one or more layers. Some neurons in these columns have output targets in the horizontal direction to areas far apart from the local area.

We are not aware of any work trying to map the entire cortical network. Scientists, however, have made mappings of small areas of cortex.

Neuron models. The neurons are the cells that perform the basic computations in the brain. They are composed of a soma, a set of dendritic arbors that receive inputs from other neurons and axons that serve as the transmitters that give inputs to other neurons. Several classes of neurons have been identified, they are classified by the form of the cell body, the length of the axon, and the distribution of the dendritic trees.

Many different models of neurons have been proposed. Here we are interested in the morphological features of neurons and in their connectivity which will be needed to simulate their function.

The GENESIS (GEneral NEural SImulation System) system [5] is based on the Hodgkin-Huxley model that describes quantitatively the dynamic behavior of the neuron membrane [8]. It can be used to model neurons at many levels of detail, from single compartments of a neuron to large networks of cells. Its main applications so far have been the modeling of single cells, although it has also been used for large scale studies such as constructing models of the hippocampal formation with about 1000 cells or studying the

steady-state activity of a circuit of the granular layer of the rat cerebellum.

In 1994 McCormick and Mulchandani proposed the use of L-systems to represent neuron morphology [12]. These systems, which had been in use to model plants, consist of a set of rules that describe the set of movements that a turtle-like device would have to do to draw the plant or neuron by displacements and changes in orientation. In 2000, Ascoli and Krichmar released L-Neuron which uses an L-system to generate and study anatomically correct neurons [2]. L-Neuron stochastically samples parameters from experimental distributions stored in a neuroanatomica to be used in the generation of a virtual neuron.

Probabilistic Roadmaps. The Probabilistic roadmap method (PRM) is a framework originally developed to compute collision-free paths for robots [10], but it has been shown to be useful in applications ranging from robotics to computational Biology (such as protein folding [14]). This framework has evolved and many methods have been proposed for particularly difficult situations [1, 4, 9]. PRMs involve the construction of a roadmap in which a set of configurations in C-space [11] are generated and connected. This roadmap can be queried for paths between given configurations.

3 Overview of our approach

Our ultimate goal is to map and understand the connectivity and geometry of the cortical network. We begin our modeling by partitioning the cerebral cortex into a set of finite elements (FEs). Each FE has a set of geometrical models of reconstructed neurons which are used to generate additional anatomically realistic synthetic neurons. The cortical network is completed by making connections or synapses between the neurons. The construction process borrows from PRM (Probabilistic Roadmap) techniques by using simple, local techniques to understand a much larger unknown space viewed as a robotics problem. Here, the workspace is enclosed inside the cortical model, and, within this space, points (neurons) are generated and connected. Each point is a different configuration of a tree-like robot. Unlike a traditional PRM, the tree-like robots have the ability to change shape (i.e., connectivity and number of links), so each roadmap node, like a reconfigurable robot, will represent not only a transformation of the robot's joints but also its topology (joint connectivity). In short, we are trying to solve a robotics problem with an infinite numbers of degrees of freedom.

A prototype system has been implemented. The general system structure is described in Figure 2. During *neuron generation*, neurons are "grown" in the *cortex* using information describing the spatial distribution of neurons and their geometry contained in the cortex FE model. The neurons generated are stored in a neuron data base, where a neuron is associated with the FE containing its soma. During *synapse generation*, neurons are connected using transformations, the geometric models constructed during the *neuron generation*, and their known affinities for synapse creation. Finally, a roadmap representing cortical connectivity is ready for analysis and simulation. In summary, the process of modeling the cortical neuron consists of two stages:

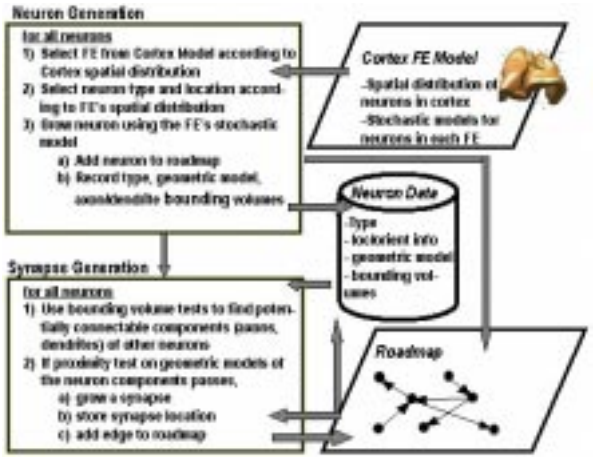


Figure 2: Framework of neuron PRM: Finite elements, Neuron generation/connection, roadmap graph, and database.

1. Generation of Neuron Nodes, and

2. Connection of Neuron Nodes.

In the following sections we show how we use the basic building blocks described so far to construct a virtual cortical network. In Figure 2, the interactions and data streams between the elements described above are shown. We now describe in more detail some of the important data structures and models of the method.

A Neuron is represented as a tree of neural components. The first components are the soma and a set of arbors. Each arbor has as children a set of segments who in turn have children that are either junctions (when they branch) or termination marks (when they are terminal segments). Each segment has a list of micro-segments which may or may not have a spine where synapses take place. Figure 3(a) shows a simplification of a neuron with a soma and two arbors. One of the arbors does not have branches while the other has a junction with two branches, the micro-segments

Figure 3: (a) A simple neuron. (b) Spines on a neuron.

of this neuron have no spines. Figure 3(b) shows a neuron with spines.

Synapses usually occur between an axon and a dendrite, between arbors, or even between arbor and soma. Given that a neuron may be composed of thousands of segments, and may have thousands of synapses, the information necessary to record the presynaptic endings (sender sites for signals) and postsynaptic endings (receptor sites for signals) of the neurons in the cortical network can be massive.

Finite Element Model of the Cortex. The cerebral cortex is partitioned into Finite Elements (FEs) which contain *local information* about types, amount, and spatial distributions of neurons in that FE (see upper-right corner in Figure 2). Each FE can be viewed as a drawer of the cortical model which is populated with many neurons. Users can open a drawer and browse those neurons inside it. The FEs are organized into an adjacency graph in which two FE's are connected if they share a face. More precisely, each vertex in this adjacency graph is an FE and each arc of the graph represents an incident face between two FEs. This graph helps users view neurons in each FE and its neighbors because they may share some local properties, and also helps to speed up the process of finding connections between neurons. More details regarding connection can be found in Section 5.

Databases contain information for growing and distributing neurons (see center part of Figure 2). There are three types of databases.

1. Types of neurons and probability for each type.
2. Spatial distribution of each type of neuron.
3. Statistics data for neuron model.

The information in the first two databases is different from one FE to another, so we put these databases inside each FE. The third database is associated with each neuron type, and its data is obtained by a statistics calculator that is described in section 4.2.2.

A Roadmap Graph (lower-right part of Figure 2) is a directed graph representing an abstract cortical network. Following a common schema used in the PRM method, Neuron-PRM (N-PRM) samples graph nodes and connects nodes using local information. Each vertex in the graph defines a configuration of a neuron and each *directed edge* represents an abstract connection or synapse between neurons. Here, we define an abstract synapse as a set of real synapses from segments of one neuron to another. Since actual synapses can be quite complex, these abstract synapses help us not only maintain data more easily but also provide a hierarchical representation for searching the cortical network.

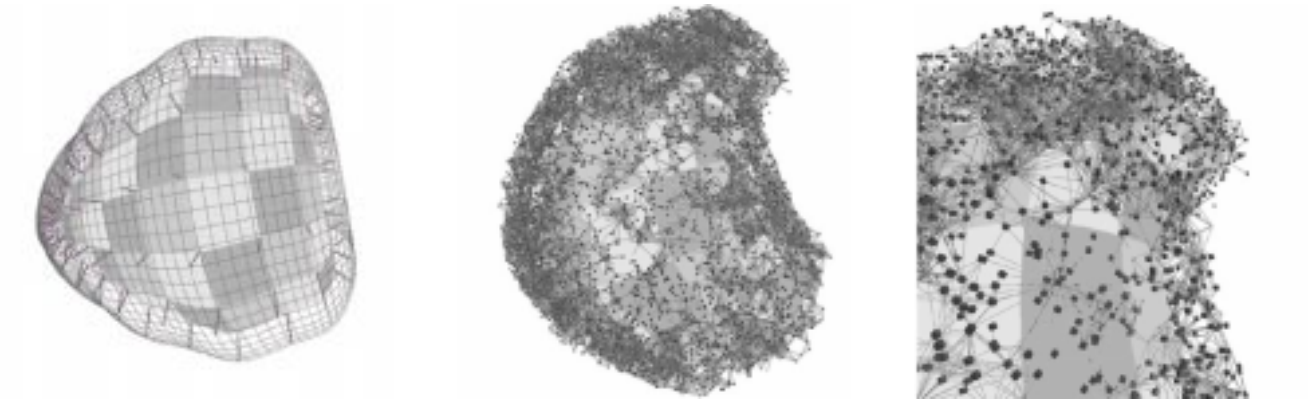
4 Generation of Neuron Nodes

Neuron PRM samples points and grows synthetic neurons during the neuron generation phase. Information gathered in the sampling stage is sent to the neuron generator to create morphologically correct neurons. Therefore a vertex in the N-PRM roadmap or cortical network contains a sampled point that represents the geometry and additional information used to determine the morphology of the neuron. We first describe how points are sampled and then how they are used to generate synthetic neurons.

4.1 Abstract Neuron Nodes

Unlike traditional PRM's where points are generated based on some functional distribution, such as a uniform or a Gaussian distribution, we generate points based on some given statistical distribution for neurons and neuron types. For example, statistically, very few neurons are found close to the upper layers of the cortex, and neuron types and numbers vary from layer to layer. Without statistical data, basic functional sampling strategies are not likely to give us reasonable results. We can use data from the literature or eventually from a system, such as the BTS described in Section 1, which can give information regarding the distributions and types of neurons from a particular brain. Each sampled point represents a neuron configuration in the cortical cortex space.

The *abstract neuron generation* starts by selecting a FE randomly. The probability of selecting a FE is based on the average of the area of its upper and lower bounding patches. Thus, larger FEs have a higher probability of being selected. Given a FE, a neuron type is chosen at random from this FE's neuron type database which defines both the types and distributions of neurons in that FE. Next, the position of the neuron is determined based on the distribution and statistics for its type in the FE's database. This position is normalized in the FE's local coordinates (a unit box), so that we don't have to worry about the 'real' geometry, location, and orientation of the FE in this step. However, in order to generate synapses



(a) (b) (c)

Figure 4: (a) Empty cortical model of mouse. Each finite element colored with different grays. (b) Cortical model filled with abstract neuron nodes and connections. (c) A closer view of cortical network.

(connections) between neurons, we still must convert to global coordinates. Finally, having all this abstract information, we are able to create synthetic neurons as described in Section 4.2. An outline of the algorithm for generating neurons is in the upper-left corner in Figure 2.

To illustrate how neuron nodes are generated in the cortex, we constructed one half of the cortex of a mouse. It resembles an egg-like shape with a curl in the hippocampus and has one of the smoothest cortical surfaces among the mammals. In Figure 4(a) the cortex of the mouse (still empty) is partitioned into FE's, each one defined by a set of upper and lower patches colored with different gray scales. In Figure 4(b) and Figure 4(c), abstract neurons are shown populating this cortex. The process of growing these neurons is described in the following section.

4.2 Growing Synthetic Neurons

To generate neurons in the cortex we made a *neuron server* which is called by N-PRM each time it has to grow a synthetic neuron. The server is provided with the position of the neuron to be created and the finite element it will be grown in. The neuron server has a set of *neuron generators* to grow neurons.

The *neuron server* should determine the type of neuron to be generated and create the proper *neuron generators* to grow one (if one doesn't already exist.) The *neuron generators* create the neuron and store it so it can be used in later steps such as the connection of the neurons and in the future the simulation of the dynamic properties of the cortical network.

(a) (b)

Figure 5: Synthetic neurons: (a) Pyramidal. (b) Granule.

4.2.1 Neuron Generators

The *neuron generator* can generate multiple types of neurons. It has a set of databases with statistical information about each type of neuron. It picks the proper database and uses it to create each component of the neuron with a L-system grammar.

An L-grammar is a context-free grammar in which all non-terminal symbols matching non-terminal symbols of the left-hand-side of the production are replaced in parallel by the right-hand-side of the corresponding productions until the string contains only terminal symbols. L-grammars were used first to model plants in order to draw them. We use them as a tool to generate synthetic neurons that can be drawn, connected with other neurons, and also used to simulate their dynamic properties.

In terms of graphics the L-grammar terminal symbols are used as commands for turtle movements that can be displacements or changes of orientation allowing the turtle to trace the model in 3D space allowing the generation of neurons.

An L-system is an L-grammar that uses random variables as the value of parameters for segments such as lengths or diameters and for rotation angles at the junctions. The random variables come from distribution functions that depend on the neuron type, area and layer in which the neuron will grow. The computation of the parameters for the distribution functions is discussed below.

The generator maps the string generated by the grammar into the tree that models the neuron. These neurons are used later in the connection stage. They also can be displayed or used by the statistics calculator, and in the future to simulate the dynamic properties of the network.

4.2.2 Statistics Calculator

The statistics calculator is responsible for measuring real neurons and generating the databases used by the *neuron generator*. Data for real neurons can be obtained from several public databases released by researchers in different electronic formats. Currently we use the SWC format also used by the database in [7]. This format describes a neuron in terms of linked nodes in a plain text file in which each line defines a compartment of the neuron. Each line contains the following information:

id type x y z radius parent_id

The id is used as a way to link different compartments (through their parent id), the type differentiates compartments from the soma, and can be defined by the producer of the file.

The statistics calculator loads a sample of neurons in SWC format into our tree-like representation. Then the statistics calculator goes through each neuron to measure diameters, lengths, orientations, amounts and rates for each interesting part of the neuron. Finally the maximum, minimum, range, average, and standard deviations are computed and stored to be used in the generation stage. In Figure 5 we can see two synthetic neurons generated with different databases. The statistics database holds information in XML format:

Neuron record in XML

```
<soma>
   <location/> <orientation/> <diameter/>
</soma>
<arbor> <amount/> <orientation/>
   <segment>
      <diameter/> <length/> <taper/> <orientation/>
      <micro-segment terminal spines>
        <spine> <length/> <orientation/> </spine>
      </micro-segment>
   </segment>
   <junction> <branch_rate/> <orientation/> </junction>
</arbor>
```

5 Connection of Neurons

After the FE model has been populated with neurons, that is, after the spatial distribution and stochastic geometric models of the neurons and fibers are ready, the cortical network is assembled by identifying potential synapses and connecting the neurons.

Statistically each neuron has thousands of synapses. An adult human being has more than one hundred billion neurons and even the cortex of the mouse has more than seventeen billion neurons. Based on current technology, it's not feasible to compute and store such a huge amount of synaptic connections between all possible pairs of neurons. The traditional PRM connection strategy of computing all pairwise distances and attempting to connect the k-closest nodes is not feasible for roadmaps with millions or billions of nodes, each of which consists of thousands of segments. To deal with such large numbers of complex nodes, we define simple distance metrics to reject neuron pairs and find potential synapses very quickly.

To reduce the number of connection tests, we only attempt to connect neurons in the same FE and in neighboring FE's. These neighboring FE's are stored in the FE adjacency graph and can be extracted efficiently as described in Section 3. Moreover, most neuron pairs are quickly rejected by a filtering test which checks for intersection of their bounding volumes. Thus, detailed distance computations between

(a) (b) (c)

Figure 6: (a) FE extracted from mouse cortex. (b) Bounding boxes of neurons in FE. (c) Connection of synthetic neurons.

the neuronal segments will only be performed for those neuron pairs that pass the bounding volume test.

Neuron node connection
1. `for` (each neuron, n_1)
2. `for` (each neuron, n_2, in same and adjacent FEs)
3. `if` (bounding volumes intersect)
4. Connect n_1 to n_2
5. `endfor`
6. `endfor`

Three different bounding volumes were tested in line 3: bounding sphere, bounding box, and convex hull. The intersection tests for bounding spheres and boxes can be computed in $O(1)$ time, and in $O(m \log m)$ for the convex hull, where m is the number of points in the neuron's geometric model. Note that, unlike typical PRM roadmaps, the edges of the roadmap are directed. This is because a synapse is defined as a gap between pre-synaptic and post-synaptic terminations. Although the bounding volumes contain no directional information, directional connections will be made during synapse discovery.

Neuron connections based on intersections of the neurons' bounding boxes (BBXs) are shown in Figure 6. Figure 6(a) shows the positions of the neurons and the connections. The denseness of the network is seen in Figure 6(b), and a close-up view of the symbolic connections between the somas of synthetic neurons is shown in Figure 6(c).

Next, a method using detailed segment information is used to more precisely compute locations. We implemented a simple algorithm based on tendencies known for (real) synapse generation. In particular, we use the fact that synapses are usually found between segments and spines (see Figure 3(b)). The synapse discovery algorithm is outlined below.

Synapse Discovery
1. `for` (each neuron, n_1)
2. `for` (each neighbor neuron, n_2, of n_1)
3. `for` (each microsegment, m_1, on n_1 and, m_2, on n_2)
4. Compute distance from endpoint of m_1 to spine of m_2
5. `if` (Distance $<$ Synapse Gap)
6. Report (m_1,m_2) as a synapse
7. `endfor`
8. `endfor`
9. `endfor`

The synapse gap in line 5 is a user specified value which defines the maximum size of gap between the pre-synaptic site and the post-synaptic site.

6 Experiments

To test our framework, we generated neurons in the mouse cortex. The model of the mouse cortex shown in Figure 4 was built in Maya and has 112 FEs. Each FE is modeled as upper and lower NURBS patches with 16 control vertices on each patch. To fill this model, 40,000 neurons, both abstract and synthetic neurons, were generated and distributed in the mouse cortex. Generation of these nodes took an average of 476.55 seconds over 10 runs. For the connection phase, we first tested bounding sphere, bounding box. and convex hull bounding volumes. These three methods were tested with the 40,000 neuron nodes from the previous phase (neuron generation phase). Data about the running times and the discovered edges (abstract synapses) is collected in Table 1.

Several implementation details should be mentioned here. First, each neuron is deleted after its bounding volume is created, and we store only the bounding volume and the seed used to generate the neuron (so it can be regenerated if needed). This is due to the large amount of memory required for each

Bounding volume strategies (Node Connection)			
Method	**Bounding Sphere**	**Bounding Box**	**Convex Hull**
Time for creating bounding volume	23.5 sec.	22.3 sec.	222.1 sec.
Time for connection	271.4 sec.	207.3 sec.	1,682.5 sec.
Number of edge found	1,353,083	307,494	87,110
Average node degree	33.8	7.7	2.2
Max node degree	374	100	38
Number of CC (Connected Component)	15	2,878	15,562
Number of Small CC (less than 4 nodes)	13	2,777	14,86
Number of isolated neuron	12	2,358	13,055
Max CC Size	39,982	35,941	5,957

Table 1: Bounding volume strategy labels for connecting 40000 neurons.

Cortical Network Refinement (Synapse Discovery)			
Input roadmap from	**Bounding Sphere**	**Bounding Box**	**Convex Hull**
Time for finding synapse	36,896.6 sec.	9,838.6 sec.	3,481.2 sec.
Number of synapse found	417,277	416,895	400,065
Max Number of synapses in single neuron	267	267	267
Avg Number of synapses between two neurons	0.3	5.5	16.0
Max Number of synapses between two neurons	241	241	241
Error Rate	97.4%	88.0%	61.5%

Table 2: Synapse discovery for connecting 40000 neurons.

neuron (each contains thousands of segments) and because only the bounding volume was used in the initial connection phase. Second, the convex hulls were constructed from end points of microsegments and the soma using the *qhull* library [3]. Unfortunately, neurons have such complex geometries that many small and narrow triangles are found on their convex hulls. These tiny triangles not only occupy memory but also slow down collision detection. Therefore, we did not compute an exact convex hull, but instead uniformly sampled points on each segment. From our experimental results, the exact and approximate convex hulls generated a similar number of abstract synapses, but the performance was dramatically improved.

The running time for building these volumes, as expected, was much faster for the bounding sphere and the bounding box than for the convex hull. Since the bounding sphere has larger volume than the bounding box and the convex hull, there were many more potential connections identified using the spheres than were found using the other two methods. Also, due to the overly conservative bounding sphere, almost all the neurons were contained in one connected component while almost a third of the neurons are isolated using the convex hull method.

After constructing the roadmaps with abstract synaptic connections, we applied the simple synapse discovery algorithm to identify real synapses. We used 0.015 as the synaptic gap size (neuron height is between 0.3 and 0.9). As in Table 2, the roadmaps from the bounding sphere, the bounding box, and the convex hull are tested separately. The running time for discovery is linear in the number of edges of the input roadmap, so the convex hull is the fastest. Note that the differences in the number of synapses found is very small between the three roadmaps. This is because many of the abstract synapses found with the bounding sphere and bounding box methods contain very few, or no real synapses. The average number of synapses between two connected neurons, and the error rate illustrate this point. The error rate here is:

$$\frac{N_{\text{empty-abs-syn}}}{N_{\text{total-abs-syn}}} \tag{1}$$

Here, $N_{\text{empty-abs-syn}}$ is the number of abstract synapses which did not contain any real synapse, and $N_{\text{total-abs-syn}}$ is total number of abstract synapses in the given roadmap. This measure gives us an indication of how accurate the bounding methods are. Although all methods have error rates higher than 50%, we believe this will decrease when we are able to more densely pack the cortex with neurons. Here, we generated only 40K and in reality there are some 16M neurons in the mouse cortex.

There are a few implementation details worth mentioning regarding the exact synapse discovery. To test for synapses between neuronal segments and spines, the actual model of the neuron is needed. However,

as previously mentioned, we (currently) cannot save the geometric models of all neurons due to memory constraints, and it would be terribly inefficient to create and delete neurons for every test. To address this problem, we created a cache of 2000 neurons and used it to hold the most recently created neurons. If the cache is full, then older neurons are deleted from it. To store synapses in the roadmap, we actually store the IDs of the two microsegments creating the synapse (each microsegment has a unique ID).

7 Conclusion and Future Work

In this paper we presented a prototype system that will eventually be used to (re)construct an entire mouse cortical network containing on the order of 16M neurons. Here, we built a coarse cortical network using N-PRM and L-System neuron generators. We tested three bounding volume methods for synapse identification. We determined that the approximate convex hull bounding volume was the fastest overall (considering both the cost of the abstract and the actual synapse discovery phases); its drawback is the increased storage requirements for the convex hull as compared to the bounding sphere or box.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

[2] G. Ascoli and J.L. Krichmar. L-Neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, 2000.

[3] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hull. Technical Report GCG53, Geometry Center, Univ. of Minnesota, July 1993.

[4] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1018–1023, 1999.

[5] J.M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* Springer-Verlag, 1998.

[6] V. Braitenberg and A. Schuz. *Cortex: Statistics and Geometry of Neuronal Connectivity.* Springer-Verlag, 1998.

[7] Duke/Southampton. Archive of neuronal morphology. `http://www.cns.soton.ac.uk/~jchad/cellArchive/cellArchive.html`.

[8] A. Hodgkin and A.Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117:500–544, 1952.

[9] D. Hsu, L. Kavraki, J-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[10] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[11] T. Lozano-Pérez and M. A. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacle. *Communications of the ACM*, 22(10):560–570, October 1979.

[12] B.H. McCormick and K. Mulchandani. L-system modeling of neurons. In *Proc. Visualization in Biomedical Computing, SPIE Vol. 2359*, pages 693–705, 1994.

[13] V. B. Mountcastle. *Perceptual Neuroscience: The Cerebral Cortex.* Harvard University Press, 1998.

[14] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 948–953, 2001.