UNIVERSITY OF CALIFORNIA, SAN DIEGO

Investigation of neurotransmitter diffusion in three-dimensional reconstructions of hippocampal neuropil

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Bioengineering

by

Justin P. Kinney

Committee in charge:

Professor Terrence J. Sejnowski, Co-chair
Professor Gabriel A. Silva, Co-chair
Professor Mark H. Ellisman
Professor Andrew D. McCulloch
Professor Charles F. Stevens

2009

The dissertation of Justin P. Kinney is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-chair

_____
Co-chair

University of California, San Diego

2009

To my parents, Susan and Larry, for instilling in me the value of education and hard work, and to my wife, Kate, for her patience and support during these last six years.

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Table of Contents v

List of Figures viii

List of Tables xi

Acknowledgments xii

Vita, Publications, and Fields of Study xiv

Abstract of the Dissertation xv

I Insights into the nature of the extracellular space 1
1. Abstract 1
2. Introduction 2
3. Methods 5
4. Results 11
5. Discussion 18
6. Conclusions 20
7. Acknowledgements 22

II Generating High-Quality Reconstructions of Neural Tissue for Morphometric Analysis and Modeling 23
1. What a mesh! Data pipeline deconstructed. 24
A. Data Acquisition: tissue to gray-scale voxels 24
B. Segmentation and Annotation: gray-scale voxels to annotated contours 26
C. Smoothing the contours: splining and sampling annotated contours 28
D. Surface Generation: annotated contours to annotated raw meshes 30
E. Surface Improvement: annotated raw meshes to annotated processed meshes 31
F. MESHMORPH: recovering the extracellular space 33
2. A CONTOUR_TILER implementation for neuropil reconstruction . 36
A. New output data format 37
B. Self-intersecting contours: detect and prohibit 38
C. High-frequency spatial perturbations 38
D. Bubbles versus pockets: adding a new capping option 39
E. Contour annotation: achieving a flexible merge policy 42

F. manifold edges: make it an option . . . 46
G. Scale-dependent mesh behavior . . . 48
3. Future Work . . . 48
A. Metadata: Part 2 . . . 49
B. File Formats . . . 50
4. Acknowledgements . . . 51

III MESHALYZER: a mesh quality analyzer . . . 52
1. Introduction . . . 52
2. Assumptions . . . 54
3. Usage Example . . . 55
4. Quick-Reference Guide . . . 67
5. Acknowledgements . . . 69

IV Monte Carlo simulation of glutamate spillover in simplified model of hippocampal neuropil . . . 70
1. Abstract . . . 70
2. Introduction . . . 71
3. First Generation 3D Model . . . 73
A. Model Design . . . 74
B. Achieving Nonzero Background Glutamate Concentration . . . 78
C. Decay Rate of Vesicular Glutamate with 0.5 μM Background Glutamate . . . 83
D. Glutamate Diffusion Extent and Decay Rate with Zero Background Glutamate . . . 85
E. Burst Release Simulations with Zero Background Glutamate Concentration . . . 86
F. Motivation of Second Generation Model . . . 93
4. Second Generation 3D Model . . . 93
A. Design Improvements Inspired By First generation Model . . . 94
B. Spillover Simulations . . . 96
5. Acknowledgements . . . 100

A Supplementary Material . . . 102
1. Steady-State Analysis of Glutamate Transporter and Receptor Kinetic Models . . . 102
A. Steady-State Glutamate Transporter Distribution and Transport Rate . . . 102
B. Steady-state AMPA and NMDA Receptor Distribution . . . 104
2. Tortuosity . . . 107
3. First-Generation Model Detail Drawings . . . 109

B Streamlining the Reconstruction Pipeline . . . 111
1. Original CA1 Reconstruction Pipeline . . . 112
2. Pipeline Improvement Phase 1 . . . 114
3. Pipeline Improvement Phase 2 . . . 116
4. Pipeline Improvement Phase 3 . . . 117

C Review of Neurotransmitter Spillover . . . 118
1. Experimental Evidence for Spillover . . . 119
2. Previous Models of Spillover . . . 124
3. Outstanding Questions . . . 127
4. Quantal content of EPSC based on coefficient of variation . . . 128

D Future Work . . . 131
1. Rules for adding AMPARs to spines . . . 131
2. Rules for adding NMDAR subtypes (NR2A and NR2B) . . . 133
3. NMDA Receptor Kinetic Models . . . 136
4. Add glutamate receptor subtypes (GLT-1 and GLAST) . . . 137
A. Simplification of Glutamate Transporter GLT-1 Kinetic Model . 141
5. Add neuronal transporters . . . 146

E Software Encyclopedia . . . 147
1. RECONSTRUCT3D . . . 147
2. CONTOUR_TILER . . . 148
3. reconstruct2contourtiler . . . 148
4. DReAMM . . . 150
5. FILTERMESH . . . 151
6. IRIT . . . 152
7. MESHALYZER . . . 154
8. meshalyzerxxl . . . 154
9. mesh_tools . . . 154
10. MESHMORPH . . . 161
11. Netgen . . . 162
12. TetGen . . . 162
13. GNU Triangulated Surface Library . . . 163
14. Blender . . . 163

References . . . 164

LIST OF FIGURES

Figure I.1 Tissue shrinkage occurs during preparation for EM imaging. . . . 4
Figure I.2 Renderings of neuropil reconstruction instrumented to measure geometric tortuosity. . . . 7
Figure I.3 Histogram of extracellular widths in the neuropil reconstructions. 12
Figure I.4 Reconstructions possess similar degrees of extracellular width uniformity. . . . 13
Figure I.5 Extracellular volume fraction covaries with median extracellular width in reconstructions . . . 15
Figure I.6 Geometric tortuosity increases as extracellular volume fraction decreases. . . . 16
Figure I.7 Macromolecules in ECS impede glutamate diffusion by 40%. . . . 17
Figure I.8 Finite membrane bend radius requires tunnels at junctions of three or more cells. . . . 21

Figure II.1 Lipid membrane orientation affects contour placement accuracy. . 28
Figure II.2 Radius of curvature dependent spline sampling. . . . 29
Figure II.3 Spline deviation statistics. . . . 30
Figure II.4 Vertex displacement is determined by net force of adjacent springs. 34
Figure II.5 Simple example of self_intersecting contour. . . . 38
Figure II.6 Sharp, thin protrusions punctuate the mesh surface. . . . 40
Figure II.7 Two different 3D geometries have identical contours. . . . 41
Figure II.8 Two capping paradigms. . . . 41
Figure II.9 Example cases of contour annotation. . . . 43
Figure II.10 Typical astrocyte contour geometry creates undesirable contour merging. . . . 45
Figure II.11 Schematic of Nonmanifold Surface Mesh. . . . 46
Figure II.12 Contours with small overlaps can generate nonmanifold meshes. . 47
Figure II.13 Example of a nonmanifold mesh from CONTOUR_TILER. . . . 47
Figure II.14 Scale-dependent mesh geometry. . . . 49

Figure III.1 Arbitrary convention for face normal vector in MESHALYZER. . 55
Figure III.2 Example surface mesh of a reconstructed dendrite (d014.mesh). . 56
Figure III.3 Examples of surface mesh attributes ‘closed’, ‘open’, ‘manifold’, and ‘nonmanifold’. . . . 59
Figure III.4 The convention for computing face normals also implies a method for determining edge traversal. . . . 60
Figure III.5 Orientation of a closed surface mesh with consistently-oriented face normals can be either (A) inwards or (B) outwards. . . . 61
Figure III.6 Self-intersecting meshes. . . . 64
Figure III.7 Equilateral triangular meshes are superior to high aspect ratio meshes. . . . 66
Figure III.8 Edge angle is defined by adjacent faces of edge. . . . 67

Figure IV.1 Definition of Crosstalk 72
Figure IV.2 First generation neuropil model was designed from morphometric values gleaned from literature. 75
Figure IV.3 Kinetic models for AMPARs, NMDARs, and Excitatory Amino Acid Transporter. 77
Figure IV.4 Measurement of geometric tortuosity for first generation neuropil model. 79
Figure IV.5 Glutamate release and uptake by transporters determine steady-state glutamate concentration in model. 81
Figure IV.6 Vesicular glutamate decay time course is insensitive to motif asymmetry. 85
Figure IV.7 Motif assymetry affects spatial spread of diffusing glutamate. . . . 86
Figure IV.8 Glutamate transporters dramatically diminish spillover but only marginally affect glutamate time course in the active synapse. . . 88
Figure IV.9 Glutamate transporters dramatically diminish NMDAR-mediated crosstalk but only marginally affect AMPAR or NMDAR activation in the active synapse. 89
Figure IV.10 NMDAR occupancy approaches saturation during the burst, while AMPAR occupancy does not. 90
Figure IV.11 Desensitization during burst release for AMPARs and NMDARs. 91
Figure IV.12 Distribution of glutamate transporters in T2 state reveals that glutamate diffuses farther each release in burst. 92
Figure IV.13 Burst release of glutamate exhibits accumulating transporter occupancy near release site. 93
Figure IV.14 Second-generation neuropil model incorporates variable glutamate transporter surface density on glial membranes. 95
Figure IV.15 Motif is representative of average hippocampal neuropil morphology as reported in literature. 96
Figure IV.16 Snapshot renderings of MCell simulation demonstrates diffusing glutamate cloud. 97
Figure IV.17 Uniform- and Variable-density glutamate transporter models differ little in transporter occupancy. 98
Figure IV.18 Spillover is not significantly different in Uniform- and Variable-density glutamate transporter models. 99
Figure IV.19 Uniform- and Variable-density glutamate transporter models differ little in mean radial glutamate diffusion distance. 100
Figure IV.20 3D rendering of glutamate transporter occupancy and NMDAR crosstalk in variable-density neuropil model. 101

Figure A.1 Schematics of First Generation Model 110

Figure D.1 Impact of vesicle size and NMDAR subunit distribution on number of open NMDARs. 136

Figure D.2 Kinetic models for NMDA receptors of (A) NR2A and (B) NR2B subtypes for third generation neuropil model. 138
Figure D.3 Kinetic models for (A) GLAST and (B) GLT-1 glutamate transporters to be used in third-generation neuropil model. 140
Figure D.4 Original Kinetic Model for Glutamate Transporter GLT-1 142
Figure D.5 Simplified GLT-1 Kinetic Model. 144

LIST OF TABLES

Table I.1	Symbols.	10
Table I.2	Membrane Area and ECS Volume of Reconstructions.	14
Table II.1	Summary analysis of Figure II.9.	44
Table IV.1	Rate Constants for AMPA Receptor.	76
Table IV.2	Rate Constants for NMDA Receptor.	76
Table IV.3	Rate Constants for Glutamate Transporter (EAAT).	78
Table IV.4	Influence of 0.5 μM background glutamate concentration on receptor and transporter state distributions.	84
Table IV.5	Half-lives of the contents of single vesicles.	86
Table D.1	Constants for NMDA Receptors of NR2A (L-Mode) Receptor Kinetic Model.	137
Table D.2	Constants for NMDA Receptors of NR2B Receptor Kinetic Model.	139
Table D.3	Rate Constants for GLAST transporter kinetic model.	141
Table D.4	Rate Constants for simplified GLT-1 transporter kinetic model.	141
Table D.5	Rate constants for original glutamate transporter GLT-1 kinetic model	143
Table D.6	Rate Constants for Simplified Glutamate Transporter GLT-1 Kinetic Model	146

ACKNOWLEDGMENTS

Terry gave me the opportunity to pursue our scientific questions under the auspices of my own design surrounded by clever colleagues who taught me so much. As a role model Terry kindled in me the philosophy that through relentless hard work and an open mind knowledge can be gleamed. Terry's unbridled energy in the pursuit of truth combined with his open devotion to all who enter his sphere of influence has made me a true Terry fan. For all of that I am truly grateful.

I would like to thank Tom Bartol for being my mentor and friend. His unquenchable optimism and passion for science inspired me to continue the search for truth even during the darkest days. Tom selflessly gave his time and energy (even though both were in high demand) to anyone who needed it including me. His encyclopedic knowledge and phenomenal memory for numbers helped me stay on the straight and narrow path. For all of that I am truly grateful.

I thank Jed Wing for taking me under his...wing and teaching me almost everything I know about computer programming. Jed's collegial spirit and sharp wit were a pleasure. He generously gave his time and knowledge and for all of that I am truly grateful.

I thank Rex Kerr for giving me the original concepts behind meshmorph and checking on my progress each time he visits the lab. I would like to extend my appreciation to Dan Keller and Vladimir Mitsner for helping me think through problems whenever I was stuck. They were often my sounding board for ideas and saved me from wasted effort on more than one occasion. For all of that I am truly grateful.

I would also to like to give a big thanks to all of the System Administrators - Chris Adams, Jorge Aldana, Chris Hiestand, Bryan Neilsen, and Chris Uebelher. Without them the computational modeling described in this document would have been immensely more difficult if not impossible. First, they provided a fantastic computing environment so well designed that I was free to focus on the science

questions and did not have to worry about the minutiae of system maintenance. By maintaining a computer cluster that rivaled the power of entire departments at universities we were able to tackle larger problems and undertake audacious research initiatives that proved successful due in large part to the integrity of the compute resources. As if that were not enough, they also responded to a myriad of technical questions that arose during the research and were always able to find an answer. It is their professionalism, keen intellect, and willingness to help for which I am truly grateful.

Chapter 1 and 2 are, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Josef Spacek, Thomas M. Bartol, Chandrajait Bajaj, Kristen Harris, and Terrence J. Sejnowski, supervised and directed the research which forms the basis of these chapters.

Chapter 3 is, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Thomas M. Bartol and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

Chapter 4 is, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Frederic D. Broccard, Thomas M. Bartol and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

VITA

1998	B.Sc., Mechanical Engineering, Georgia Institute of Technology
2000	M.Sc., Mechanical Engineering, Georgia Institute of Technology
2002–2009	Research Assistant University of California, San Diego
2009	Ph.D., Bioengineering University of California, San Diego.

ABSTRACTS

Kinney JP, Spacek J, Bartol TM, Bajaj C, Harris K, Sejnowski TJ, 2007: Effect of extracellular space width on geometric tortuosity in 3D reconstructions of neuropil, *Society for Neuroscience Abstracts*, program no. 360.19.

Kinney JP, Bartol TM, Sejnowski TJ, 2006: Influence of background glutamate on synaptic transmission in a Monte Carlo model of hippocampal neuropil, *Society for Neuroscience Abstracts*, program no. 794.12.

Kinney JP, Bartol TM, Sejnowski TJ, 2005: Influence of glutamate transporters on spillover in a Monte Carlo model of hippocampal neuropil, *Society for Neuroscience Abstracts*, program no. 731.6.

Kinney JP, Bartol TM, Sejnowski TJ, 2004: Monte Carlo simulation of glutamate spillover, *Society for Neuroscience Abstracts*, program no. 952.12.

ABSTRACT OF THE DISSERTATION

Investigation of neurotransmitter diffusion in three-dimensional reconstructions of hippocampal neuropil

by

Justin P. Kinney
Doctor of Philosophy in Bioengineering
University of California, San Diego, 2009
Professor Terrence J, Sejnowski, Co-chair
Professor Gabriel A. Silva, Co-chair

A comprehensive explanation of neuronal function requires that we understand how cellular structure seen in the brain shapes neuronal activity. For instance hippocampal CA1 synapses do not confine glutamate to the cleft following vesicular release of neurotransmitter but instead allow glutamate to diffuse out into the extracellular space, a phenomenon called "spillover". Combining accurate representations of the cellular structure with Monte Carlo simulations of glutamate diffusion in the extracellular space following vesicular release allows us to investigate where glutamate goes after it is released into a synapse.

Here we describe a process for the creation of three-dimensional reconstructions of neuropil from two-dimensional EM images. We employed the method to generate three-dimensional reconstructions of the extracellular space from electron microscopy images and subsequent corrections informed by in vivo morphological parameters reported in the literature. Quantitative measurements of the reconstruction are consistent with reports that fixed tissue is shrunken compared to in vivo state with an especially large reduction in extracellular volume fraction. The reconstruction most likely to reflect in vivo conditions has an extracellular volume fraction of 22%, median extracellular width of 40 nm, and glutamate diffusion constant in the extracellular space of 4.5e-6 cm^2/sec.

As a prelude to simulations of spillover in the reconstruction we constructed a simplified three-dimensional model of hippocampal neuropil and used the model to perform Monte Carlo simulations of spillover following high-frequency burst release of neurotransmitter. The mean radial diffusion distance of a quantum of transmitter was independent of quantal size over the range tested. More than 90% of the diffusing neurotransmitter stays within 2 μm of the release site after synaptic vesicular release. Glutamate transporters suppress NMDAR spillover activation almost entirely, while AMPAR spillover activation is nonexistent with or without transporters. Glutamate transporters are not saturated with vesicle size of 3000 glutamate even after 100Hz burst in either model. Our results suggest that glutamate spillover is insignificant in neuropil models with canonical geometry and can be ignored. However, it remains to be seen whether spillover is relevant in the heterogeneous milieu of real neural tissue.

I

Insights into the nature of the extracellular space

I.1 Abstract

One-fifth of the brain's volume is extracellular space. Surprisingly little is known about the morphology of this space which serves as an important channel for volume communication between cells. Our task was to integrate disparate observations of ECS morphology from iontophoretic measurements and microscopy data into a cohesive model to provide improved estimates of the ECS geometry in vivo. This document describes our method for generating three-dimensional reconstructions of the extracellular space from electron microscopy images and subsequent corrections informed by in vivo morphological parameters reported in the literature. Quantitative measurements of the reconstruction are consistent with reports that fixed tissue is shrunken compared to in vivo state with an especially large reduction in extracellular volume fraction. The reconstruction most likely to reflect in vivo conditions has an extracellular volume fraction of 22%, median extracellular width of 40 nm, and glutamate diffusion constant in the extracellular space of 4.5e-6 cm^2/sec. The extracellular space in the processed reconstruction is best modeled as uniform-width sheets with near uniform extracellular width

punctuated by an interconnected network of tunnels with diameter slightly larger than the sheet width. Our results suggest that morphometric analysis of neural tissue and dynamic biochemical simulations of neuronal activity should account for possible shrinkage in tissue geometry.

I.2 Introduction

The extracellular space (ECS) is an important determinant of cellular signaling given that it serves as a communication channel between neighboring cells. For example, activation of metabotropic receptors and extrasynaptic NMDA receptors occurs via neurotransmitter diffusion out of the synaptic cleft (Scanziani et al. (1997), Chen and Diamond (2002)). Ectopic neurotransmitter release implies diffusion of transmitter through the ECS to its receptor in neighboring active zones (Coggan et al., 2005). The cerebellar glomerulus morphology encourages widespread diffusion of neurotransmitter through the ECS after vesicular release (Nielsen et al., 2004). The ECS also serves as an ion reservoir important for electrically active neurons (Hodgkin and Huxley, 1952). Change in extracellular concentration of various ions during periods of high neuronal activity (Moody et al. (1974), Bellinger et al. (2008)) likely depends on local ECS volume. Understanding these diverse phenomena requires accurate knowledge of ECS morphology.

The ECS morphology is dictated by a balance of forces with diverse origins. The intracellular cytoskeleton which resists compressive forces provides a rigid framework to the cell (Ingber et al., 1994). To the cytoskeleton is attached the cell membrane which experiences hydrostatic pressure depending on the osmolarity of the extracellular fluid and the activity of membrane-bound ion pumps (Alberts et al., 2002). Finally, membrane-bound extracellular proteins such as integrins help anchor the cells to their environment (Sabouri-Ghomi et al., 2008). While much is known about these structural components of the cell, predicting the geometry of the ECS using models of these force balances is still a challenge.

For now we must look to alternative sources of information about the ECS such as imaging methods.

The width of the ECS is small - on the order of tens of nanometers - and no imaging method yet exists to accurately measure the geometry of the ECS *in vivo*. Magnetic resonance imaging with a spatial resolution of microns (Xu et al., 2008) is too course, and even the highest resolution light-microscopes with diffraction-limited submicron pixels (Mainen et al., 1999) lack sufficient resolution. The high vacuum requirements of electron microscopy (EM) precludes its use on living tissue.

Measurement of the ECS in fixed-tissue slices is feasible using EM which delivers nanometer voxel resolution in the plane of the image (Harris et al., 2006). Resolution perpendicular to the plane of the cut is determined by the slice thickness typically on the order of 50 nanometers. Unfortunately, the slice thickness is large relative to the minimum features sizes of the ECS thus hampering accurate determination of ECS morphology. In a typical experiment to gather EM data an animal is anesthetized, perfused with a mixed aldehyde fixative, and sacrificed. A block of tissue is excised, postfixed in osmium, stained, dehydrated in alcohol, and embedded in an epoxy resin. The tissue block is then cut into thin sections and imaged using an electron microscope (Jensen and Harris, 1989).

Potential artifacts introduced during any of the steps of the process obscure the true nature of the ECS when visualized in EM sections (Figure I.1). A brief period of anoxia results in cellular swelling at the expense of ECS (Fox et al., 1985). Rapid freeze-substitution fixation of the superficial layers of cerebellum reveals appreciable extracellular space between axons; whereas after eight minutes of anoxia little to none extracellular space is seen (van Harreveld et al., 1965). Tissue fixation using aldehydes involves the crosslinking of proteins and 4% tissue shrinkage (Fox et al., 1985). Most significantly dehydration can result in 12.5% shrinkage in linear dimension of the tissue (Fox et al. (1985), Kirov et al. (1999) but see Schuz and Palm (1989)). Loss of ECS is posited to account for some of

the shrinkage (Kirov et al., 1999). Finally, compression, stretching, and skew of the thin sections during slicing and handling can deform the ECS (Harris et al., 2006). Each of these artifacts of the fixation process undermine the estimation of ECS morphology in vivo based on measurements of fixed tissue. Fundamental aspects of the changes to tissue morphology that occur during shrinkage are still unknown. Is shrinkage uniform across cell types? Do intracellular and extracellular spaces shrink equal amounts? Does neural tissue shrink isotropically? How are cell membranes which are composed of lipid bilayer and membrane-bound proteins deformed during shrinkage?

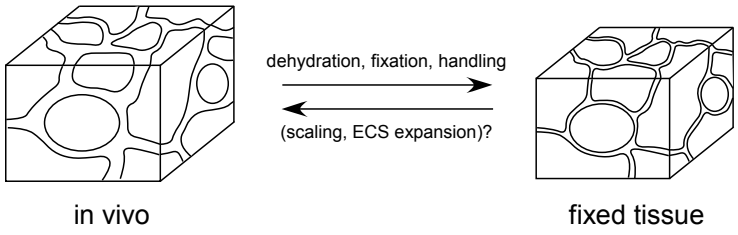

Figure I.1 **Tissue shrinkage occurs during preparation for EM imaging.** Shrinkage is caused by anoxia, fixation, dehydration, and tissue handling. Could linear scaling and selective expansion of the extracellular space recover the original morphology?

Fortunately, some aspects of the ECS have been measured in vivo. The extracellular volume fraction (EVF)[1] in cortex has been estimated from experiments that measure the rate of diffusion of small probe molecules through the ECS (Nicholson and Phillips, 1981). By choosing an inert probe molecule such as tetra-methyl-ammonium that is not transported inside cells, the measured diffusion profile is only affected by the morphology of the ECS. These experiments provide in vivo estimates of the EVF (22%) and the total impedance to diffusion (1.6) of the probe molecule. The latter measurement is also called the total tor-

[1]EVF is the fraction of total tissue volume that lies outside of cells.

tuosity (λ_t) referring to the increase in diffusional distance imposed by the ECS barriers at both the nanoscopic and microscopic scales. Additionally, the extracellular width (ECW) has been estimated (35-64 nm) in vivo by tracking the diffusion of quantum dots of known size through the ECS (Thorne and Nicholson, 2006).

Our task was to integrate disparate observations of ECS morphology into a cohesive model to provide improved estimates of the ECS geometry in vivo. This document describes our method for reconstructing the ECS from fixed tissue and subsequent corrections informed by in vivo morphological parameters reported in the literature. We generated multiple three-dimensional (3D) reconstructions of a region of rat hippocampal neuropil from EM serial sections. The EVF in the reconstruction was 9%, less than half of the estimated fraction in vivo. Custom software was written to manipulate the 3D reconstruction and recover the lost ECS increasing the EVF to 0.22 as estimated in vivo. We also explore the utility of linearly scaling the reconstruction to account for volume shrinkage. Assuming the ECW is approximately uniform, we provide an estimate of the extracellular width (40 nm) that compensates for shrinkage artifacts. Furthermore, by simulating diffusion of glutamate through the ECS of the reconstruction, we estimate the diffusion constant of glutamate in the ECS and show a 40% reduction in diffusion rate compared to a pure aqueous environment.

I.3 Methods

We began by generating a 3D reconstruction of 180 cubic microns of adult rat (postnatal day 77) hippocampal neuropil in CA1 stratum radiatum. Perfusion-fixed tissue was stained and cut in 50 nm thick sections before being photographed with transmission electron microscopy through serial sections. The membranes of every dendritic, axonal, and glial process in each section were manually traced from the micrographs at a resolution of 2.3 nm per pixel using RECONSTRUCT3D (`synapses.clm.utexas.edu/tools/reconstruct/reconstruct.stm`). The set of

contours from each process were used to reconstruct the surface of the plasma membrane. The 3D computer model of neural tissue consists of water-tight, triangulated surfaces representing the outer surface of the lipid bilayer of each process in the volume.

We wrote custom software to recover the ECS by manipulating the location of the surfaces on the nanometer scale. The software implements a simple model of forces on the membrane surface involving springs between neighboring regions on a surface and between surfaces on different neighboring processes. The collection of all springs describes a potential energy system which is relaxed to determine the final location of the surfaces. The spring forces will tend to generate an ECS with uniform width controllable by a user-specified target value. The effect of extracellular width on the extracellular volume fraction was investigated by sweeping the extracellular width to generate different configurations of the reconstruction. For each configuration of the model the extracellular volume fraction, median extracellular width, and geometric tortuosity were measured.

Tortuosity

On the *micron* scale the morphology of the ECS is dictated by the twisting and turning of the cell membranes. Looking much like a plate of spaghetti, the convoluted structure of neuropil demands that diffusing molecules travel a farther distance between two locations in the ECS compared to a free environment, an attribute referred to as geometric tortuosity (λ_g). The geometric tortuosity of each configuration of the reconstruction was calculated by measuring the diminished rate of diffusion in MCell simulations (Stiles and Bartol, 2001) of glutamate molecules in the ECS. The effective rate of diffusion of ten thousand molecules through the ECS was measured using concentric sampling boxes centered on the release site (Tao and Nicholson, 2004). The largest sampling box (8.5 μm on a side) was located fully within the reconstruction. The MCell simulation ended when 4% of the molecules had diffused out of the largest sampling box to minimize boundary

effects (Figure I.2A). To maximize the utility of the reconstruction three reflective planes were used to artificially increase the volume of the reconstructed neuropil eight fold (Figure I.2B). This was necessary since the particles had to sample a large enough subset of the reconstruction geometry by diffusing several microns to yield accurate and precise estimates of the geometric tortuosity. Furthermore, a large apical dendrite centered in the reconstructed volume precluded glutamate release in the center of the reconstruction. The site of molecule release was located a mere 250 pm away from the point of intersection of the three mirror planes so that the cloud of molecules undergoes simulated diffusion as if from a single point source.

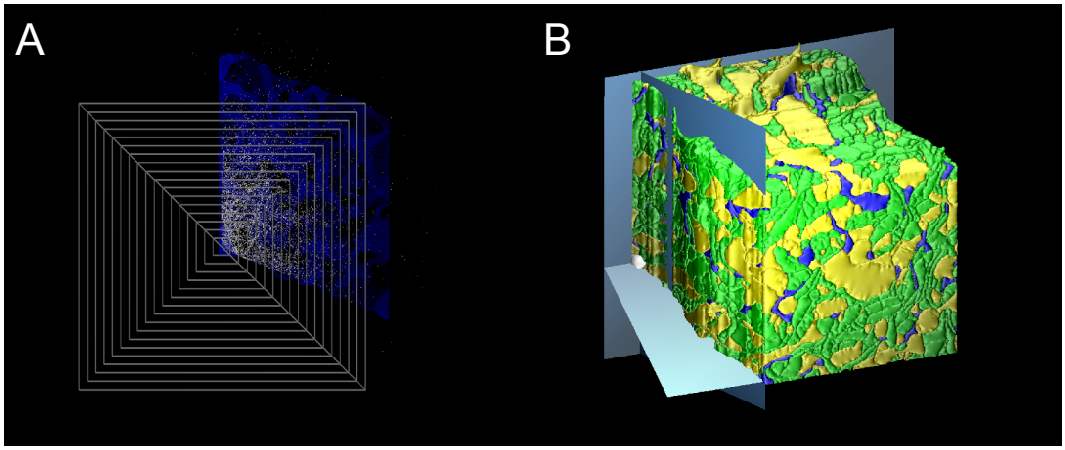

Figure I.2 **Renderings of neuropil reconstruction instrumented to measure geometric tortuosity.** (A) Sampling boxes measured glutamate shown here in full extent of diffusion for tortuosity measurement. Largest sampling box is 8.5 μm on a side. (B) Three mirror planes artificially expanded geometry around site of glutamate release (white).

The rate of molecule diffusion through the ECS is also diminished relative to a free environment due to macromolecules obstructing the diffusion path on the *nanometer* scale. The diffusion constant of a molecule in the ECS can be calculated from its free diffusion constant in water (7.5 cm^2/sec, Longsworth (1953)) and the

ratio of geometric to total tortuosity in the ECS (Sykov and Nicholson, 2008):

$$D_{ECS} = D_{free}\left(\frac{\lambda_g}{\lambda_t}\right)^2. \tag{I.1}$$

Tissue shrinkage

Before tissue preparation for EM imaging the original tissue volume V is composed of extracellular space, V_E, intracellular space, V_I, and membrane volume. Because the membrane lipid bilayer is incompressible (Wolfe and Bryant, 1999), its volume is assumed constant. After surgery, fixation, staining, dehydration, and imaging the shrunken tissue volume $V^{'}$ reflects a diminished extracellular space, $V_E^{'}$, and diminished intracellular space, $V_I^{'}$. Let the overall tissue volume shrinkage be captured by the ratio $V^{'}/V = \gamma$. Since $V^{'} < V$, then $0 < \gamma < 1$. By convention the extracellular volume fraction, V_E/V, is labeled α, and likewise, $V_E^{'}/V^{'} = \alpha^{'}$. Note that $0 < \alpha < 1$ and $0 < \alpha^{'} < 1$.

With these few quantities we can derive several interesting measurements of the ECS before and after preparation for EM. For instance, the fraction of extracellular space remaining after shrinkage, $V_E^{'}/V_E$, is equal to $V^{'}\alpha^{'}/V\alpha$ or

$$\alpha^{'}\gamma/\alpha. \tag{I.2}$$

The fraction of intracellular space remaining after shrinkage, $V_I^{'}/V_I$, is equal to $(1-\alpha^{'})V^{'}/(1-\alpha)V$ or

$$(1-\alpha^{'})\gamma/(1-\alpha). \tag{I.3}$$

The volume of extracellular space lost during shrinkage, $V_E - V_E^{'}$, is equal to $V\alpha - V^{'}\alpha^{'}$, and the volume of intracellular space lost during shrinkage, $V_I - V_I^{'}$, is equal to $V(1-\alpha) - V^{'}(1-\alpha^{'})$. Therefore, the ratio of extracellular volume lost during shrinkage to intracellular volume lost during shrinkage is $(V\alpha - V^{'}\alpha^{'})/(V(1-\alpha) - V^{'}(1-\alpha^{'}))$ or

$$(\alpha - \gamma\alpha^{'})/((1-\alpha) - \gamma(1-\alpha^{'})). \tag{I.4}$$

Scaling the shrunken tissue volume by $1/\gamma$ recovers the original tissue volume ($V^{'}/\gamma = V$) but the scaled extracellular volume fraction does not reflect in vivo conditions as shown here. The scaled intracellular volume, $V_I^s = V_I^{'}/\gamma$, is related to the original intracellular volume: $V_I^s = V_I(1-\alpha^{'})/(1-\alpha)$. Similarly, the scaled extracellular volume, $V_E^s = V_E^{'}/\gamma$, is related to the original extracellular volume: $V_E^s = V_E\alpha^{'}/\alpha$. Given observations that the extracellular volume fraction decreases disproportionately with tissue shrinkage (Kirov et al. (1999) and Figure I.5), then $\alpha^{'} < \alpha$. Consequently, $\alpha^{'}/\alpha < 1$, $1-\alpha^{'} > 1-\alpha$ and $(1-\alpha^{'})/(1-\alpha) > 1$. It follows that $V_I^s > V_I$ and $V_E^s < V_E$. In other words, scaling the shrunken tissue by γ yields the correct total tissue volume but overestimates the intracellular volume in vivo and underestimates the extracellular volume in vivo.

Scaling the shrunken tissue volume by $1/\gamma$ overestimates the in vivo membrane surface area in the tissue volume. To see why let us begin by making three strong assumptions about the tissue shrinkage. First, suppose the tissue shrinkage is isotropic. Second, assume no difference in volume shrinkage by cell type and, third, no gradients of shrinkage amount in the slice, such as greater volume shrinkage near cut faces. These assumptions are concessions so we can say that equal tissue shrinkage, $1/f = \sqrt[3]{\gamma}$, occurs along three mutually orthogonal axes. As a result scaling the tissue volume by $1/\gamma$ will scale the ECW by f: $ECW^s = ECW^{'} * f$. Furthermore, assume the extracellular width is uniform, so that the extracellular volume and total membrane area, A, are related: $V_E = ECW * A/2$. The membrane area of the scaled tissue, A^s, can be related to the membrane area of the shrunken tissue, $A^{'}$: $A^s = 2 * V_E^s/ECW^s = 2 * V_E^{'}/(\gamma ECW^s) = 2 * V_E^{'}/(\gamma ECW^{'} f) = A^{'}/(\gamma f)$. As mentioned before, a typical value for the linear tissue shrinkage is 12.5% which yields a volume shrinkage of 33% assuming isotropic contraction. Therefore, $\gamma \approx 0.67$ and $f \approx 1.15$. If finally we assume that the total membrane area in the volume decreases by 9% during tissue shrinkage ($A^{'}/A = \phi = 0.91$, Lis et al. (1982) pure lipids), then the relationship between the original membrane area and the scaled area is $A^s = A\phi/(\gamma f) \approx 1.18A$. The in vivo membrane area is

overestimated by 18% in the scaled reconstruction.

To recover the in vivo morphology more than mere scaling of the reconstruction is required. It is clear that the proportions of intracellular and extracellular volume must be adjusted also. To do this, we will use our custom software to manipulate the location of the membrane surfaces, expanding the extracellular space and contracting the intracellular space.

Table I.1 **Symbols.** Note that 'in sileo' refers to fixed tissue.

Symbol	**Description**
λ_t	total tortuosity
λ_g	geometric tortuosity
D_{free}	diffusion constant of glutamate in water
D_{ECS}	diffusion constant of glutamate in ECS
V	tissue volume in vivo
V_E	extracellular tissue volume in vivo
V_I	intracellular tissue volume in vivo
$V^{'}$	tissue volume in sileo
$V_E^{'}$	extracellular tissue volume in sileo
$V_I^{'}$	intracellular tissue volume in sileo
V^s	tissue volume scaled
V_E^s	extracellular tissue volume scaled
V_I^s	intracellular tissue volume scaled
α	extracellular volume fraction in vivo
$\alpha^{'}$	extracellular volume fraction in sileo
γ	tissue volume shrinkage
f	linear scaling factor
ECW	median extracellular width in vivo
$ECW^{'}$	median extracellular width in sileo
ECW^s	median extracellular width scaled
A	tissue membrane area in vivo
$A^{'}$	tissue membrane area in sileo
A^s	tissue membrane area scaled
ϕ	1 - tissue membrane area shrinkage

I.4 Results

The extracellular space of the raw (nonscaled, non-ECS-expanded) reconstruction deviated from in vivo conditions in several ways. First, the extracellular volume fraction of the raw reconstruction was 8%, well below estimates of the in vivo extracellular volume fraction (22%). Second, the histogram of extracellular widths in the raw reconstruction (Figure I.3) revealed that extracellular width was negative in some places, implying that surfaces in the raw reconstruction passed through each other. Obviously, intersecting cells is physically impossible and represents an artifact of the reconstruction process. The extracellular width of the raw reconstruction had a median value of 9 nm and was noticeably non-uniform. The uniformity of the extracellular width in vivo is unknown, but estimates in the literature of the mean ECW are around 20 nm (Rusakov and Kullmann (1998a) [17 nm], Barbour (2001) [20 nm], Wahl et al. (1996) [15 nm], but see Thorne and Nicholson (2006) [35 nm]).

Assuming a tissue volume shrinkage of 33% we can calculate the fraction of extracellular volume *lost* during processing as 0.75 (Equation I.2). Similarly we can calculate the fraction of intracellular volume lost during processing as 0.2 (Equation I.3). Interestingly, the ratio of extracellular to intracellular volume lost during shrinkage is 1.0 (Equation I.4). In other words, compared to in vivo conditions the raw reconstruction has a 75% reduction of extracellular space and 20% reduction of intracellular space, representing equal volume loss from both spaces.

To investigate strategies for correcting these deficiencies we generated multiple reconstructions by expanding the ECS of both the raw reconstruction and a copy scaled by $f = 1.15$. The diminished extracellular space in the raw and scaled reconstructions was grown incrementally in five steps, from 0.05 to 0.4, while maintaining uniform extracellular width (Figure I.3). As the ECS expanded, the median extracellular width of the reconstructions increased as well from 8 nm to 75 nm. The degree of uniformity of the ECW after each step was not equal

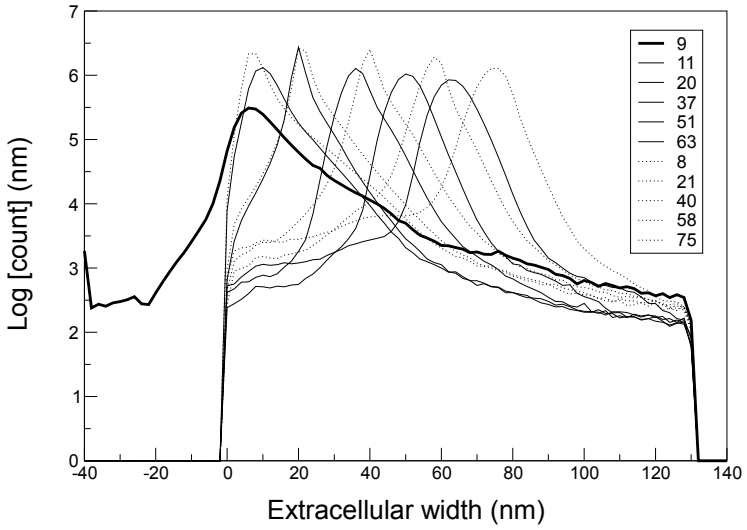

Figure I.3 **Histogram of extracellular widths in the neuropil reconstructions.** Beginning with the raw reconstruction (thick solid line) five new versions of the reconstruction were created (thin solid lines) by expanding the extracellular space. Five additional versions of the reconstruction were created by first linearly scaling the reconstruction along three orthogonal axes then expanding the extracellular space (thin dashed lines). The histogram of extracellular widths in the raw reconstruction reveal nonphysiological negative widths due to artifacts of the reconstruction process. An arbitrary upper limit of 130 nm on extracellular widths was enforced near the outer boundary of the reconstruction where the membrane surfaces had no opposing surface to define an extracellular width. The legend indicates the median extracellular width in nanometers of each reconstruction. The extracellular width in each reconstruction (other than raw) was roughly uniform. The modest differences in height of the reconstructions reflects varying degrees of ECW uniformity probably due to incomplete relaxation of the spring networks underlying the reconstructions.

(probably due to incomplete relaxation of the spring model) but always greater than the raw reconstruction.

A simple relationship between ECW, ECS volume, and membrane area

allowed comparison of ECW uniformity between reconstructions. Assuming the ECW is perfectly uniform then the ECS is exactly described by a sheet model with width given by $ECW = 2 * V_E/A$. The ECS volume and membrane area were measured for each reconstruction and used to calculate the width of the ECS were it uniform (Table I.2). Comparing the actual median ECW in the reconstructions to the estimated ECW (Figure I.4) we see that all reconstructions possess similar degrees of ECW uniformity and deviate from perfectly uniform width suggesting that the ECS is not well described by a sheet model.

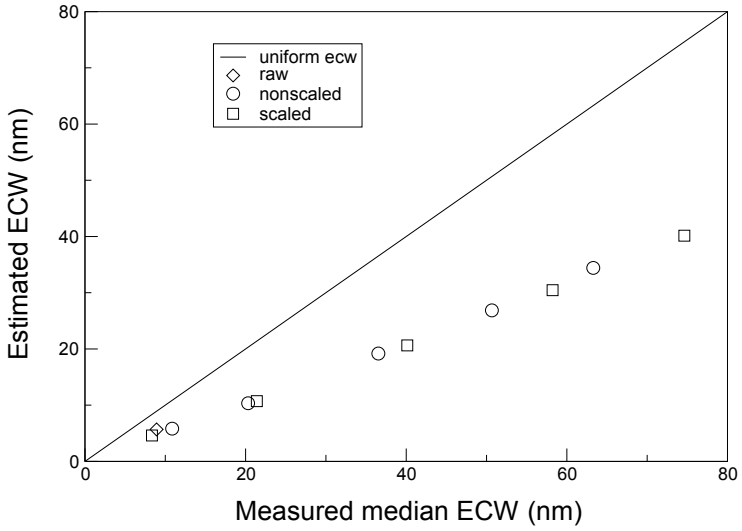

Figure I.4 **Reconstruction possess similar degrees of extracellular width uniformity.** The membrane area and ECS volume of each reconstruction were used to estimate the ECW of the reconstruction were the ECS perfectly uniform. The actual median ECW and estimated ECW are plotted for each reconstruction. The line of slope 1 defines perfectly uniform ECW. By visual inspection We see that both the scaled-and-ECS-expanded and ECS-expanded reconstruction data points lie along a straight line, iimplying that the discrepancies in height of the ECW histogram (Figure I.3) are negligible. Difference between the reconstructions and the line of slope 1 suggests that ECS is not well described by a sheet model.

As the extracellular space increased the extracellular volume fraction also

Table I.2 **Membrane Area and ECS Volume of Reconstructions.** 'ECW' is median width of ECS measured in nanometers measured from each surface at a density equivalent to tiling the surface with equilateral triangles of side length 40 nanometers and then measuring the ECW from the barycenter of each triangle. Membrane area has units of 1e9 square nanometers. Extracellular volume has units of 1e10 cubic nanometers. The raw reconstruction has a median ECW of 8.93 nm, a membrane area of 2.54e9 square nanometers, and 1.44e10 cubic nanometers.

	Nonscaled			**Scaled**	
ECW	**Area**	**Volume**	**ECW**	**Area**	**Volume**
10.9	2.32	1.35	8.3	3.12	1.43
20.3	2.28	2.35	21.4	3.03	3.24
36.5	2.08	3.99	40.1	2.79	5.75
50.7	1.99	5.34	58.2	2.63	8.02
63.3	1.90	6.52	74.6	2.50	10

increased from 0.05 to almost 0.4 (Figure I.5). The ECW varies quadratically with EVF (R-value=0.99), and as expected the fit passes close to the origin reflecting an extracellular width of zero when extracellular volume is reduced to zero. The raw reconstruction deviates from the other reconstructions likely because of non-uniform extracellular width. The measured in vivo extracellular volume fraction of 22% corresponds to an estimated extracellular width range of 36 to 42 nm.

The geometric tortuosity was calculated for each of the reconstructions by measuring the spread of 10000 molecules in Monte Carlo simulations of diffusion (Tao and Nicholson, 2004). Our values range from 1.18 to 1.31 and lie between previous estimates (Figure I.6). No significant difference was observed between the geometric tortuosity estimates of the scaled and nonscaled reconstructions. Both classes of reconstruction exhibited higher geometric tortuosities for smaller extracellular volume fractions. As expected the geometric tortuosity estimates are above the minimum allowed value (1, corresponding to free space) and below the maximum expected value (1.6, corresponding to total tortuosity measured in vivo).

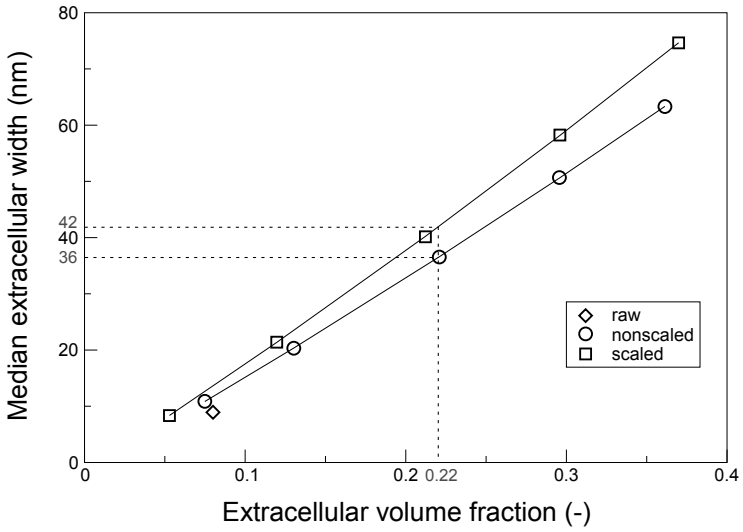

Figure I.5 **Extracellular volume fraction covaries with median extracellular width in reconstructions.** Best fit line is $EVF = A0 + A1 * ECW - A2 * ECW^2$ where (A0, A1, A2) equals (-1.5, 161.6, 49.7) for nonscaled reconstruction (R-value=0.99) and (-1.5, 184.5, 58.3) for scaled reconstruction (R-value=0.99). As expected the fit passes close to the origin reflecting an extracellular width of zero when extracellular volume is reduced to zero. The raw reconstruction (diamond) deviates from the other reconstructions likely because of non-uniform extracellular width. The measured in vivo extracellular volume fraction of 22% corresponds to an estimated extracellular width range of 36 to 42 nm, double previous estimates in literature.

The glutamate diffusion constants in the ECS required for a total tortuosity of 1.6 given the geometric tortuosity values in Figure I.6 were calculated using Equation I.1. Our values range from 4.1 to 5.1 (1e-6 cm^2/sec) and also lie between previous estimates of glutamate diffusivity in the ECS. No significant difference was observed between the glutamate diffusivity of the scaled and nonscaled reconstructions. Reconstructions with smaller extracellular volume fractions have higher geometric tortuosity which consequently require larger diffusion constants to achieve a total tortuosity of 1.6. As expected the estimates of glutamate dif-

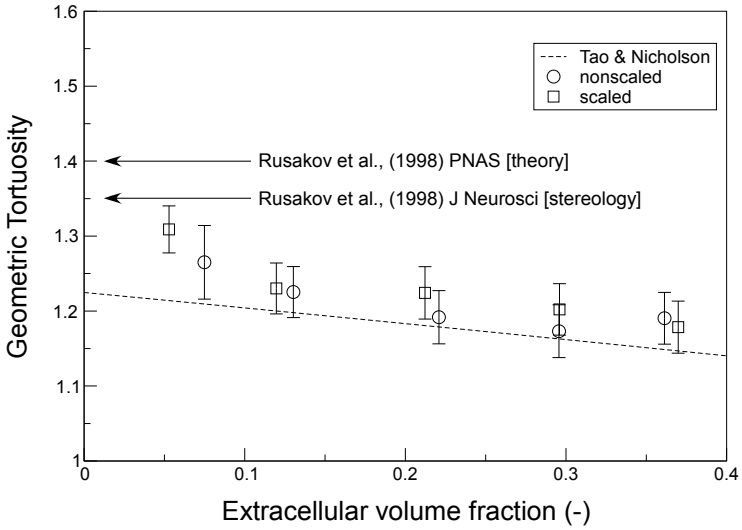

Figure I.6 **Geometric tortuosity increases as extracellular volume fraction decreases.** No significant difference was observed between the geometric tortuosity estimates of the scaled and nonscaled reconstructions. The quadratic relationship between geometric tortuosity and EVF ($\lambda_g = sqrt((3 - EVF)/2)$) described by Tao and Nicholson (2004) provides a lower bound (dashed line) to our data, while both stereological (Rusakov and Kullmann, 1998a) and theoretical (Rusakov and Kullmann, 1998b) estimates in the literature lie above. Error bars describe range of geometric tortuosity that explains 90% of difference between the actual time course and mean of glutamate molecules in each sampling box during simulation.

fusion constant in the ECS are above the minimum allowed value (0) and below the maximum allowed value (D_{free}=7.5e-6, corresponding to glutamate diffusion in water).

The membrane area of lipid membranes is known to decrease by 9% under compressive lateral forces (Lis et al., 1982), loading that might be experienced during tissue shrinkage. Is the membrane area that is lost during shrinkage later recovered after ECS expansion? Considering just the two reconstructions with EVF closest to 0.22, we get two very different answers to this question. The nonscaled

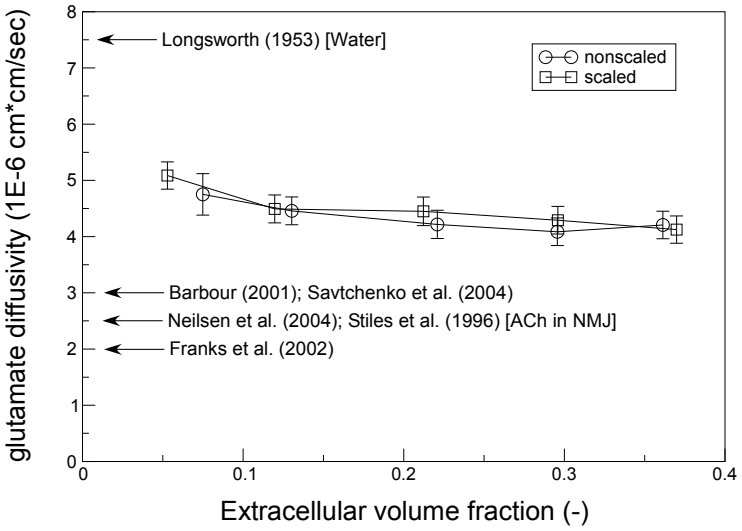

Figure I.7 **Macromolecules in ECS impede glutamate diffusion by 40%.** Reconstructions with smaller extracellular volume fractions have higher geometric tortuosity which consequently require larger diffusion constants to achieve a total tortuosity of 1.6. No significant difference was observed between the glutamate diffusivity of the scaled and nonscaled reconstructions. Our estimates of glutamate diffusion constant in the ECS are higher than previous estimates in literature (Barbour (2001), Savtchenko and Rusakov (2004), Nielsen et al. (2004), Stiles et al. (1996), Franks et al. (2002)).

reconstruction with EVF of 0.22 has membrane area of 2.08e9 square nanometers, 20% smaller than the membrane area in the raw reconstruction measured as 2.54e9 square nanometers. This further reduction in membrane area is difficult to interpret. However, the closest scaled reconstruction with EVF of 0.21 has a membrane area of 2.79e9 square nanometers, 10% larger than the reduced area of the raw reconstruction. This agrees well with surface shrinkage estimates of 9%, so membrane area lost during tissue volume shrinkage is recovered after scaling and ECS expansion.

I.5 Discussion

Sources of error in the reconstructions

The observed difference between the extracellular volume fraction (8%) of the raw reconstruction (before manipulating the location of the surfaces) and in vivo estimates (22%) could have arisen due to both errors in the reconstruction process and shrinkage of the tissue during preparation for EM imaging. Tissue preparation for EM presents numerous risks for altering the tissue geometry from dehydration to handling. The ECS in each image was hand-contoured by tracing the plasma membrane of every process. Simple deviations in the contour placement from the obvious center of the membrane can result in shrinkage or expansion of the reconstructed ECS. More certainly errors arise when the plasma membrane is difficult to discern and the proper placement of contours is no longer obvious. This situation frequently arises when a process is cut along its medial axis thereby causing its membrane to cast a wide faint shadow in the EM image. Even if the proper location of the contours on each slice is known (which was not the case), assumptions must be made during the reconstruction process about the location of the membrane between contours which inevitably lead to errors in the ECS. The severity of the errors is captured in the extracellular width histogram of the raw reconstruction which contains negative values indicating collisions between neighboring processes.

Our results still on a single reconstruction of a 5 x 5 x 5 μm volume of neural tissue, so replicating the findings by reconstructing additional tissue volumes is important. The particular tissue volume used here contains a large apical dendrite running down the middle and may not be representative of the hippocampus in general. Furthermore, our results suggest that characterizing geometric tortuosity requires at least a tissue volume 10 μm on a side for 90% confidence interval to be approximately 0.1 units in length on the tortuosity scale. We resorted to using reflective planes to artificially amplify the reconstruction which effectively

oversampled the data possibly introducing biases into our estimates of geometric tortuosity.

Linearly scaling the reconstruction geometry assumes isotropic shrinkage, no cell-type differences, and no systematic shrinkage gradients across the slice. Since little evidence supports these assumptions, skepticism of the scaled data is warranted. Fortunately, little difference was seen between the scaled and non-scaled reconstructions for median extracellular width or geometric tortuosity. Our hypotheses that the median ECW in vivo is near 40 nm and that the glutamate diffusion constant in the ECS reflects a 40% reduction compared to free diffusion still stand. One important difference between the two reconstruction classes is surface area. For instance, surface area in the scaled reconstruction with 21 nm median ECW was 30% larger than in the nonscaled reconstruction with comparable ECW (Table I.2). And, of course, due to the scaling all distances in the nonscaled reconstructions are 15% greater in the scaled reconstructions.

For both classes of reconstruction the width of the ECS was made as uniform as possible. Again there is little evidence for or against this assumption. Even the definition of uniformity is debatable depending on whether deviation of a few nanometers constitutes nonuniformity or if the width must swell to twice the median value to be considered nonuniform. Recent experimental work involving advanced dissection protocols with a cryogenically cooled knife argues against uniform ECW and for the existence of large voids in the ECS especially around synapses (Ohno et al., 2007). Along a different line of thought, the synaptic cleft is speculated to possess a different ECW than nonsynaptic regions owing to the high concentration of synaptic adhesion molecules in the cleft (Latefi and Colman, 2007). We measured the fraction of ECS in synaptic clefts in our reconstructions to be less than 2% suggesting that even if the ECW is different in synapses the contribution to tortuosity, either geometric or total, is negligible. The restriction of our investigation to uniform ECW was in part dictated by the technical challenge of systematically exploring deviations from uniformity. Perhaps future work will

allow a characterization of the effect of ECS lacunarity on tortuosity.

Pore and sheet model of neuropil

Despite our best attempts the width of the extracellular space was not perfectly uniform (Figure I.4). Visual inspection of the ECS-expanded reconstructions (data not shown) revealed an ECS structure composed of sheets with near uniform ECW between pairs of cells punctuated by tunnels (Thorne and Nicholson, 2006) with ECW larger than the median at the junction of three or more cells precisely as observed in EM images of mouse cerebellum following rapid freeze-substituion (van Harreveld et al., 1965). Tunnels in the reconstruction are caused by finite polygon size in the surface meshes but have a physiological basis in the finite bend radius of the cell membrane as dictated by the lipid bi-layer and cytoskeleton. The relationship between our mesh size and the biophysics of the cell membrane is not quantitatively accurate but qualitatively sound. Because plasma membrane has a minimum radius of curvature, the ECS has tunnels. At a minimum, a network of tunnels form at the intersection of three or more cellular processes Figure I.8. In between these tunnels lie sheets formed by the parallel membranes of two processes. These sheets can sometimes extend for hundreds of nanometers. What is unknown is whether tunnels form here also. We cannot say that we have found all possible tunnels in the reconstruction, but we have found tunnels - a minimal set morphologically dictated - that persist even when subjected to an energy relaxation protocol designed to remove them. ECS with uniform ECW can be modeled as an interconnected network of tunnels spanned by nearly uniform-width sheets.

I.6 Conclusions

We generated multiple 3D reconstructions of 180 cubic μm of rat hippocampal neuropil from EM serial sections in order to make quantitative mea-

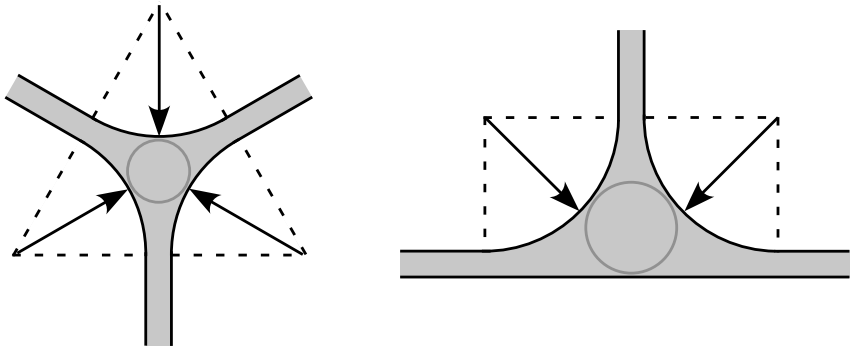

Figure I.8 **Finite membrane bend radius requires tunnels at junctions of three or more cells.** Our model of the ECS is an interconnected network of tunnels spanned by uniform-width sheets. Tunnels form at the junction of three or more cells due to the finite bend radius of cell membrane. The pore size is expected to be a function of ECW, bend radius, and junction geometry as illustrated here. The pore diameter in B is 50% larger than in A even with the same ECW and bend radius, Conceivably tunnels allow diffusion of larger molecules through the ECS than through the uniform width sheets (Thorne and Nicholson, 2006).

surements of the extracellular space. The extracellular volume fraction in the raw reconstruction is diminished compared to in vivo estimates (8% compared to 22%). We explored both linear scaling of the reconstruction and uniform-width extracellular space expansion as compensation for tissue shrinkage during electron microscopy preparation and errors in the reconstruction process itself. Scaling by itself was insufficient to recover lost extracellular space. The reconstruction most likely to reflect in vivo conditions has an extracellular volume fraction of 22%, median extracellular width of 40 nm, and glutamate diffusion constant in the extracellular space of 4.5e-6 cm^2/sec. The extracellular space in the processed reconstruction is best modeled as uniform-width sheets with near uniform extracellular width punctuated by an interconnected network of tunnels with diameter slightly larger than the sheet width.

I.7 Acknowledgements

This chapter is, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Josef Spacek, Thomas M. Bartol, Chandrajait Bajaj, Kristen Harris, and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

II

Generating High-Quality Reconstructions of Neural Tissue for Morphometric Analysis and Modeling

A comprehensive explanation of neuronal function requires that we understand how the diversity of cellular structure seen in the brain shapes neuronal activity. To reach this goal the astounding morphological intricacy of the brain and the heterogeneity of the cellular structures must first be recorded and quantified. The immensity of this task is appreciated when considering that a volume of rat hippocampus the size of a single human red blood cell contains 450 synapses, 500 axons, and 150 dendrites. What is needed is an automated procedure for recording cellular structure in neural tissue and transforming this data into models suitable for morphometric analysis and simulations. This chapter describes a process for the creation of three-dimensional (3D) reconstructions of neuropil from two-dimensional EM images. In Section II.1 each step of the reconstruction process is explained in detail from images to contours all the way though to 3D surface meshes, highlighting at each step potential pitfalls and how to overcome

them. Section II.2 focuses solely on the software algorithm that converts contours into surface meshes, providing a thorough account of the problems encountered when the meshing software was applied for the first time to the task of neuropil reconstruction. Finally in Section II.3 streamlined versions of the reconstruction process are envisioned and modifications to the data pipeline are proposed that will dramatically improve the reconstruction process speed and accuracy.

II.1 What a mesh! Data pipeline deconstructed.

In this section the process of neural tissue reconstruction is followed from image to model. The reconstruction began with serial-section electron micrographs centered on a large apical dendrite in rat CA1 stratum radiatum with all dendritic, axonal, and glial process membranes traced by hand. (Sections II.1.A and II.1.B). The traces (or contours) on each section were splined and sampled to smooth the contours and achieve higher sampling density in parts of a contour with high curvature and lower sampling density in regions with low curvature (Section II.1.C). A surface mesh of each membrane was generated by tiling between the smoothed contours on adjacent sections (Section II.1.D). Several postprocessing steps were necessary to curate the surface meshes to conform to known topology of cell membranes involving a patchwork of scripted, and manual mesh manipulations (Section II.1.E). Finally the extracellular width was expanded by making small deformations of the surface meshes to return the extracellular space volume fraction (EVF) to estimated in vivo amount (Section II.1.F).

II.1.A Data Acquisition: tissue to gray-scale voxels

Three-dimensional reconstructions of neural tissue for both morphometric analysis and modeling of dynamic biochemical processes in neurons at subcellular level require detailed information of cell morphologies at high-resolution. A typical length scale of neuronal ultrastructure is on the order of nanometers. For example,

the extracellular width is tens of nanometers wide (Thorne and Nicholson, 2006), and the cell membrane is eight nanometers thick (Hochmuth et al., 1983). Meanwhile, both the mushroom spine head diameter and distance between mushroom spines along the dendrite are one micron (Bourne and Harris (2008), Petrak et al. (2005)). Investigating structure-function relationships at the subcellular level in neurons requires the acquisition of microns of tissue data with nanometer resolution.

The electron microscope (EM) is a principal tool used for high-resolution capture of cellular morphology for 3D reconstructions. Detailed anatomical structure is revealed in transmission EM images which have a typical in-plane resolution of less than three nanometers and field-of-view of several microns. Spatial resolution orthogonal to the image plane depends on the specific EM technique employed. For example serial section Transmission Electron Microscopy (ssTEM) involves slicing the tissue into 30 to 50 nm thin sections before imaging in the EM. Alternatively Serial Electron Tomography (SET) involves taking a series of images of a one to two micron thick tissue slice from different tilt angles and then relying on a computer to make virtual slices with a typical thickness of 11 nm. Ultimately both techniques generate voxelized representations of micron-sized chunks of neural tissue. The voxel data sets can be huge and may contain a billion voxels, which translates to a gigabyte of data if 8 bit gray-scale values are assigned to each voxel.[1] The suitability of both techniques to neuropil reconstruction has been demonstrated by the complete or partial reconstruction of synapses from different brain regions. For instance, reconstructed synapses include lizard neuromuscular junction, chick ciliary ganglion, rat CA1 apical dendrite, rat CA3 mossy fiber, and rat cerebellar glomerulus. Other synapses of scientific interest to be reconstructed are the calyx of held, node of Ranvier, olfactory bulb glomerulus, and a whole cerebellar glomerulus.

[1]The reconstructions described in Chapter I use a 6 μm x 6 μm x 5 μm region of tissue consisting of 101 images of size 4096 x 4096 pixels where each voxel is 2.3 nm x 2.3 nm x 50 nm, occupying 247 MB of disk space.

On the horizon we see technology improvements associated with the automation of tissue slicing and handling that may dramatically increase the size of these reconstructions, especially for ssTEM. Thin slices are susceptible to stretching and tearing which imposes an upper limit on their size (typically on the order of microns). However, careful handling of the slices could allow an increase in the x and y dimensions of the tissue sections up to 100 microns and more. Additionally, damaged slices interrupt the slice series and hence limit the extent in the z dimension of reconstructions. Better slice handling could also allow thin-section series extending in the z dimension to 100 microns and beyond. Finally, more sophisticated methods for making the slices may decrease the section thickness to 30 nm and lower. Taken together these technological advances could allow larger reconstructions and investigations into new questions in biology, but at a price. The number of voxels in future reconstructions may soon grow too large for manual data handling, thus necessitating the automation of the data pipeline for neuropil reconstruction.

Before EM images of neural tissue can be used for reconstruction, they must be aligned to correct for distortions of the images incurred during handling. Thin slices may undergo translation, rotation, and stretching prior to imaging, thus the resulting images must be processed to undo these undesirable affine transformations. The software program Reconstruct3D (Fiala, 2005) implements the necessary alignment tools for manual registration of EM images and was used for the reconstructions presented in Chapter I.

II.1.B Segmentation and Annotation: gray-scale voxels to annotated contours

In the data pipeline presented here EM images are not used directly for neuropil reconstruction but instead define the placement of contours which

represent the location of cell structures in the plane of the image.[2] Contours are drawn as polylines (set of points connected by straight lines) by tracing the outline of each cellular component on the image, a process called segmentation. Our data was segmented by hand using Reconstruct3D.

Segmentation is a nontrivial task for two reasons. First, distinguishing the many different cellular components in a cell is challenging as all are stained with equal density by the dyes applied to the tissue. For instance, a dark region in an EM image may correspond to the lipid bilayer, mitochondria, presynaptic vesicle, endoplasmic reticulum, etc. Past efforts to automate image segmentation were stymied when software programs would inadvertently connect the cell membrane with a mitochondria, for example. The task becomes both restricting a given contour to a single cellular component and identifying the component. The later step of assigning a name to a contour is called annotation and is discussed in more detail below. The second difficulty encountered during segmentation involves regions of an EM image that are neither light (no stain) nor dark (heavy stain) but are gray (low to medium stain). These regions occur when cellular components lie at a small angle relative to the plane of the image (Figure II.1). Accurately placing contours in gray regions is difficult and error-prone.

The segmentation process generates contours which are not useful unless they are annotated. Annotating contours involves the assignment of one or more names to each contour. The names, or tags, serve to organize the contours and group together collections of contours into logical sets. For example, all of the contours belonging to a single cell would be assigned a tag for the name of the cell such as 'dendrite1'. Currently, the software program used for segmentation and annotation in this project (Reconstruct3D) only supports a single name tag, but one can imagine tagging contours with multiple names based on subcellular morphology as discussed in Section II.3. Reconstruct3D stores all contour data and annotations in XML files which in turn are read and processed by subsequent

[2]Since the gray-scale image represents electron density of a tissue slice of a certain thickness, the 'plane' of the image is an ambiguous term. Here we refer to the midplane of the thin section.

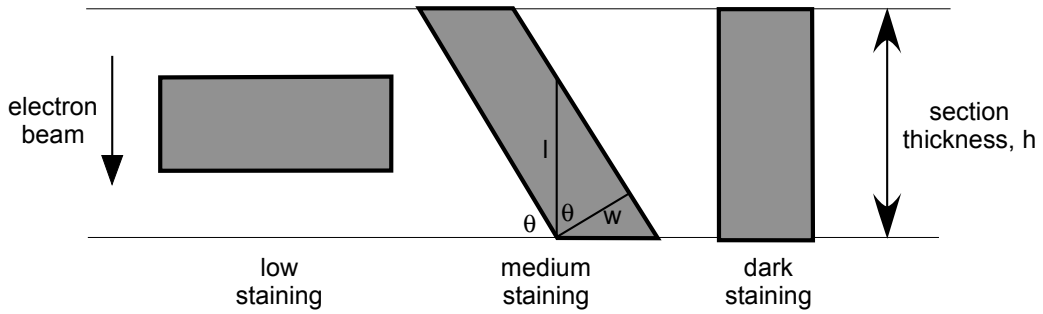

Figure II.1 **Lipid membrane orientation affects contour placement accuracy.** The degree of staining l for lipid membranes is a function of the membrane width w (typically 8 nanometers) and the angle of the membrane relative to the image plane θ: $l = w/cos(\theta)$. For a given section thickness h (typically 50 nanometers) values of θ in the range $\theta^* \leq \theta \leq \pi/2$ result in dark staining where $cos(\theta^*) = w/h$. For all other values of θ $(0 \leq \theta \leq \theta^*)$ the image region will have medium to low staining.

software programs to create surface meshes.

II.1.C Smoothing the contours: splining and sampling annotated contours

The contours in our input data set were generated by manually tracing the cell membranes in each electron micrograph. Contours were laid down along the membrane each time the tracer executed a mouse click using RECONSTRUCT3D. Rather than use these raw contour points directly for generation of surface meshes, we fit nonrational uniform cubic b-splines to the raw points and then generated new contour points using radius of curvature dependent sampling.

By sampling splines finely in regions with high curvature and coarsely where there is low curvature, the fine structure of contours can be represented with fewer points. For example, in Figure II.2 there are 20% fewer interpolated points than raw points, yet the shape of the contour is well preserved. Additionally

spline sampling exhibits a form of noise rejection since the spline will filter out short spatial scale deviations of the contour points away from a smooth path.

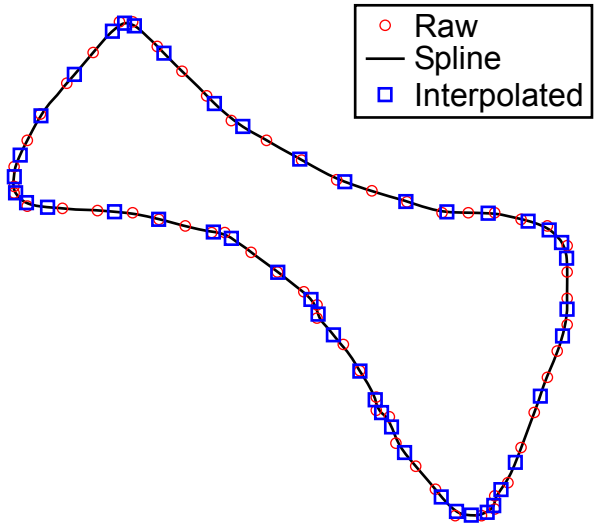

Figure II.2 **Radius of curvature dependent spline sampling is memory efficient and smooths the contour path.** The interpolated points (squares) are concentrated in regions of high curvature, thus preserving the fine detail of the contour with fewer points.

The splines are derived as weighted sums of the raw contour points and do not necessarily interpolate the raw contour points. This could lead to problems, so we compute the deviation of the spline from the raw contours and present the results in a histogram to the user as shown in Figure II.3. The spline deviation from the contour points is controllable by a user-defined maximum allowed deviation. In the example below a threshold of 5 nm was used to reduce the largest deviation from 15 nm to 3.6 nm. We wrote a c++ program (see Section E.3) that reads in XML contours from Reconstruct3D, fits splines to the contours and samples the splines in a curvature-dependent manner, and generates output files used as input to the surface meshing algorithms described in Section II.1.D.

```
Spline deviation statistics:

  Avg deviation = 0.22752 +- 0.375319
  Smallest deviation:               0   |  Largest deviation:      3.6002

  Deviation histogram:
        0 - 0.25716      :      3361    | 2.0573 - 2.3144         :         5
  0.25716 - 0.51431      :       547    | 2.3144 - 2.5716         :         4
  0.51431 - 0.77147      :       469    | 2.5716 - 2.8287         :         2
  0.77147 - 1.0286       :       233    | 2.8287 - 3.0859         :         0
   1.0286 - 1.2858       :        99    | 3.0859 - 3.343          :         0
   1.2858 - 1.5429       :        42    |  3.343 - 3.6002         :         1
   1.5429 - 1.8001       :        21    | 3.6002 - 3.8574         :         0
   1.8001 - 2.0573       :        17    | 3.8574 -                :         0
```

Figure II.3 **Spline deviation statistics report the difference between the splines and the original contour points.** A user-defined threshold controls how well the splines follow the original contours.

II.1.D Surface Generation: annotated contours to annotated raw meshes

The surface mesh reconstructions in this project were generated with an algorithm that uses the points in the contours to construct triangulated surface meshes (Bajaj et al., 1996). This algorithm, implemented as a c program named CONTOUR_TILER, differs from methods which operate on the voxel data directly such as marching cubes (Lorensen and Cline, 1987). The input format for CONTOUR_TILER is its own custom format. Each object is meshed independently, and a small config file ('object_name.config') summarizes the object's contour set. Additionally, the object's contours on each slice are stored in a pts file ('object_nameSlice_number.pts'). CONTOUR_TILER's custom output format ('object_name.poly') is simply a collection of faces with each face described by three vertex coordinates. The algorithm generated surface meshes for the entire reconstruction in one hour running on a 2 GHz processor with 4 GB of RAM.

II.1.E Surface Improvement: annotated raw meshes to annotated processed meshes

The surface meshes generated by CONTOUR_TILER could not be used directly as they exhibited numerous topological qualities inappropriate for neuronal cell membranes.[3] The meshes were modified to fix these problems using a multitude of software packages. First our diagnostic program meshalyzer (Chapter III) revealed that some meshes had nonmanifold edges (Section II.2.F) and others had the wrong genus. Second, we visualized the meshes with a rendering program named DReAMM (`www.mcell.psc.edu/DReAMM`) and noticed sharp thin protrusions (Section II.2.C). These fins were removed by smoothing the mesh via remeshing in a program called FILTERMESH (Hoppe et al., 1993). FILTERMESH output meshes have the additional desirable qualities of highly uniform face aspect ratios and uniform edge length. Unfortunately, these mesh improvements are tarnished by systematic drift - where convex regions of a surface shifted medially and concave regions shift distally - of the mesh surface from its original location. Although later processing steps move the surfaces anyway, these artifacts were problematic as they occasionally created intersections between neighboring meshes. Furthermore we later discovered that FILTERMESH can generate meshes with nonmanifold vertices which also had to be fixed.

We also noticed that invaginations where a cell dimpled the surface of a neighboring cell were meshed as bubbles not pockets (Section II.2.D), so we deleted the bubbles and subtracted the intersecting meshes using a constructive solid geometry program called IRIT (`www.cs.technion.ac.il/~irit`). The decision to use IRIT was born out of necessity and was not without unfavorable consequences. To understand the cost of using IRIT consider two aspects of IRIT

[3]The meshing program was not designed for the application of neuropil reconstruction. Each of the deficiencies identified in the surface meshes is in fact a perfectly reasonable interpretation of geometrical ambiguities present in the input contours. In Section II.3 we describe modifications to CONTOUR_TILER that equate to generating meshes based on alternative valid interpretations of the contour geometry.

behavior. First, IRIT's output format is STL which does not list a unique set of vertices but instead describes each face by explicitly defining the locations of each vertex, repeating vertices for adjacent faces. Second, IRIT's constructive solid geometry operations are allowed to create new faces with edges of arbitrarily small length. Now, imagine what happens when the STL output is analyzed to extract a unique set of vertices as required for later processing steps in the pipeline. The most obvious algorithm for such a task is to simply declare vertices separated by a small distance to be the same vertex. This generally logical and common approach will fail occasionally resulting in output meshes with open or nonmanifold edges.

Once the mesh set was free of errors, we began correcting for fixation artifacts and other accumulated errors that manifested as deviations in the extracellular width using a program called MESHMORPH (Section II.1.F). It was during this process that we needed higher polygon density in regions of the surface mesh with high curvature, so we remeshed the surfaces again using a program called Netgen (Schoberl, 1997) which varies the size of faces based on local surface curvature while maintaining low face aspect ratio.

All of the aforementioned mesh problems were fixed either by writing scripts and programs or by manual process. This particular morphed data set is effectively irreproducible since the number of manual steps reached 200+ and not all changes to the mesh were adequately documented. Section II.3 describes improvements to the data pipeline that should render manual corrections unnecessary so that future reconstruction projects would be faster and repeatable.

At this point the reconstruction of each cell in the tissue block was confirmed by meshalyzer to be valid surface meshes for morphometric analysis or modeling. Each surface mesh was a closed manifold with consistently-oriented, outward pointing face normal vectors with the correct genus and no self-intersections (Chapter III). However, the collection of surface meshes together (representing a reconstruction of neural tissue) did not possess the correct amount of volume outside the cells. The extracellular volume fraction, defined as the ratio of ex-

tracellular volume to total volume, was too low in the reconstruction due to the fact that cells were too close to one another. Neighboring surfaces were much too close (and even intersected in some places) resulting in a median width of the extracellular space that was too low. In other words, while each surface mesh in isolation was acceptable, the relationship between neighboring surfaces needed further refinement. The solution we chose was to write a software program named MESHMORPH which performs local deformations of the surfaces to control the distance between cells, thus increasing the extracellular volume fraction to recover the lost extracellular space.

II.1.F MESHMORPH: recovering the extracellular space

MESHMORPH is a c++ program that accepts collections of surface meshes as input and then deforms the meshes to control the distance between neighboring surfaces. The program compares the actual distance between surfaces to a user-defined target width and either expands or contracts the extracellular space as needed. Local corrections to the extracellular width are iteratively performed until the entire reconstruction has an approximately uniform extracellular width of the user's specification.

MESHMORPH works by modeling a massive system of springs connecting vertices and faces of the surface meshes and calculating the force in the springs to determine how to deform the meshes. The surfaces are moved so as to decrease the force in the springs until the forces in the spring system are balanced. Equivalently, the initially large potential energy of the spring system is dissipated as MESHMORPH relaxes the system into a lower energy state by moving the meshed surfaces.

The system is relaxed by moving each vertex in the mesh into a progressively lower energy position based on the forces in adjacent springs. Each vertex is influenced by linear springs connected to adjacent vertices on the same surface, angular springs connected to adjacent faces, and a linear spring connected to the

neighboring surface (Figure II.4). The force in each spring is determined by the deviation from its resting position and a spring constant. The net force on the vertex varies nonlinearly with vertex position for two reasons. First, trigonometric terms arise from the dependence of the angular spring force on adjacent edge angles. Second, as the vertex moves the closest point C on a neighbor surface to the vertex may change in unpredictable ways. Consequently, MESHMORPH uses a proportionality constant to convert from net force to displacement rather than analytically solve for the position V' that balances the spring forces on the vertex.

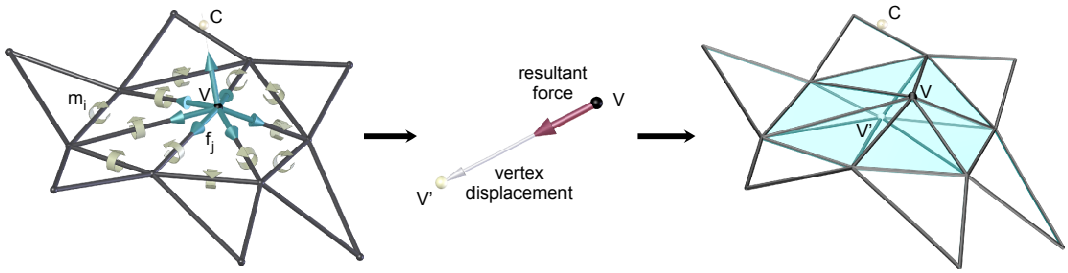

Figure II.4 **Vertex displacement is determined by net force of adjacent springs.** The displacement of vertex V is proportional to the resultant force on the vertex which is calculated as the vector sum of all forces f_j (cyan) from adjacent linear springs (including to closest point C) and reaction forces from moments m_i (gold) from adjacent angular springs.

In designing MESHMORPH we faced two questions. What proportionality constant to use for calculating vertex displacement given the net force on the vertex? In what order should vertices be moved? Given that we could calculate the change in potential energy of the spring system for an arbitrary vertex move, implementing simulated annealing was a possible answer to both questions. Simulated annealing is a theoretical framework that provides a recipe for relaxing a potential energy system that guarantees that the global minimum energy state is reached (Kirkpatrick, 1997). Here simulated annealing would have MESHMORPH move every vertex one-at-a-time in random order in random directions by amounts cho-

sen from a distribution based on the 'temperature' of the model. All vertex moves resulting in lower potential energy would be allowed along with a small fraction of the moves that raise the energy of the spring system. Simulated annealing would begin by 'melting' the spring system at high temperature, effectively randomizing the locations of all vertices by using a distribution of displacements with a large mean value. Then as vertices are moved the temperature would be slowly lowered (translating to lowering the mean of the distribution of vertex displacements) until the surface mesh vertices settle into the configuration that minimizes the potential energy of the spring system.

We elected not to implement simulated annealing in MESHMORPH for two reasons. First, we assumed that relaxing the spring system to a local energy minimum is acceptable. Visual inspection of the reconstruction suggested that small deformations of the surface meshes would recover the lost extracellular space and that randomizing the vertex locations to achieve the global energy minimum was unnecessary. Second, we assessed that a local energy minimum could be reached much faster than the global minimum. For example, ignoring regions of a surface mesh that are at the correct distance away from neighboring surfaces might save time. For these reasons MESHMORPH deviates from simulated annealing in the following ways. Rather than move vertices in random directions, MESHMORPH uses the net force vector to determine the displacement vector of vertices. MESHMORPH maintains a list of vertices ordered according to the magnitude of the calculated vertex displacement. Instead of moving vertices in random order, MESHMORPH tries to move the vertices with the largest displacement first. All vertex moves are kept regardless of the change in potential energy (which is usually negative). To save time vertices with very small calculated displacements are not moved at all. This situation arises by a vertex starting with balanced forces or by being moved to a position where the net force goes to zero.

Because we chose to not use simulated annealing, several novel constraints on the sequence of vertex moves and on the input mesh geometry were required to

reach an acceptable local energy minimum. The fundamental obstacle to reaching the desired energy minimum is oscillatory surface deformations resulting in potential energy oscillations. One source of these oscillations is vertex displacement overshooting the minimum energy position. Similar to simulated annealing, we chose to lower the the proportionality constant between force and displacement according to a schedule, which is called gain scheduling in control theory. With gain scheduling vertex overshoot can still occur, but as the proportionality constant is lowered the magnitude of overshoot will diminish and the vertex position should converge on the minimum energy position. Another source of oscillations in MESHMORPH arises from the choice to always move the vertex with maximum calculated displacement. To prevent the same vertex from moving repeatedly we instituted a refractory period and max number of moves m of a single vertex per n moves. We noticed that in addition to oscillations of individual vertices, multiple vertices in small regions could oscillate together. Our strategy for avoiding oscillations of vertex groups was to geometrically decouple neighboring vertices to the extent possible. For example, MESHMORPH tries to maintain approximately equilateral polygons in the surface meshes during relaxation (Figure III.7). Finally, unrelated to oscillations, it was noticed that MESHMORPH does not handle self-intersecting surface meshes well. We could not in short order devise an automatic scheme for MESHMORPH to separate all self-intersecting faces. Consequently, any residual self-intersections in the MESHMORPH output meshes currently have to be removed manually.

II.2 A CONTOUR_TILER implementation for neuropil reconstruction

We have demonstrated the feasibility of neural tissue reconstruction from EM images to contours yielding surface meshes with the software program CONTOUR_TILER. However, due to the fact that CONTOUR_TILER was not de-

signed for neuropil reconstructions, we have identified several opportunities for adaptation of the algorithm to this application. This section describes modifications to CONTOUR_TILER that would further automate the surface meshing process and improve the accuracy of the meshing by achieving high fidelity reproduction of the original contours and their implicit cellular geometry.

II.2.A New output data format

CONTOUR_TILER's custom output format ('object_name.poly') is simply a collection of faces with each face described by three vertex coordinates. This stl-like format is less than ideal for two reasons. First, it is not the most compact representation of the data. Additionally, identifying corresponding vertices on neighboring faces requires floating-point number comparison which can be difficult when vertices lies close to one another. For example, consider the poly file below representing two triangular faces sharing an edge. The only information given by the poly file is that the object contains two faces with each face containing three vertices and the locations of the six vertices.

```
3
2606.5098 3716.1991 7250
2626.708998 3705.361806 7250
2616.267135 3733.44785458333 7225
3
2626.708998 3705.361806 7250
2629.980111 3704.063639 7250
2616.267135 3733.44785458333 7225
```

In contrast, the same geometry represented in a mesh file contains more information in a slightly more compact form. The four unique vertices in the mesh are listed, and we now see explicitly that the two faces share an edge by virtue of the vertex index references. By outputting the surface meshes in a mesh-like format, CONTOUR_TILER would save the effort of identifying the unique vertices and avoid the potential for error involved.

```
Vertex 1 2606.5098 3716.1991 7250
Vertex 2 2626.708998 3705.361806 7250
Vertex 3 2616.267135 3733.44785458333 7225
Vertex 4 2629.980111 3704.063639 7250
Face 1 1 2 3
Face 2 2 4 3
```

II.2.B Self-intersecting contours: detect and prohibit

For the task of reconstructing neuropil, self-intersecting contours are nonsensical and should be avoided. Ideally, detection of self-intersecting contours would be performed in Reconstruct3D to facilitate rapid contour correction. Additionally, CONTOUR_TILER should also check each contour for self-intersections if it is not already doing so.

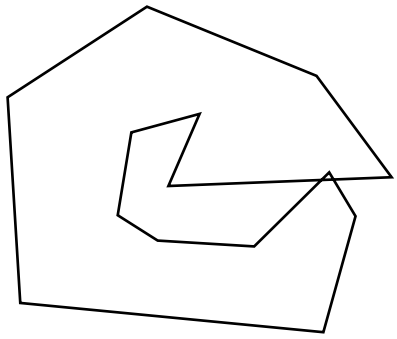

Figure II.5 **Simple example of self_intersecting contour.**

II.2.C High-frequency spatial perturbations

In some circumstances CONTOUR_TILER generates sharp, thin protrusions in the output surface mesh that we ignominiously call "shark fins".[4] These fins are easily detectable in renderings of the output meshes (Figure II.6) as they deviate dramatically from surrounding surface. We suspect that fins violate the

[4]We included a FILTERMESH step which smoothed the fins by completely remeshing, a particularly nonoptimal solution since FILTERMESH introduces systematic local changes in surface location and global deviations in surface area and volume (Section II.1.E).

minimum bend radius of cellular lipid bilayer and are therefore nonphysiological and undesirable. CONTOUR_TILER seems to produce fins under either of two conditions. When the density of points in a local region of a contour varies dramatically from one slice to the next, CONTOUR_TILER resolves the discrepancy by adding fins to consume the extra points. The solution to this problem will likely involve modulation of spline-sampling density across slices to avoid large changes in contour point densities. Fins also arise when contours on adjacent sections are orthogonal in some region of the contour (Figure II.6). The ultimate cause of this problem (like many of the obstacles encountered in the reconstruction process) stems from undersampling the cellular morphology due to constraints on how thin the tissue sections can be. Ideally, a solution will be found that accommodates 50 nm thin sections.

II.2.D Bubbles versus pockets: adding a new capping option

The CA1 morphology includes both bubble motifs, .e.g vesicles, mitochondria, and pocket motifs, e.g. indentation of a dendritic spicule into an axon. Interestingly, the contours of both of these motifs often have the same form: concentric rings of contour points (Figure II.7A). However, from concentric contours of a single object CONTOUR_TILER only reproduces bubble motifs. This is in fact a perfectly reasonable outcome considering the ambiguity of the contour geometry given as input to the tiler. Consider the case described in Figure II.7, where contours of an object on three sequential sections can clearly be seen to arise from two distinctly different geometries. If we interpret contour C2b as a bubble, then a cross-section of the reconstructed mesh along axis w-w should look like Figure II.7B. Alternatively, if contour C2b is treated as a pocket then a topologically different reconstruction is appropriate and the cross-section will look like Figure II.7C. The reconstruction that is correct is necessarily context dependent since the input contours themselves are samples of two different, but valid structures.

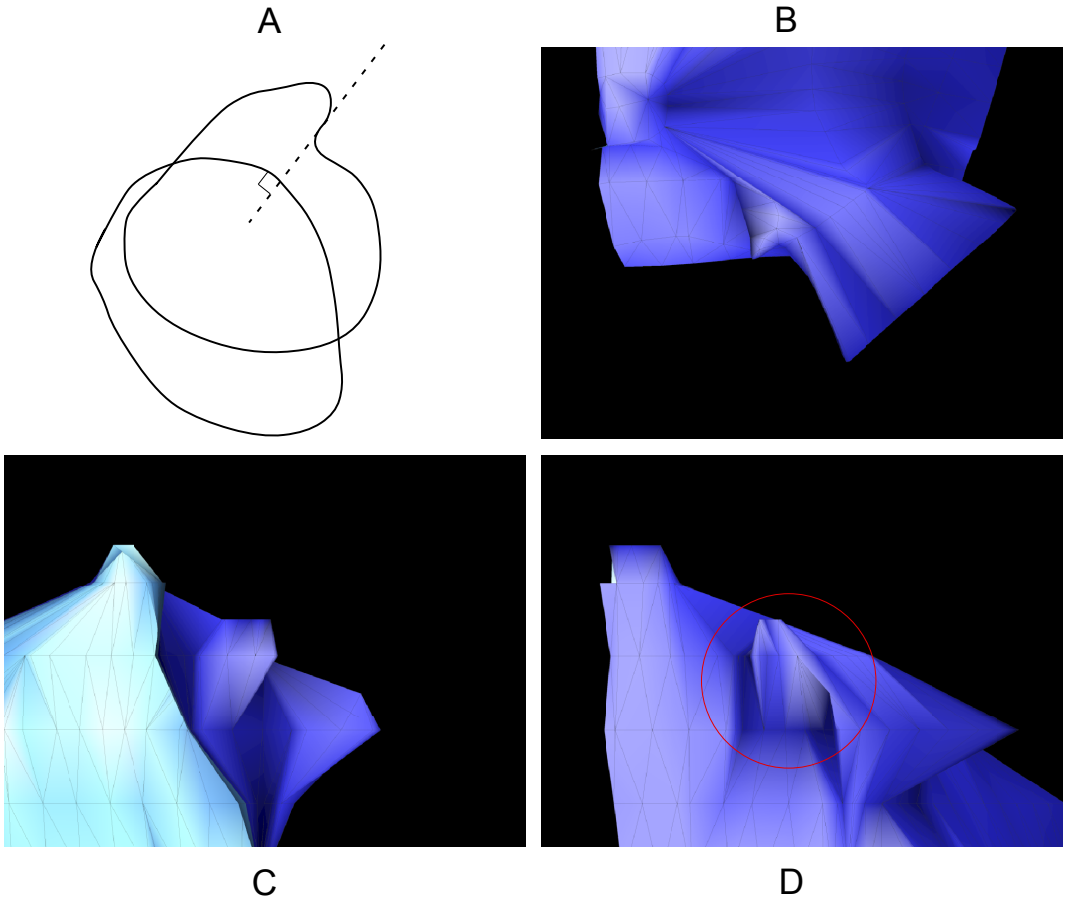

Figure II.6 **Sharp, thin protrusions punctuate the mesh surface.** (B-D) Top, left, and front views respectively of surface mesh with fins. These arise when in a local region of a contour either the density of points varies dramatically from one slice to the next or (A) contours are orthogonal.

Since both reconstructed geometries are useful, the most attractive resolution is to architect both construction paths into CONTOUR_TILER and allow user-directed choice. Consequently, we propose to add a pocket meshing algorithm to CONTOUR_TILER and define a new command line option to CONTOUR_TILER that specifies either all pocket caps or all bubble caps.

The fundamental difference between the two meshing outcomes can be framed as a mesh capping choice. Given concentric contours either cap the in-

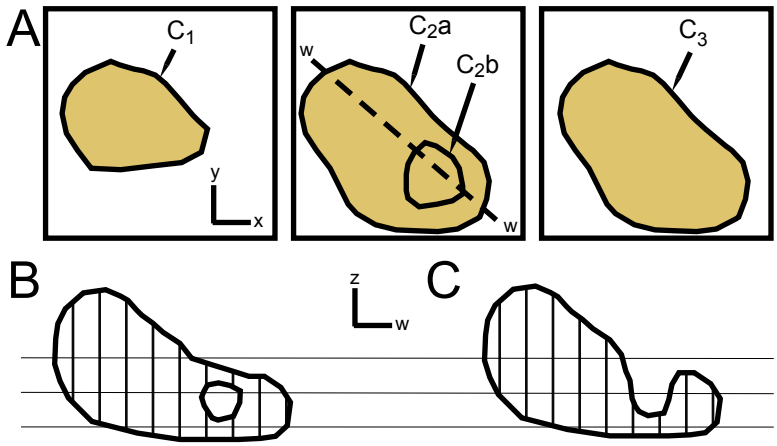

Figure II.7 **Two different 3D geometries have identical contours.** Not surprisingly examples of both geometries are well represented inside the CA1 tissue volume. Notably, the bubble in B can represent a vesicle inside an axon, while C might describe the indentation of a spicule into an axon forming a pocket. Currently, CONTOUR_TILER meshes cellular indentations as intracellular bubbles rather than as pockets in the cell membrane.

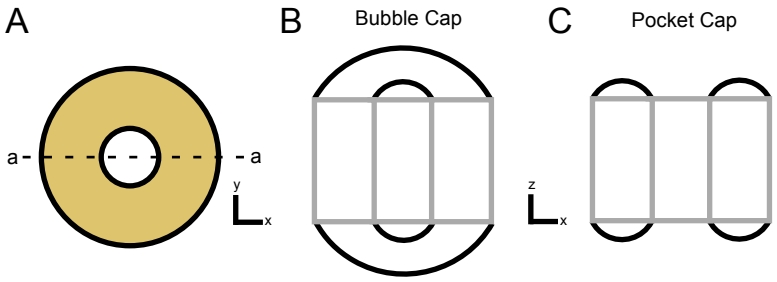

Figure II.8 **Two capping paradigms.**

ner and outer contours separately, thus creating a bubble, or cap only the space between the contours, creating a pocket. With no loss of generality, consider the two capping options of a toroid as illustrated in Figure II.8. Figure II.8B and Figure II.8C represent slices through section a-a under the two different capping paradigms.

Our data set contains approximately 100 C2b-like contours all of which generated bubbles in CONTOUR_TILER. We manually identified and deleted the meshed bubbles then created pockets by subtracting the indenting mesh from the indented mesh using the constructive solid geometry software tool IRIT (Section II.1.E). This plan required a postprocessing MESHMORPH step generating meshes that only approximate the original contours.

A versatile implementation of this new capping functionality would be via contour annotation as addressed in Section II.2.E. Additionally, a command-line switch for CONTOUR_TILER which specified global capping directive would be useful and sufficient for these reconstructions. For instance, the default state for CONTOUR_TILER would be to exclusively create pockets and no bubbles with an optional command-line flag to create bubble caps in current manner and no pockets. In fact, it is the command line switch that would be easiest to implement and should be pursued first. Note that a global stipulation that CONTOUR_TILER exclusively create pocket caps should only apply to concentric contours, not to single contours. Then, should the need arise for intracellular bubbles such as for mitochondria, endoplasmic reticulum, vesicles, etc. we can simply assign the c2b-like contours a unique name and mesh them as independent objects. In this way we can preserve the current functionality and gain the ability to create mesh pockets.

II.2.E Contour annotation: achieving a flexible merge policy

The EM data used for our reconstructions has sufficient in-plane resolution (2.3 nm pixels) to accurately capture the intricate morphology of neural tissue. However, the 50 nm section thickness is larger than the smallest length scales found in the neuropil and undersamples the cellular geometry. Consequently, cellular topology is not well preserved in surface mesh reconstructions. We posit that topological analysis is required to recognize and correct reconstruction errors due to relatively course sectioning. Currently, this kind of high-level knowledge

involves human intervention to direct the reconstruction process. We propose to add and support new contour metadata as a platform for incorporating human guidance in creating surface meshes from contours.

A survey of the instances in which the surface mesh reconstructions differed topologically from established neuronal shape revealed the underlying cause to be the meshing between contours on adjacent sections that ought not to have been merged. CONTOUR_TILER obeys a strict policy for determining when contours will be connected together with polygons. Two contours from the same object will be connected if and only if (1) the contours lie on adjacent sections and (2) the contours overlap. It comes as no surprise that geometries arise in the complex, spaghetti-like neuropil of CA1 for which such a strict policy results in an incorrect reconstruction. Figure II.9 attempts to distill the myriad different configurations of processes seen in the CA1 reconstruction into a concise representative geometry. Let the illustrated geometry represents the correct morphology of a few cellular processes. The current CONTOUR_TILER contour merge algorithm will merge contours 2a, 3a, and 4a yielding an incorrect surface mesh.

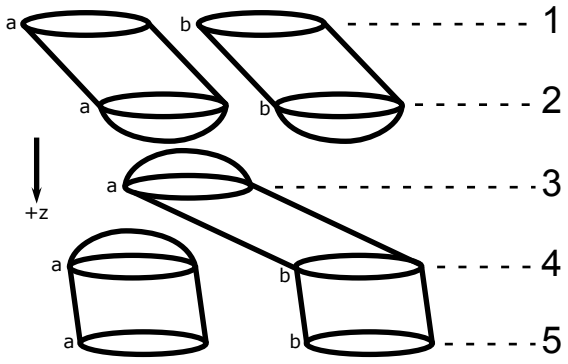

Figure II.9 **Example cases of contour annotation.** CONTOUR_TILER has a strict merge policy that would erroneously join contours 2a, 3a, and 4a. We propose the adoption of metadata to create a flexible merge policy.

To correctly handle geometries such as in Figure II.9 we propose to add

and support new contour metadata to achieve contour-specific merging. In Table II.1 we demonstrate that the addition of two new directives ('merge' and 'nomerge') to the contour meta-data specification are sufficient for accurate reconstruction of Figure II.9. User-defined contour annotation will selectively deviate CONTOUR_TILER merging and capping from a global user-modifiable default behavior. Table II.1 assumes that by default overlapping contours will be merged (auto merge) and contours with no overlapping contours will be capped (auto cap). The metadata XML syntax could be defined as "<Contour key='ID' [merge='LIST'] [nomerge='LIST'] />" where ID is a unique string that identifies the contour and LIST is a comma-separated list of contour keys. A contour with a merge list will be merged with all contours in the list. A contour with a nomerge list will NOT be merged with any contours in the list. The syntax will require conflict resolution to handle incompatible merge directives.

Table II.1 Summary analysis of Figure II.9.

Section	Contour	Overlaps	Directives
1	a	2a	auto merge 2a
1	b	2a, 2b	merge 2b, nomerge 2a
2	a	3a	nomerge 3a
2	b	0	auto cap
3	a	4a	nomerge 4a, merge 4b
4	a	5a	auto merge 5a
4	b	5b	auto merge 5b

A special case of contour geometries that is especially difficult to handle arises frequently during the reconstruction of astrocyte membranes. Figure II.10A illustrates a c-shaped motif common in astrocyte contours. The meshing problem arises when the 'mouth' of the contours on sequential sections drifts substantially and the left side of the mouth in contour 2 (blue) is located to the right of the right side of the mouth of contour 1 (red). As a result, the left arm of contour 2 will be merged with both the left and right arms of contour 1, a most undesirable outcome. Unfortunately, the simple 'merge' and 'nomerge' directives proposed

above will not solve this outstanding problem.

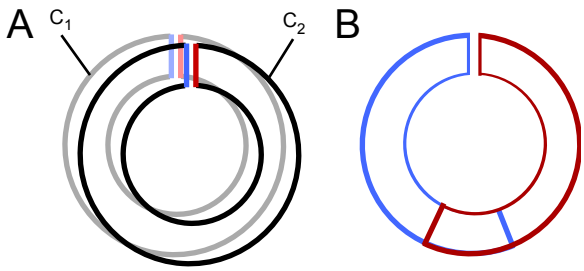

Figure II.10 **Typical astrocyte contour geometry creates undesirable contour merging.** (A) The left side of the mouth in contour 2 (blue) is located to the right of the right side of the mouth of contour 1 (red). As a result, the left arm of contour 2 will be merged with both the left and right arms of contour 1, a most undesirable outcome. (B) The solution we employed was to divide each contour into two overlapping subregions. The left hand side and right hand sides are meshed as independent objects and afterwards merged through a union operation in the constructive solid geometry tool IRIT. This special cases has arisen five times in the CA1 reconstruction.

It was frequently the case in the CA mesh set that adjacent processes of the same object had been merged by CONTOUR_TILER as previously described. For the reconstructions described here we handled geometries such as Figure II.9 by renaming the contours 3a, 4b, and 5b with a unique name, meshing this new object separately from the rest of the parent object and then stitching the processes back together. This is a nontrivial, manual task but has proven effective in most cases. In many other cases we have taken the liberty to actually shift the contours slightly to avoid or create overlap as need be. The complicated geometry of Figure II.10 inspired the strategy depicted in Figure II.10B.

II.2.F manifold edges: make it an option

When contours on adjacent sections overlap by a small amount, CONTOUR_TILER generates meshes that represent nonmanifold surfaces. The nature of the nonmanifoldness is abstracted in Figure II.11 where an edge e-f is shared by more than two polygons (e.g. four in the example, two red and two blue). In our current data set, we have observed ten of these examples of nonmanifold geometry in the CONTOUR_TILER output.

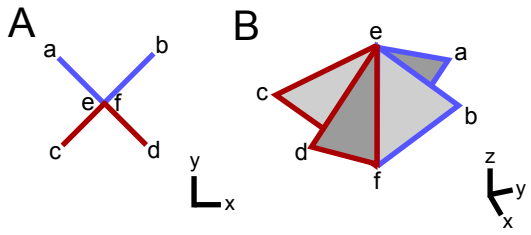

Figure II.11 **Schematic of Nonmanifold Surface Mesh.** Polygons e-f-c, e-d-f, e-a-f, and e-b-f all share edge e-f.

This geometry arises when the degree of overlap between contours on sequential sections is small. For example, in Figure II.12 two contours of the same object on sequential sections are shown overlapping slightly in the region defined by the blue circle. Note significant and nonoffensive contour overlap in the lower-left corner of the figure. The resulting mesh from CONTOUR_TILER using the contours in Figure II.12 is nonmanifold as seen in Figure II.13. The left and right branches of the object merge at the edge defined by the red and green vertices. This edge is shared by four polygons and renders the mesh nonmanifold. Nonmanifold edges were manually removed from the CONTOUR_TILER output meshes when building the CA1 reconstructions. The fix always involves duplicating at least one vertex, moving the vertices and adding at least two new polygons. MESHALYZER can detect and report the location of nonmanifold edges.

The current behavior of CONTOUR_TILER is in fact correct when one recognizes that the underlying problem for both nonmanifold edges and the obsta-

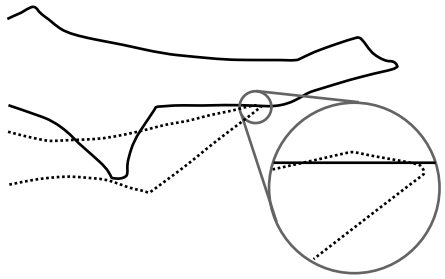

Figure II.12 **Contours with small overlaps can generate nonmanifold meshes.**

cles described in Section II.2.E is undersampling along the sectioning axis. If the section thickness was smaller, the abrupt transitions between contour geometry as seen in Figure II.12, Figure II.9 and Figure II.10 would be alleviated by intervening contours. However, in practice the section thickness is governed by technical limitations, so we are compelled to bias the tiling process with external, expert knowledge.

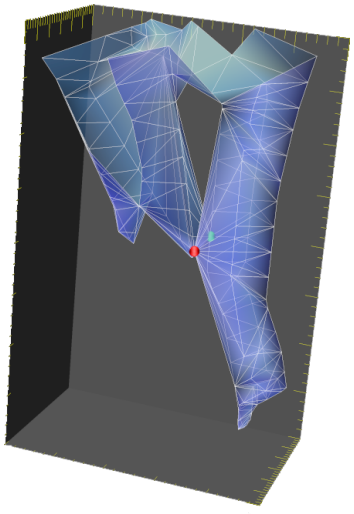

Figure II.13 **Example of a nonmanifold mesh from CONTOUR_TILER.** The edge defined by the read and green vertices is shared by four polygons.

We propose to define a new command line option to CONTOUR_TILER

that prohibits nonmanifold edges. The new default state would result in manifold edges exclusively, while a command-line option will continue creating nonmanifold edges in current manner by following a policy whereby small percentages of overlap imply correspondence between the overlapping regions of contours. This fix by itself will prevent nonmanifold edges but is unlikely to generate output meshes that optimally conform to expectation. Minimal contour overlap due to course sampling will still arise as cases of ambiguous correspondence similar to the special case in Figure II.10.

II.2.G Scale-dependent mesh behavior

Capping in CONTOUR_TILER behaves in a scale-dependent manner. That is to say the scale of the input contours affects the output mesh cap. For example, we serendipitously discovered that converting the contour points into nanometers results in better behaved caps than when the contour points are in microns. Furthermore, in nanometer units the prevalence of very thin protrusions from the caps decreases. The attractive solution would be to parameterize this effect and expose it to the user for capping geometry control.

II.3 Future Work

The reconstructions described above were generated from a patchwork of software tools woven together more from necessity than by design. The odd assortment of large programs, small scripts, and manual operations entered on the command line reflects a poor degree of tool integration. Streamlining the data pipeline would accelerate the process of reconstructing tissue through better automation and also improve the accuracy and repeatability of the reconstruction. We propose a global framework for streamlining the data pipeline. This global framework is provided by expanded metadata and a new file format that supports extensible metadata. Built on top of this framework will be software programs

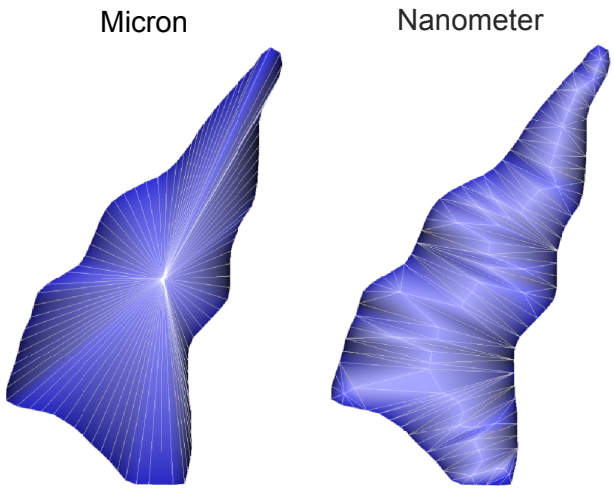

Figure II.14 **Scale-dependent mesh geometry.** Both meshes were built with the same collection of contour points as evidenced by identical vertices along the mesh borders in the two objects. However, the input contour points for the right hand mesh were scaled by 1000 before meshing. We prefer the mesh geometry generated at the nanometer scale.

that can read input metadata, use it for improved reconstructions, and preserve the metadata in output files.

II.3.A Metadata: Part 2

Currently, contours can be annotated with only a limited set of metadata tags. This repertoire must be expanded to improve meshing as described in the CONTOUR_TILER improvements (Sections II.2.D and II.2.E). However, a second use for contour metadata could be the specification of subcellular regions for incorporation of molecular machinery for biochemical modeling, e.g. the creation of regions for MCell model building. Thus, one might imagine two ways in which the creation of regions for MCell model building occurs - (1) by the assignment of metadata to surface mesh elements (such as vertices, faces, etc.) directly and (2) by the assignment of metadata to contours with the provision that the metadata is

transferred to the output surface meshes constructed from the contours. Extending the use of metadata for both contours and meshes and preserving this data along the software pipeline would support a richer variety of automatable tasks. The following groups of metadata were imagined to address the need for manual intervention using current methods.

- contour subcellular annotation - e.g. contour is located at the spine head, spine neck, dendrite shaft, axon hillock, soma, etc. and after surface generation a corresponding mesh region should be created.

- mesh subcellular annotation - e.g. this mesh element or collection of elements constitutes the spine head, spine neck, dendrite shaft, axon hillock, soma, spicule, postsynaptic density, active zone, etc. This mesh element metadata may have been inherited from contour metadata or annotated by the user after surface generation.

II.3.B File Formats

The collection of disparate software programs and scripts utilized a variety of languages (C, C++, perl, php, python and bash) and data file formats (DX, mesh, VTK, STL, and STL-like). The software programming language is not critical per se as long as the tool gets the job done. Besides, the process of tool integration will naturally lead to a convergence on one or a few languages. Of greater importance is data file formats. We need to pick a file format that accommodates tool integration, metadata, and is extensible to large data sets possibly incorporating out-of-core methods. Candidates so far are (1) mesh+, i.e. extended mesh format and (2) mesh and auxiliary metadata file. Mesh formats have the advantage over STL-like formats of compactnesses and clarity owing to the explicit definition of vertex and face relationships. While a binary format would offer increased data compression and easier parsing by software programs, the ASCII format of mesh is preferred for human-interpretability.

II.4 Acknowledgements

This chapter is, in part, in preparation for submission for publication. The dissertation author was co-lead investigator and co-lead author along with Andrew Gillette. The co-authors, Thomas M. Bartol, Chandrajait Bajaj, Kristen Harris, Daniel Johnston, Jose Rivera, Clifton Rumsey, and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

III

MESHALYZER: a mesh quality analyzer

III.1 Introduction

Our computer simulations of neural activity in rat hippocampus utilized electron micrographs with nanometer resolution from which we built three-dimensional replicas of every cell in a micron-size chunk of tissue. Creating the reconstruction was not trivial; simply having a collection of polygons (or mesh) that represents the surface of a cell membrane does not guarantee that the representation is valid. The mesh must possess certain qualities to be consistent with known cellular topology, compatible with the software tools used for reconstruction, and to maximize the accuracy of the final surface reconstruction.

By trial-and-error we identified a set of mesh qualities that are necessary and sufficient to insure the highest degree of compatibility with mesh analysis/manipulation software. We divide these qualities into four attributes and several characteristics. The attributes are addressed by the following questions. (1) Is the mesh closed? Are there any holes in the mesh lattice or is it water tight? (2) Is the mesh a two-dimensional manifold? Is the mesh neighborhood around every vertex homeomorphic to a plane? (3) If the mesh is a two-dimensional closed

manifold, then do the mesh polygons have consistent orientation? Do all polygons have normal vectors oriented the same direction (inward or outward) or are they of mixed orientation? (4) If the mesh has consistent orientation, then do all polygons have normals pointing outward? A mesh must have these four attributes to be compatible with the reconstruction software. The reconstruction process is a pipeline where the output mesh of one software program is the input mesh for the next program in the pipeline. Ideally, each software program would be written to insure that its output mesh has the four important attributes. However, at this time that is not the case, so meshes must be checked and corrected if necessary. In most cases, the correction of meshes that violate these attributes is a manual process involving the removal of nonmanifold mesh elements, the reorientation of flipped meshes, and the insertion of polygons into mesh holes.

In addition to the four attributes, several more mesh qualities are indicative of a good mesh. These mesh characteristics also impact the success of the reconstruction process as they influence the algorithms used by the software programs to manipulate the meshes. The important mesh characteristics that we identified concern the shape of the polygons that make up the surface mesh and the angle between adjacent polygons that share an edge. All of our meshes strictly use triangular polygons and the highest quality triangular polygon is equilateral. By characterizing the degree to which the surface mesh polygons are equilateral one assesses the quality of the mesh and its propensity for favorable outcomes with the reconstruction software.

To facilitate the reconstruction process we distilled these criteria for measuring the quality of surface meshes into a software program called MESHALYZER. The purpose of the program is to quickly analyze and report on both general characteristics and four topological attributes of meshes. Running MESHALYZER on a mesh file will provide enough information in a single report to determine if the mesh will fail as input to any of the manipulation processes described in this project. This document is a user guide for MESHALYZER, provides sample out-

put from the program, and illustrates both high-quality meshes that possess all of the requisite attributes and characteristics we desire and poor meshes that our experience has shown will only cause trouble during the reconstruction process.

III.2 Assumptions

Before describing the usage of MESHALYZER to analyze mesh objects, it is important to mention the assumptions made by this algorithm. First, the mesh file is assumed to be in the Hughes Hoppe mesh file format which first lists the location of all vertices followed by a list of faces that reference the vertices. Vertices are specified by a vertex index i and three coordinate positions; x, y, z: $[Vertex \quad i \quad x \quad y \quad z]$. Faces are specified by a face index j and three vertex indices; $v1$, $v2$, $v3$: $[Face \quad j \quad v1 \quad v2 \quad v3]$. All indices are positive integers larger than zero. For example a cube of side length one in the +x,+y,+z octant with one corner at the origin could be as follows.[1]

```
Vertex 1  0 1 0
Vertex 2  1 1 0
Vertex 3  1 0 0
Vertex 4  0 0 0
Vertex 5  0 1 1
Vertex 6  1 1 1
Vertex 7  1 0 1
Vertex 8  0 0 1
Face 1  1 2 4
Face 2  4 2 3
Face 3  3 2 6
Face 4  3 6 7
Face 5  6 2 1
Face 6  6 1 5
Face 7  5 1 4
Face 8  5 4 8
Face 9  8 7 6
Face 10  8 6 5
Face 11  4 7 8
Face 12  4 3 7
```

As is evident from the example, the polygons (faces) that make up the mesh are triangular and contain exactly three vertices. The third assumption we

[1]Note the sequence of neither vertex positions nor faces is unique since both vertices and faces can be shuffled while preserving cube geometry.

make specifies how to compute the normal vector of a face as shown in Figure III.1. A face referencing vertex indices {v1,v2,v3} has has a normal vector, $\overrightarrow{n} = (\overrightarrow{v_2} - \overrightarrow{v_1}) \times (\overrightarrow{v_3} - \overrightarrow{v_1})$ where $\overrightarrow{v_i}$ is the vector location of vertex index vi. One can imagine two faces that each reference the same three vertices but in two different sequences such that the face normals point in opposite directions. This is a common source of confusion when working with mesh manipulation software. For example, some software packages, such as DReAMM (`www.mcell.psc.edu/DReAMM/`), VTK (`www.vtk.org`), FILTERMESH (Hoppe et al., 1993), and MESHALYZER, assume that meshes are constructed so that the face normals all point towards the outside of the object. In contrast, other programs, such as IRIT (`www.cs.technion.ac.il/~irit`) and CONTOUR_TILER (Bajaj et al., 1996), expect face normals to point towards the inside of the object. Fortunately, once one is aware of these differences it is a trivial matter to flip the direction of face normals in a mesh file simply by reversing the sequence of the vertex indices for each face.

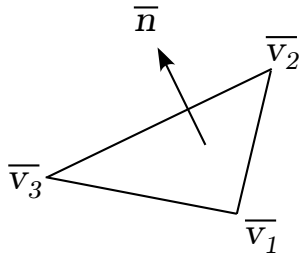

Figure III.1 **Arbitrary convention for face normal vector in MESHALYZER.** A face referencing vertex indices {v1,v2,v3} has a normal vector, $\overrightarrow{n} = (\overrightarrow{v_2} - \overrightarrow{v_1}) \times (\overrightarrow{v_3} - \overrightarrow{v_1})$

III.3 Usage Example

Here we demonstrate the process of analyzing a surface mesh. Suppose we have the mesh file for the object rendered in Figure III.2, ('d014.mesh'). This mesh was specifically chosen because it has no flaws, but suppose for the sake of

illustration that the mesh quality is unknown and wish to know more about it.

Figure III.2 **Example surface mesh of a reconstructed dendrite (d014.mesh).** The object in this example is small and can be meshalyzed in two seconds. The largest single mesh object used in the reconstructions (the glial cell) contains a couple hundred thousand vertices and takes sixty seconds to meshalyze. Every mesh object (599 meshes) in the reconstruction can be meshalyzed in less than ten minutes.

Issuing a MESHALYZER command ('meshalyzer d014.mesh') yields the output shown below. The output of MESHALYZER can be viewed as three distinct parts: file integrity, mesh attributes, and mesh characteristics. Let us examine each of the parts one at a time beginning with file integrity.

```
/* ********************** OBJECT ********************** */
name: d014.mesh

Checking vertex integrity for object [d014.mesh]..................complete.
```

```
Checking face integrity for object [d014.mesh].....................complete.
Create edges for object [d014.mesh]................................complete.
Finding vertex adjacencies for object [d014.mesh]..................complete.
Checking if object [d014.mesh] is closed...........................complete.
Checking if object [d014.mesh] is manifold.........................complete.
Checking if object [d014.mesh] faces are consistently oriented....complete.
Checking if object [d014.mesh] faces are oriented outward.........complete.
Bound object [d014.mesh]...........................................complete.
Check if vertices are distinguishable for object [d014.mesh]......complete.
Analyze vertex adjacent faces for object [d014.mesh]..............complete.
Identify separate components of object [d014.mesh].................complete.
Compute face area and aspect ratio for object [d014.mesh].........complete.
Find intersecting faces for object [d014.mesh].....................complete.
Identify boundaries for object [d014.mesh].........................complete.
Analyze edge lengths for object [d014.mesh].......................complete.
Analyze edge angles for object [d014.mesh].........................complete.
Compute volume of object [d014.mesh]..............................complete.
Compute genus of object [d014.mesh]...............................complete.

MESH FILE INTEGRITY

    # orphan vertices: none
    # missing vertices: none
    # degenerate faces: none
    # duplicate vertex indices: none
    # duplicate face indices: none
    contiguous vertex indexing from 1: yes
    contiguous face indexing from 1: yes
```

The first set of tests evaluate the integrity of the file format. With no consideration to the locations of the vertices, MESHALYZER first verifies that the file conforms to proper mesh syntax. If the mesh file is missing information or has ambiguities then the format is fatally flawed and no more testing can be done. For example do any faces refer to missing vertex indices that are not defined in the file? Do any degenerate faces refer to the same vertex index more than once? Was a duplicate vertex index used in the list of vertices? Was a duplicate face index used in the list of faces? Are there any orphan vertices which are not referenced by any face?[2] If any of these cases are true then MESHALYZER ends execution after the integrity check without evaluating mesh attributes or characteristics. In our example mesh were found no orphan vertices, missing vertices, degenerate faces, duplicate vertex indices, nor duplicate face indices.

[2]Arguably, orphan vertices could be treated as nonfatal violations of mesh file format since they can simply be ignored.

Other integrity checks are not fatal such as vertex and face index contiguousness. Here MESHALYZER is checking that the index numbers of both faces and vertices start at one and increase monotonically without skipping any numbers. Noncontiguous indexing is reported when found but does not impede further analysis of the mesh file. The indexing of the vertices and faces in our example mesh are contiguous. Mesh files that fail the integrity test are of no use to the reconstruction process and must be fixed. All meshes used in the reconstruction process pass the integrity test. If mesh file integrity is found to be intact, then MESHALYZER continues with analysis of the mesh's attributes, as shown below.

```
MESH ATTRIBUTES

    mesh is closed: yes
    mesh is manifold: yes
    mesh has consistently oriented face normals: yes
    mesh has outward oriented face normals: yes
```

In our nomenclature a mesh has four attributes as discussed before. (1) Is the mesh closed (e.g. a soap bubble) or open (e.g. a burst water balloon)? To test for whether a mesh is closed MESHALYZER examines each edge in the model. If each edge is shared by two or more faces, then the object is closed. If any edge has only a single adjacent face, then the object is open. (2) Is the object manifold or nonmanifold? If for each vertex all adjacent faces of the vertex are reachable by edge-hops starting from any of the adjacent faces then the object is manifold; otherwise it is nonmanifold. This means that for an arbitrary vertex in a manifold object one can start from any adjacent face of the vertex and by identifying an adjacent face of the previous face circumnavigate the vertex touching all adjacent faces of the vertex in the process. Manifold meshes have the property that all edges have no more than two adjacent faces. A mesh can be open and manifold if is has some edges with only one adjacent face (Figure III.3B). Note that a mesh with any edge having more that two adjacent faces is nonmanifold (Figure III.3D), but a mesh with all edges having exactly two (or one) adjacent faces is NOT necessarily

manifold (Figure III.3C).

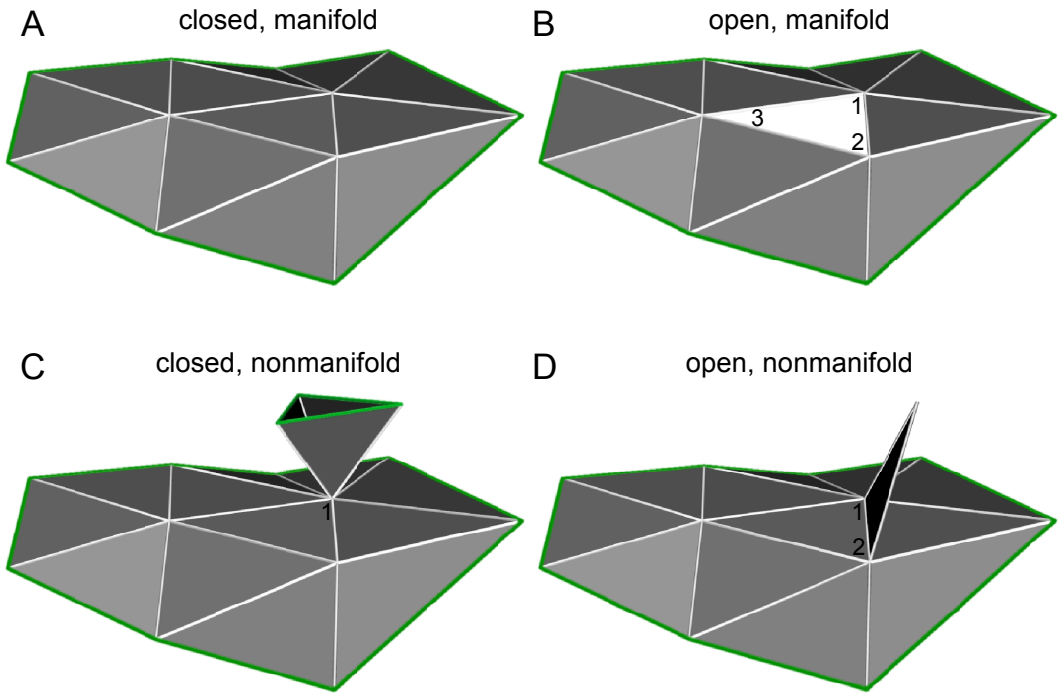

Figure III.3 **Examples of surface mesh attributes 'closed', 'open', 'manifold', and 'nonmanifold'.** Surface meshes were cut (green edges) to facilitate rendering. (A) All edges have exactly two adjacent faces, and all vertices are manifold. (B) Edges 1-3, 3-2, and 2-1 have a single adjacent face, hence the mesh is open. (C) Vertex 1 is nonmanifold. (D) Nonmanifold edge 2-1 has three adjacent faces.

(3) Do the faces of the mesh have consistent orientation or not? MESHALYZER tests face orientation using face edges and the direction that each edge is traversed. For example suppose we have Face 1 1 2 3 and Face 2 4 2 1 that reference vertices 1, 2, 3, and 4 (Figure III.4). We say that Face 1 traverses edge 1-2 in the 1-2 direction, while Face 2 traverses edge 1-2 in the the 2-1 direction, the opposite direction of Face 1. Assuming a surface mesh is manifold (since the orientation of a nonmanifold mesh is undefined), if no edge is crossed twice in the same direction, then all face normal vectors are consistently oriented. In other

words, all the face normal vectors point inwards or all point outwards. Note that the consistency of face orientation can be determined for a non-closed mesh as long as it is manifold.

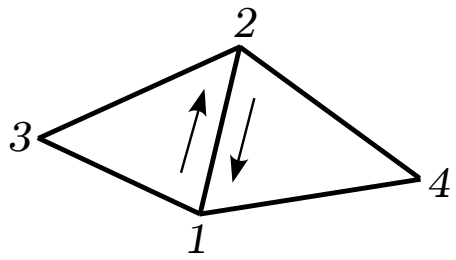

Figure III.4 **The convention for computing face normals also implies a method for determining edge traversal.** Given Face 1 1 2 3 and Face 2 4 2 1 that reference vertices 1, 2, 3, and 4, we say that Face 1 traverses edge 1-2 in the 1-2 direction, while Face 2 traverses edge 1-2 in the the 2-1 direction, the opposite direction of Face 1.

(4) Do all of the face normal vectors point outwards or inwards (Figure III.5)? To evaluate this attribute the mesh must have consistently oriented face normals (which in turn requires that the mesh be manifold). Furthermore the mesh must be closed to define the inside and outside of the mesh. Assuming that the mesh is closed, manifold, and consistently-oriented, MESHALYZER traces the normal vector of an arbitrarily chosen face from the face centroid to infinity and counts the number of times the the ray intersects other faces of the mesh object. If the intersection count is even (including zero), then the face normals point outwards. If the intersection count is odd, then the face normals point inwards.

Our example mesh is closed, manifold with consistently-oriented face normals pointing outward. In fact, all of the meshes used in the reconstructions have these same attributes and for good reason. The meshes represent the outer surface of neuronal and glial lipid membranes which are closed, manifold objects. The orientation of the face normals is important since many of the software tools use the orientation information to manipulate the meshes. For example, IRIT can

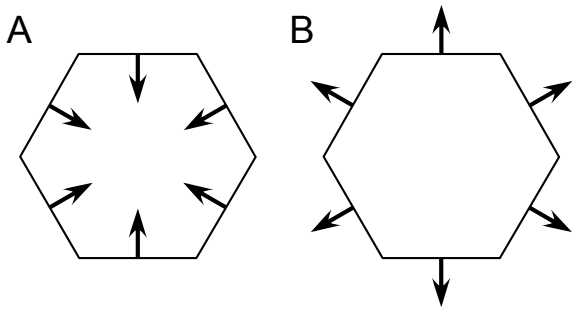

Figure III.5 **Orientation of closed surface meshes with consistently-oriented face normals can be either (A) inwards or (B) outwards and is determined by ray tracing.**

take as input two objects A and B and return the part of object A that is outside of object B, where 'outside' is determined by the direction of the face normals of object B. In another example the software program MESHMORPH calculates the distance between two mesh objects by finding the closest point on object B that lies in the direction of the face normals of object A, requiring that the face normals point outwards. Finally, biochemical simulations involve populating the surfaces with membrane-bound receptors with the extracellular and intracellular receptor domains (as dictated by the face normals) correctly oriented.

```
MESH CHARACTERISTICS

    # vertices: 10042
    # faces: 20080
    # edges: 30120
    # components: 1
    # boundaries: none
    # indistinguishable vertices: none
    object area: [(data units)^2]
    object area: 1.76761e+07
    object volume: [(data units)^3]
    object volume: 1.5886e+09
    object genus: 0
    bounding box: [data units]
    bounding box: [xmin,ymin,zmin][xmax,ymax,zmax]
    bounding box: [5238.44,3945.85,4180.09][7953.6,7590.53,8067.78]
    # edges with indistinguishable vertices: none
    # intersecting faces: none

    Vertex adjacent face statistics [faces]:
       min      3
```

```
   max      10
   median   6
   mean     5.99881
   variance 0.691364

Vertex adjacent face histogram [faces]:
      0 - 0           :         0  |         8 - 8           :       361
      1 - 1           :         0  |         9 - 9           :        52
      2 - 2           :         0  |        10 - 10          :         9
      3 - 3           :         2  |        11 - 11          :         0
      4 - 4           :       186  |        12 - 12          :         0
      5 - 5           :      2336  |        13 - 13          :         0
      6 - 6           :      5308  |        14 - 14          :         0
      7 - 7           :      1788  |        15 - 15          :         0

Face area statistics [(data units)^2]:
   min      0.385088
   max      8755.17
   median   625.012
   mean     880.284
   variance 667912

Face area histogram [(data units)^2]:
      0 - 0           :         0  |      1342 - 1559        :       795
      0 - 33.93       :       395  |      1559 - 1777        :       586
  33.93 - 251.9       :      2342  |      1777 - 1995        :       439
  251.9 - 469.8       :      4287  |      1995 - 2213        :       335
  469.8 - 687.7       :      4027  |      2213 - 2431        :       278
  687.7 - 905.7       :      2561  |      2431 - 2649        :       244
  905.7 - 1124        :      1678  |      2649 - 2867        :       203
   1124 - 1342        :      1160  |      2867 - 8755        :       750

Face aspect ratio statistics [unitless]:
   min      1.15974
   max      193.232
   median   1.53555
   mean     1.67684
   variance 4.54421

Face aspect ratio histogram [unitless]:
  1.155 - 1.5          :      9052  |        15 - 25          :         3
    1.5 - 2            :      8375  |        25 - 50          :         2
      2 - 2.5          :      1935  |        50 - 100         :         1
    2.5 - 3            :       438  |       100 - 300         :         3
      3 - 4            :       198  |       300 - 1000        :         0
      4 - 6            :        52  |      1000 - 10000       :         0
      6 - 10           :        12  |     10000 - 100000      :         0
     10 - 15           :         9  |    100000 -             :         0
  (Aspect ratio is longest edge divided by shortest altitude)

Edge length statistics [data units]:
   min      2.97847
   max      191.459
   median   39.0379
   mean     42.5984
   variance 385.693

Edge length histogram [data units]:
      0 - 0           :         0  |     37.36 - 42.6        :      3763
      0 - 5.939       :       155  |      42.6 - 47.84       :      3046
```

```
  5.939 - 11.18        :       641  |    47.84 - 53.07       :      2323
  11.18 - 16.41        :       579  |    53.07 - 58.31       :      1722
  16.41 - 21.65        :      1543  |    58.31 - 63.55       :      1340
  21.65 - 26.89        :      2881  |    63.55 - 68.78       :      1054
  26.89 - 32.12        :      3764  |    68.78 - 74.02       :       774
  32.12 - 37.36        :      4252  |    74.02 - 191.5       :      2283

Edge angle statistics [degress]:
   min      10.016
   max      308.474
   median   180
   mean     181.265
   variance 126.604

Edge angle histogram [degress]:
      0 - 0           :         0  |    178.3 - 181.3       :     17376
      0 - 160.3       :       819  |    181.3 - 184.3       :      1540
  160.3 - 163.3       :       299  |    184.3 - 187.3       :      1285
  163.3 - 166.3       :       393  |    187.3 - 190.3       :      1051
  166.3 - 169.3       :       557  |    190.3 - 193.3       :       799
  169.3 - 172.3       :       803  |    193.3 - 196.3       :       603
  172.3 - 175.3       :      1068  |    196.3 - 199.3       :       447
  175.3 - 178.3       :      1398  |    199.3 - 308.5       :      1682
```

After determining these qualitative topology attributes of the mesh, MESHALYZER executes quantitative measurements of the mesh characteristics. Not all of the characteristics can be measured for every mesh. In fact, the constellation of attributes described above dictate which characteristics will be reported for a given mesh. The following mesh characteristics are always measurable regardless of the mesh attributes. First, the total number of vertices, faces, and edges in the model are counted. MESHALYZER then determines the number of components (or separate objects) in the mesh file. A mesh object has one or more components. In our reconstructions the number of components was usually one but sometimes greater than one when the cell being reconstructed lay partially outside of the reconstructed volume. For open meshes MESHALYZER reports the number of boundaries found in the object. Here a boundary is defined as a set of open edges (with exactly one adjacent face) that are contiguous and whose circuit is a closed path. An open mesh has one or more boundaries, while a closed mesh has exactly zero boundaries. MESHALYZER flags vertices that lie really close to each other, so close in fact that they are effectively indistinguishable, and reports a count of these indistinguishable vertex pairs. Vertex pairs are considered indistinguishable

if they are separated by less than a distance epsilon, where epsilon is approximately 1E-10. Face edges that reference indistinguishable vertices are also counted and reported. The bounding box of the mesh object is determined using vertex locations. The cumulative surface area of all faces is compiled and reported. MESHALYZER identifies and reports face pairs that intersect each other. The default behavior is to simply return the number of intersecting faces, but additional command line options will have MESHALYZER report the identity of the intersecting faces. Note that no differentiation is made between intersections of faces in a single mesh component and intersections of faces from different components in the same mesh file as illustrated in Figure III.6. Both cases are considered self-intersections.

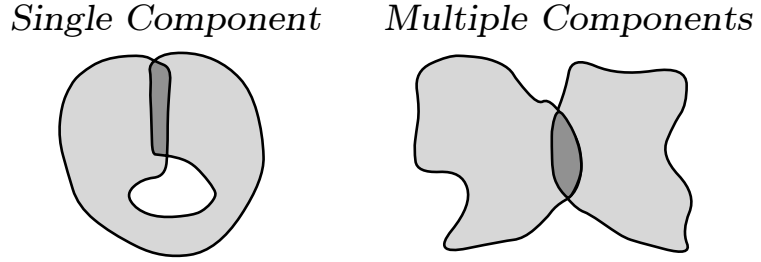

Figure III.6 **Self-intersecting meshes.** The faces of a single mesh object may intersect under two scenarios. A single component may intersect with itself. Also a mesh object may consist of more than one component, and these components may intersect. Distinguishing between these two scenarios is left to the user.

Finally, MESHALYZER measures the distribution of values for face area, face aspect ratio, edge length, and the number of adjacent faces for each vertex. For each of these distributions MESHALYZER reports the minimum, maximum, median, mean, and variance. Additionally, histograms of the values are presented, since the distributions need not be Gaussian nor symmetric. The minimum number of adjacent faces per vertex for a closed mesh object is three. Most vertices in the meshes used in the reconstructions have six adjacent faces. The number six arises because care is taken to keep the faces as equilateral as possible. Interior angles of equilateral triangles are 60 degrees, so a vertex on a flat plane will have 360

degrees of surface around which will accommodate six adjacent equilateral faces.

For reconstructions the ideal face shape is an equilateral triangle, corresponding to the minimum face aspect ratio of 1.155. Face aspect ratio is defined as the inverse of the ratio of the longest edge to the perpendicular distance from the longest edge to the third vertex. To understand the benefits of an equilateral mesh consider an alternative meshing with high aspect ratio faces as shown in Figure III.7. The process of reconstruction involves manipulating the location of the surface meshes by moving the vertices one at a time (Section II.1.F). One failure mode of a vertex move operation is the creation of degenerate faces, i.e. faces with zero area (or infinite aspect ratio) where all vertices lie along a straight line. One advantage of an equilateral triangle is its lower aspect ratio hence greater robustness to degeneracy during vertex moves. Additionally, one would like to be able to manipulate one region of a surface mesh without affecting other regions. However, when a vertex is moved all adjacent faces of the vertex move with it and the extent of the surface changes is a function of the lengths of the adjacent edges of the vertex. Compared to high aspect ratio faces equilateral triangles minimize the maximum edge length in the triangle, thereby maximally decoupling the mesh geometry.

Some mesh characteristics are only measurable if the mesh has specific attributes. For example edge angle can only be measured for meshes which are both manifold and have consistently-oriented face normals. An edge's angle is defined by the adjacent faces of the edge and is only measured for edges with exactly two adjacent faces, all other edges excluded.[3] This implies that edge angles can be measured for open meshes, since the edges with only one adjacent face will be ignored. As illustrated in Figure III.8, edge angles range from 0 to 360 degrees. Note that if face normals point outward, then exterior edge angles are measured; whereas if face normals point inward, then interior edge angles are measured. MESHALYZER measures the distribution of values for edge angle and reports the

[3]The edge angle is not to be confused with the interior angles of triangular polygons which are not calculated explicitly but are indirectly measured by the face aspect ratios.

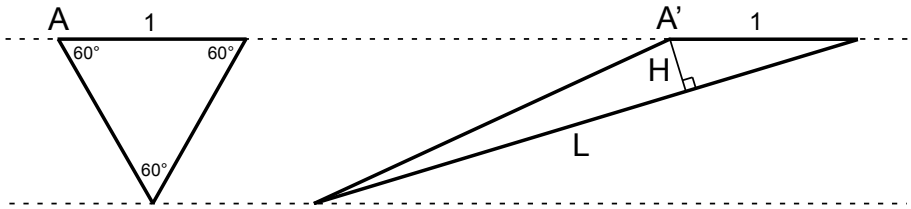

Figure III.7 **Equilateral triangular meshes are superior to high aspect ratio meshes.** The equilateral triangle on the left is the preferred geometry for meshing over the nonequilateral triangle on the right for two reasons. While both triangles have the same area, the equilateral triangle has a lower aspect ratio (1.155 compared to 10) which means that point A can be moved a large distance without making the triangle degenerate (zero area). In contrast, merely moving point A' of the triangle on the right a distance H would make the triangle degenerate. Secondly, the triangle on the right is less localized than the equilateral triangle because of the high aspect ratio face's longer edges. Here edge L is three times longer than the edge lengths of the equilateral triangle.

minimum, maximum, median, mean, and variance and a histogram of values.

For meshes that are manifold, have consistently-oriented face normals and are closed, two additional characteristics can be measured - volume and genus. If face normals point inwards, then the calculated volume will be negative, but the magnitude will be correct. MESHALYZER computes mesh object genus using a version of the Euler-Poincare formula: vertices-edges+faces=2-2*genus. Genus can be thought of as a measure of the number of holes in the meshed object. For example, a sphere has genus zero, while a toroid has genus 1.[4] This formula applies to individual components only. The current version of MESHALYZER requires that the mesh object be a single component to compute the genus. Future versions of MESHALYZER could compute the genus of multiple-component objects by

[4]Actually, the formula only requires that the mesh be orientable. In other words future versions of MESHALYZER could compute the genus of an object if the face normals could be consistently-oriented without requiring that they currently be consistently-oriented.

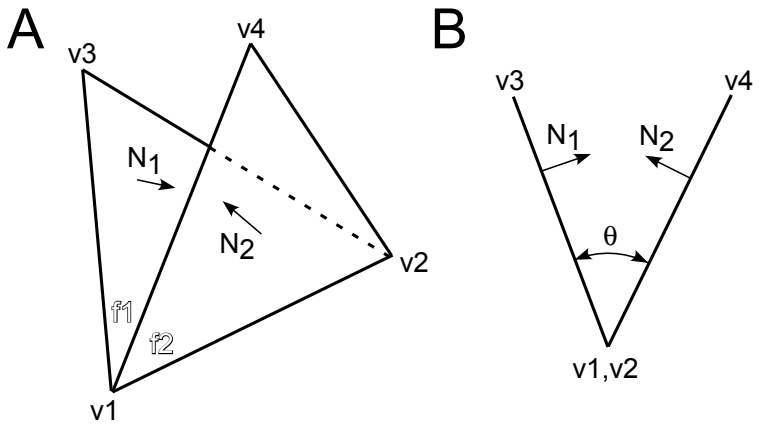

Figure III.8 **Edge angle is defined by adjacent faces of edge.** MESHALYZER adopts the convention to measure and report angle θ (as opposed to angle 360-θ) using face normal vectors.

examining each component separately.

III.4 Quick-Reference Guide

The same documentation listed below is available be executing MESHALYZER with the '-h' option on the command line.

```
NAME
       meshalyzer - mesh quality analyzer

SYNOPSIS
       meshalyzer [options] FILE|DIR

DESCRIPTION
       Meshalyzer is a general purpose mesh analyzer useful for
       generating a complete summary of the current state of a mesh.
       Meshalyzer assesses mesh integrity (e.g. missing data),
       mesh attributes (e.g. closed, manifold, oriented), and
       mesh characteristics (e.g. number of vertices, faces, edges).
       Batch processing is easy by passing a directory name
       as input on the command line.

EXAMPLES
       meshalyzer filename
              Evaluate mesh integrity, attributes,
              and characteristics for the single mesh file.
```

```
       meshalyzer directoryname
              Evaluate mesh integrity, attributes,
              and characteristics for each single mesh file in directory.

       meshalyzer -p filename
              Evaluate mesh integrity, attributes, and characteristics
              for the single mesh file, print the results, and print
              the mesh elements preventing the mesh from being good with
              regards to the mesh characteristics and the attributes, if any.

       meshalyzer -a -p filename
              Evaluate the five mesh attributes for the single mesh file,
              print the state of each attribute, and print the mesh elements
              preventing the mesh from being good with regards to the attributes, if any.

       meshalyzer -b 10.0 -c 1.0 -p filename
              Evaluate mesh integrity, attributes, and characteristics
              for the single mesh file, print the results, and print
              the mesh elements preventing the mesh from being good with
              regards to the mesh characteristics and the attributes, if any.
              Additionally, screen faces with aspect ratios larger than 10.0 and
              screen edges with lengths larger than 1.0, and print detailed
              information about the offending mesh elements.

OPTIONS
       -a
              Evaluate the attributes of the mesh and report the results.
              Skip the evaluation of mesh characteristics.

       -b NUM
              Detect edges with length smaller than NUM.
              Units are same as FILE.

       -c NUM
              Detect edges with length greater than NUM.
              Units are same as FILE.

       -d NUM
              Detect edges with angle between two adjacent faces
              smaller than NUM degrees.

       -e NUM
              Detect edges with angle between two adjacent faces
              greater than NUM degrees.

       -f NUM
              Detect faces with aspect ratio greater than NUM.

       -h
              Print meshalyzer man page.

       -i
              Detect intersections between faces from different objects.
              Faceintersection detection is performed once all objects
              are loaded into memory. Single object intersection detection
              is omitted.

       -p
              Print detailed information about offending mesh elements
              (i.e. flipped faces, borders, nonmanifold edges,
```

```
              nonmanifold vertices, intersecting faces).

       -q
              Same as '-p' option, but prints vertex information
              in dreamm custom points format.
       -v
              If folder passed as argument, then only print total
              set volume and nothing else.

Justin Kinney 2007/10/01
```

III.5 Acknowledgements

This chapter is, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Thomas M. Bartol and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

IV

Monte Carlo simulation of glutamate spillover in simplified model of hippocampal neuropil

IV.1 Abstract

Following vesicular release of glutamate in hippocampal CA1 synapses, the neurotransmitter diffuses out of the synaptic cleft into the extracellular space, a phenomenon called "spillover". Whether glutamate spillover leads to synaptic crosstalk and how crosstalk would affect synaptic plasticity is unknown. We constructed a simplified 3D model of hippocampal neuropil and used the model to perform Monte Carlo simulations of spillover following high-frequency burst release of neurotransmitter. This chapter describes the model geometry, glutamate receptor and transporter kinetic models, and quantitative observations of spillover and crosstalk in glutamate diffusion simulations. We found that a constant 5 Hz synaptic firing rate yields a steady-state mean extracellular glutamate concentration of 0.5 μM that is neither temporally nor spatially uniform. With this background glutamate concentration, 15% of glutamate transporters have bound glutamate and NMDARs exhibit significant priming. The mean radial diffusion distance of

a quantum of transmitter was independent of quantal size over the range tested. More than 90% of the diffusing neurotransmitter stays within 2 μm of the release site after synaptic vesicular release. Glutamate time course and receptor activation in the active synapse are largely insensitive to the presence of glutamate transporters in the extrasynaptic space. Glutamate transporters suppress NMDAR spillover activation almost entirely, while AMPAR spillover activation is nonexistent with or without transporters. Concentrating glutamate transporters near synapses yields only marginally different amounts of spillover than uniform transport density model. Glutamate transporters are not saturated with vesicle size of 3000 glutamate even after 100Hz burst in either model. Peak spillover EPSCs were 20% of EPSCs at the active synapse. Our results suggest that glutamate spillover is insignificant in neuropil models with canonical geometry and can be ignored. However, it remains to be seen whether spillover is relevant in the heterogeneous milieu of real neural tissue.

IV.2 Introduction

Glutamatergic synapses are the dominant excitatory synapse type in the hippocampus and cortex (Danbolt, 2001). Neurons in hippocampus communicate via synaptic transmission whereby presynaptically-released glutamate diffuses across a synaptic cleft and binds to postsynaptic receptors. Glutamate is not confined to the synaptic cleft but instead diffuses into the extracellular space (ECS) where it is rapidly removed from the ECS and sequestered inside astrocytes richly adorned with transporters which bind and translocate glutamate. Whether some of the glutamate evades capture by astrocytes and diffuses into neighboring synapses (spillover) where it binds to receptors (crosstalk) is unknown (Figure IV.1).

The goal of this project was to construct a simplified three dimensional (3D) model of the hippocampal neuropil and use it in a Monte Carlo computer simulation of synaptic neurotransmitter release to characterize the effect of geometrical

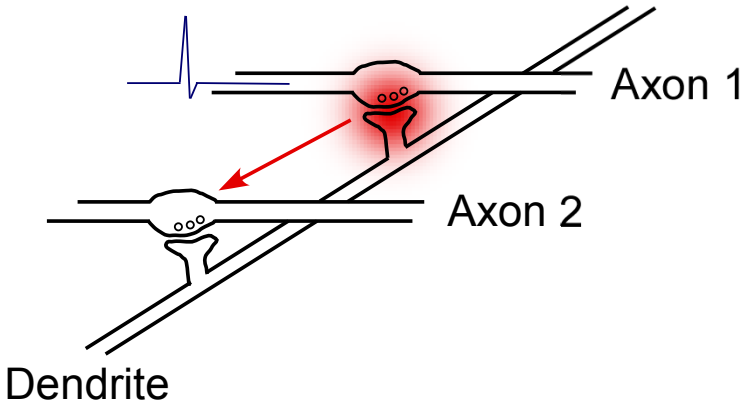

Figure IV.1 **Definition of Crosstalk**. Suppose an action potential on Axon 1 causes synaptic release of neurotransmitter. The crosstalk due to spillover of neurotransmitter causes excitation of a neighboring synapse on the dendrite as if Axon 2 had also fired.

and biophysical parameters of hippocampal neuropil on glutamate spillover. The 3D neuropil model incorporated known morphological constraints on membrane geometry and transporter distributions. The model was used to assess the impact of several novel model parameters on spillover, including (1) nonzero background glutamate concentration, (2) synaptic burst release of neurotransmitter, and (3) transporter distribution. We report glutamate concentration time courses in both the synaptic clefts and extrasynaptic space, allowing calculation of mean radial diffusion distances for individual vesicle populations. Transporter occupancy was monitored around active synapses to assess the efficacy of transporters on limiting glutamate spillover. Glutamate receptor activity is reported not just in terms of open probability, but also receptor occupancy and desensitization. The simulations were repeated in a second-generation 3D model designed to correct morphological inaccuracies in the original model. The new model was also used to explore the effect of nonuniform surface density of glutamate transporters on glutamate spillover and crosstalk.

IV.3 First Generation 3D Model

In this section, the construction of the first-generation model is described in detail and the results of room-temperature simulations of background glutamate concentration are presented. It was found that synaptic firing rates of 5 Hz produce an extracellular glutamate concentration of 0.5 μM; however, the neurotransmitter concentration is neither temporally- nor spatially-uniform (Figure IV.5). The background glutamate concentration results in approximately 15% of the glutamate transporters being occupied, and some priming of NMDA receptors is observed, while none is seen for AMPARs (Table IV.4). Specifically, 10% of NMDARs are doubly-bound, sensitized and 30% are singly-bound. The half-life of one quantum of transmitter in the extracellular space is approximately 0.5 ms and increases with quantal size and background glutamate concentration (Section IV.3.C). The mean radial diffusion distance of a quantum of transmitter was independent of quantal size over the range tested (Section IV.3.D). More than 90% of the diffusing neurotransmitter stays within 2 μm of the release site after synaptic vesicular release (Figure IV.6).

Spillover simulations in the first-generation model with zero background glutamate show that the glutamate time course and receptor activation in the active synapse are largely insensitive to the presence of glutamate transporters in the extrasynaptic space (Figure IV.8). In the absence of transporters, spillover activation (open state) of NMDARs is 25% of peak levels in the active synapse (Figure IV.9). Glutamate transporters suppress NMDAR spillover activation almost entirely, while AMPAR spillover activation is nonexistent with or without transporters. Increased transporter occupancy levels around the active synapse (Figure IV.13) explains small increases in glutamate spillover distance (Figure IV.12). Transporters are not saturated when measured with 60 nm bins, consistent with previous model results (Barbour, 2001).

IV.3.A Model Design

Geometry

A first-generation 3D model of neuropil was constructed and used as a simulation environment in MCell. This simplified model of a 5 x 5 x 5 μm^3 region of hippocampus consists of 1 x 1 x 1 μm^3 motifs stacked in a 3D array (Figure IV.2B). The motif contains four independent synapses, consisting of a presynaptic bouton and postsynaptic spine, surrounded by glial processes (Figure IV.3A and Section A.3). Simple boxes are used to model cell processes. The motif was designed to match experimentally measured values of the hippocampal neuropil microstructure. The intersynaptic distance is 0.52 μm (Ventura and Harris (1999), Rusakov et al. (1999)), extracellular volume fraction is 0.177 (Sykova, 1997), and a uniform inter-membrane spacing of 20 nm is used (Rusakov and Kullmann, 1998a). The glial surface area density is 11 μm^2 per cubic micron of neuropil which is one order of magnitude higher than estimates in literature (Lehre and Danbolt, 1998).[1] The glial surface density of glutamate transporters was decreased by a factor of ten to achieve the correct volume concentration of transporters.

The synapse is square, 0.38 μm on a side, separated by a 20 nm gap, narrowing to 10 nm at the edges (Rusakov and Kullmann, 1998b) with a postsynaptic density 0.31 μm in diameter (Harris and Stevens, 1989). This synapse area is within the range described by Ventura and Harris (1999). At each postsynaptic surface, there are 80 AMPARs (i.e. 1058/μm^2) and 20 NMDARs (i.e. 265/μm^2) uniformly distributed (Franks et al., 2002). Experimental evidence has been shown for co-localization of AMPARs and NMDARs in at least 75% of Schaffer-collateral-commissural synapses onto CA1 pyramidal cells (Racca et al., 2000). All glial surfaces were populated with transporters at a surface density of 1600/μm^2 (Lehre and Danbolt, 1998).

[1] Glial surface area density was corrected in the second-generation model (Section IV.4.A).

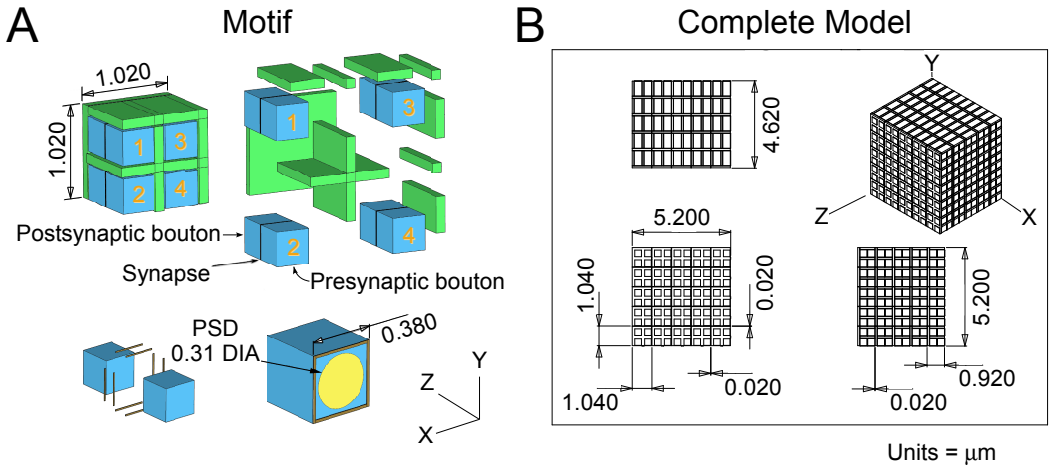

Figure IV.2 **First generation neuropil model was designed from morphometric values gleaned from literature.** (A) Detail of the model motif containing four idealized synapses (blue) surrounded by glial processes (green). The postsynaptic density (yellow) contains 80 AMPARs and 20 NMDARs. The intermembrane spacing is uniformly 20 nm. The synaptic cleft width is also 20 nm, narrowing to 10 nm at the synapse perimeter. Glial surfaces contain glutamate transporters with a surface density of 1600 per μm^2. Dimensions in microns. (B) Complete First Generation Model composed of 125 Motifs.

AMPAR, NMDAR, and Glutamate Transporter Kinetic Models

The kinetic model for AMPA receptors is based on patch clamp measurements of unitary excitatory postsynaptic currents of the mossy fiber terminal-CA3 pyramidal cell synapse in rat hippocampus at 22 °C (Jonas et al., 1993) (Figure IV.3). The kinetic model for NMDARs is taken from Lester and Jahr (1992) also based on outside-out patch-clamp recordings from cultured rat hippocampal neurons. Rate constants for AMPAR and NMDAR are given in Table IV.1 and Table IV.2.

The glutamate transporter model used here is identical to that used by Geiger et al. (1999), which is itself a simplified version of the schemes described

in previous reports (Tong and Jahr (1994), Takahashi et al. (1996), Diamond and Jahr (1997), Otis and Jahr (1998)). The kinetic model and rates conform to the previously published measured values of apparent affinity (approximately 20 μM, (Arriza et al., 1994)) and slow turnover rate (10-20 s^{-1}, (Wadiche et al., 1995)). No distinction is made between the two dominant types of glutamate transporter in rat hippocampus astroctyes: GLT-1 and GLAST. The physiology of these two transporter subclasses and a proposal for separating the transporter population along these lines is discussed in Section D.4. Rate constants for the EAAT model are given in Table IV.3.

Table IV.1 Rate Constants for AMPA Receptor.

C0C1	C1C0	C1C2	C2C1	C2O	OC2
$M^{-1}s^{-1}$	s^{-1}	$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}
4.59e6	4.26e3	2.84e7	3.26e3	4.24e3	900
C1C3	C3C1	C3C4	C4C3	C2C4	C4C2
s^{-1}	s^{-1}	$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}
2.89e3	39.2	1.27e6	45.7	172	0.727
C4C5	C5C4	OC5	C5O		
s^{-1}	s^{-1}	s^{-1}	s^{-1}		
16.8	190.4	17.7	4		

Table IV.2 Rate Constants for NMDA Receptor.

C0C1	C1C0	C1C2	C2C1	C2D
$M^{-1}s^{-1}$	s^{-1}	$M^{-1}s^{-1}$	s^{-1}	s^{-1}
1e7	4.7	5e6	9.4	8.4
DC2	C2O	OC2	OC1	
s^{-1}	s^{-1}	s^{-1}	s^{-1}	
1.8	46.5	91.6	9.4	

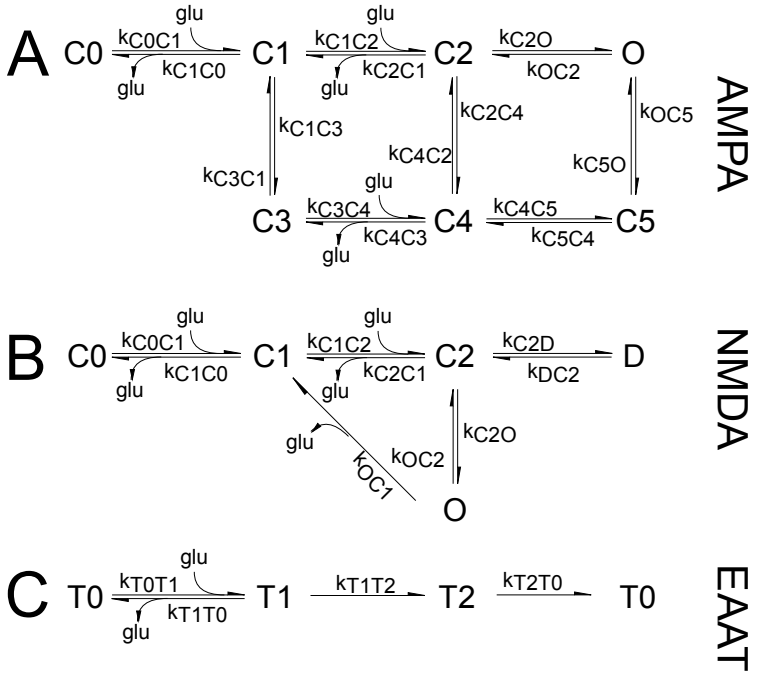

Figure IV.3 **Kinetic models for AMPARs, NMDARs, and Excitatory Amino Acid Transporter.** (A) C0 is the unbound closed state, and C1 and C2 are bound closed states. C3, C4, and C5 are desensitized, closed states, and O is the doubly-bound open state. Conductance of open state is 10 pS (Spruston et al., 1995). (B) C0 is the unbound closed state, C1 and C2 are singly-bound and doubly-bound closed states, respectively, D is the desensitized, closed state, and O is the doubly-bound open state. Conductance of open state is 45 pS (Franks et al., 2002).[3] (C) This excitatory amino acid transporter (EAAT) kinetic model has three states (unbound, T0; reversibly-bound, T1; irreversibly-bound, T2) with four reaction rates (binding, T0T1; unbinding, T1T0; transition to irreversibly-bound, T1T2; translocation and resetting, T2T0).

Tortuosity

The geometric tortuosity of the first-generation model was measured according to Tao and Nicholson (2004). An MCell simulation was performed where 1000 molecules were released in the center of the model (with no transporters) and

Table IV.3 Rate Constants for Glutamate Transporter (EAAT).

T0T1	T1T0	T1T2	T2T0
$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}
1.8E7	180	180	25.7

allowed to diffuse for 1000 ms. Concentric, transparent sampling cubes recorded the diffusion process by counting the number of molecules contained in each box every 20 ms as shown in Figure IV.4. A curve fitting algorithm solved for the geometric diffusion constant, D_g, that best fit each of the eight data sets. Using the average geometric diffusion constant and the particle diffusivity programmed in MCell (D_{mcell}), the geometric tortuosity was calculated as

$$\lambda_g = \sqrt{\frac{D_{mcell}}{D_g}},$$

and equals 1.42, higher than predicted by Tao and Nicholson for convex polyhedra based on the EVF since the model is a regular array of *concave* polyhedra. Since D_{mcell} is set to $D/3$ for all simulations with the first-generation model (where D is the diffusivity of glutamate in water), the total tortuosity is approximately 2.46, much larger than the target 1.6 (Section A.2). This excessive diffusional impedance is corrected in simulations with the second-generation model.

IV.3.B Achieving Nonzero Background Glutamate Concentration

The extracellular glutamate concentration in parietal cortex and hippocampus as measured by microdialysis has been reported to be in the range of 2 to 5 μM with fluctuations up to 0.5 μM over 5 minutes (Hamberger and Nyström (1984), Hazell et al. (1993), Zhang et al. (2005), but see Herman and Jahr (2007)). The origin of this glutamate signal is unknown and could be due to vesicular release from glutamatergic neurons, astrocytes (Volterra and Meldolesi, 2005), or stoichiometry of the glutamate transporters themselves (Bouvier et al., 1992).

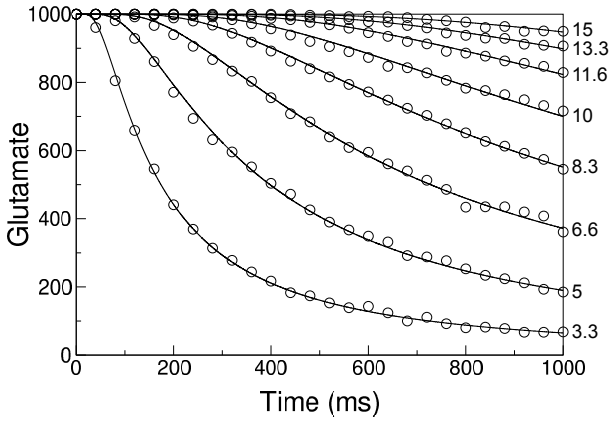

Figure IV.4 **Measurement of geometric tortuosity for first generation neuropil model.** Tortuosity measurement as per Tao and Nicholson. The number of molecules in eight concentric sampling boxes was recorded every 20 ms (circles). An analytical description of each population time course was fit (line). Numbers on right side indicate size of sampling box in microns.

Here we treat this low neurotransmitter concentration as a constant background glutamate signal, weak in comparison to the millimolar glutamate concentrations reached inside the synaptic cleft (Clements et al., 1992). Of scientific interest is whether an elevated basal level of glutamate suppresses crosstalk by increasing NMDA receptor desensitization or exacerbates crosstalk by priming NMDA receptors via partial receptor activation. By assessing the impact of a spatially- and temporally-uniform background glutamate concentration on spillover, we evaluate the relevance of a weak background glutamate signal to crosstalk and investigate the source of the weak extracellular glutamate concentration.

Spillover was measured with a nonzero extracellular glutamate concentration and compared to the results with zero background extracellular glutamate. To approach a spatially- and temporally- uniform background glutamate concentration, single glutamate molecules were released at a specified grid of N extracellular sites some of which were extrasynaptic. The release probability p, uniform across all sites and constant during the simulation, was calculated to yield a desired av-

erage release rate r in the model ($p = r/N$). In the model glutamate transporters bind extracellular glutamate, translocate it across the membrane, and destroy it. As the extracellular concentration of glutamate and the distribution of transporter states approach steady-state values, the rate of glutamate removal from the extracellular space by glial transporters balanced the rate of glutamate release. While the release of neurotransmitter from extrasynaptic locations on a per molecule basis ass certainly nonphysiological, it allows inquiry into a scientifically interesting question, namely what are the effects of a constant low concentration of glutamate on spillover. In fact, we find that the idea of a spatially- and temporally-uniform background extracellular glutamate concentration is only valid when averaged over 100s of microns and 100s of milliseconds such as occurs in microdialysis probes, and that on the scale of individual synapses much heterogeneity of concentration exists.

Uniformity of Glutamate Concentration

Single glutamate molecules were released from 1100 spatially-distributed locations, both synaptic and extrasynaptic, in a probabilistic manner such that on average 5.88 glutamate molecules were released each microsecond to achieve a temporally- and spatially-uniform 0.5 μM background concentration (Figure IV.5A). The predicted steady glutamate release rate required to achieve 0.5 μM concentration based on well-mixed system is 6.62 molecules per μs (Appendix A.1) which agrees well with the actual rate. The small discrepancy may be due to deviations from well-mixed assumption, i.e. spatial-gradients in glutamate concentration. The steady release drives the system to an equilibrium total glutamate count of 6722, corresponding to 0.5 μM, regardless of the amount of glutamate in the system at the initial condition.

The implications of the required steady-release rate in the context of this model are interesting. Consider that this model has 250 synapses and suppose a hypothetical average vesicle size of 3000 glutamate molecules. A steady release

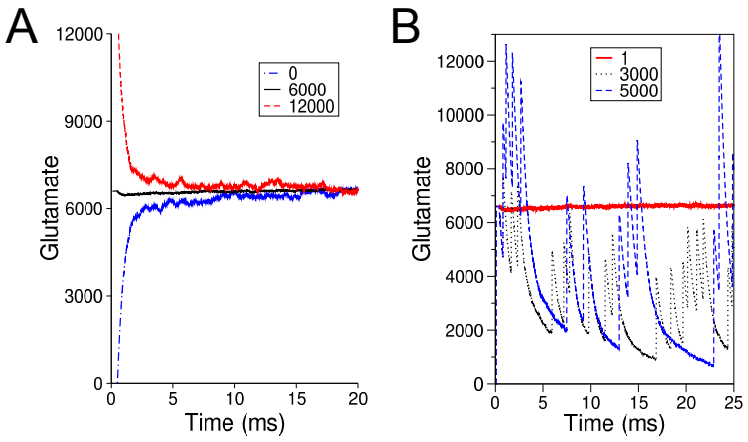

Figure IV.5 **Glutamate release and uptake by transporters determine steady-state glutamate concentration in model.** (A) Global glutamate count converges to target value regardless of initial condition. Legend indicates initial number of glutamate molecules in model. (B) Comparison of synaptic vesicular release and extrasynaptic single molecule release. Legend indicates number of glutamate molecules released per release site.

rate of 5.88 glutamate molecules per μs could be achieved in a more physiological manner by initiating an average synaptic firing rate of approximately 8 Hz. Assuming a 5000 molecule vesicle, the rate drops to 5 Hz. Both of these values are well within known physiological firing regimes of CA3 neurons (Buzsáki, 2002).

If we constrain glutamate release to synapses with vesicle-sized quanta and repeat the MCell simulation, it becomes clear that the resulting background glutamate concentration will be neither temporally-constant nor spatially-uniform. Figure IV.5B compares the total glutamate count resulting from single molecule release from 1100 sites with synaptic vesicular release of 3000 and 5000 glutamate. Clearly, if synaptic vesicular release is the only source of glutamate in the model, the total glutamate count can range from 1 μM to near zero. These simulations predict that the background glutamate concentration is neither temporally- nor spatially-uniform on the scale of millisecond and microns. Experimental validation

of this assertion will likely require an imaging approach, since microdialysis probes do not have the spatial resolution to reveal glutamate concentration gradients on these scales. Alternatively, glutamate release from astrocytes could serve to smooth out the glutamate concentration nonuniformities and contribute to more constant background glutamate signal (Volterra and Meldolesi, 2005).

Effect of Background Glutamate on NMDARs

Glutamate's high affinity for NMDARs (equilibrium dissociation constant $K_D \sim 3\mu$M (Patneau and Mayer, 1990)) suggests that a fraction of the NMDAR population will have bound glutamate given a sufficient background concentration of neurotransmitter. To quantify this effect four different analyses were compared (Table IV.4). First, the steady-state NMDAR, AMPAR, and EAAT state distributions from the aforementioned unimolecular, steady-release MCell model (Uniform) and the more physiological synaptic vesicle release MCell model (Spikey) were recorded (Figure IV.5B). Second, a differential equation model (Analytic) was built (Appendix A.1) assuming homogeneous concentrations of glutamate and transporters to calculate the expected amount of glutamate binding to the NMDARs. Third, an MCell simulation was configured with modified kinetic models of receptors and transporters (Glu-Indep); the glutamate binding step was replaced by a constant 0.5 μM glutamate concentration assumption. As a consequence, state transition was independent of glutamate, and no molecules were released. This simulation allowed a comparison between the kinetic models used in the analytic treatment and MCell. Furthermore, it provided a way to investigate the assumption spatial uniformity of the glutamate concentration.

The general trend exhibited by all four models was the consumption of a fraction (15%) of glutamate transporters, a large shift of NMDARs from the unbound state to the single-bound and desensitized states, and no change in AMPAR state distribution. The analytical and glutamate-independent model results are identical as expected and similar to the state distributions for the uniform MCell

simulation (Table IV.4). The uniform MCell simulation exhibits transporter distributions skewed to the unbound state and AMPAR and NMDAR distributions skewed to the bound state. What is different about this MCell simulation? The total number of free glutamate molecules in the extracellular space for the MCell simulation was confirmed to match the expected value of 0.5 μM (data not shown). Perhaps the glutamate molecules are concentrating inside the synapses of the uniform model leaving the extrasynaptic space at a lower glutamate concentration. The mean *synaptic* glutamate concentration in the uniform MCell simulation was approximately 0.7 μM higher than the average concentration of the entire ECS. In this simulation approximately half of the glutamate release sites were inside synapses, so one interpretation is that the cleft narrowing was restricting the diffusion of glutamate out of the cleft. There was no evidence that glutamate was accumulating in the synapse over the course of the simulation (data not shown) implying only a spatial-nonuniformity of glutamate concentration. Interestingly, the Spikey model had a unique profile with a transporter distribution similar to the Uniform model, an AMPAR distribution similar to the Analytic model, and an NMDAR distribution between the Analyitc and Uniform models. The interpretation of the Spikey profile based on statistics averaged over all synapses and all time steps as presented in Table IV.4 is difficult. Future work will entail a more detailed inquiry of the Spikey model by separating the state distributions of the active and inactive synapses.

IV.3.C Decay Rate of Vesicular Glutamate with 0.5 μM Background Glutamate

The time course of glutamate transport of single vesicles released from each of four synapses in a single motif with a 0.5 μM background glutamate concentration was recorded. On a semi-log plot the profile has a dual exponential decay and exhibits virtually no dependence on which of the four synapses in the motif is chosen for release. Half-life values were found to be independent of release

Table IV.4 **Influence of 0.5 μM background glutamate concentration on receptor and transporter state distributions.** The fraction of transporter and receptor populations in each state for four simulations - (Analytic) a differential equation model, (Glu-Indep) an MCell simulation with modified kinetic models of receptors and transporters where the glutamate binding step was replaced by a constant 0.5 μM glutamate concentration assumption, (Uniform) a unimolecular, steady-release MCell model, and (Spikey) the more physiological synaptic vesicle release MCell model. AMPAR and NMDAR states are averages over all synapses in the model. $\varepsilon < .001$.

State	Description	Analytic	Glu-Indep	Uniform	Spikey
EAAT-T0	Unbound	0.83	0.83	0.85	0.85
EAAT-T1	Single	.02	.02	.02	.02
EAAT-T2	Single	0.15	0.15	0.13	0.13
AMPA-C0	Unbound	0.962	0.962	0.88	0.945
AMPA-C1	Single	ε	ε	ε	ε
AMPA-C2	Double	ε	ε	ε	ε
AMPA-C3	Double	.038	.038	0.12	0.055
AMPA-C4	Double	ε	ε	ε	ε
AMPA-C5	Double	ε	ε	ε	ε
AMPA-O	Double	ε	ε	ε	ε
NMDA-C0	Unbound	0.3	0.3	0.16	0.21
NMDA-C1	Single	0.3	0.3	0.31	0.28
NMDA-C2	Double	0.05	0.05	0.1	0.08
NMDA-D	Double	0.3	0.3	0.4	0.4
NMDA-O	Double	0.05	0.05	0.03	0.03

synapse, and the values presented in Table IV.5 apply to all four release positions. The average half-life of glutamate was 666 ms for vesicle size of 5000 and 591 ms for vesicle size of 3000. Results for zero initial glutamate concentration were comparable (data not shown).

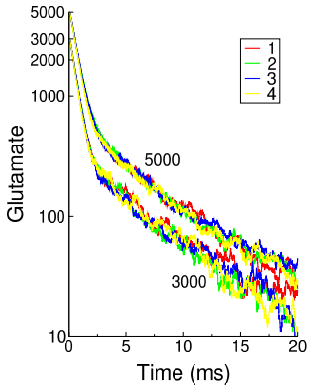

Figure IV.6 **Vesicular glutamate decay time course is insensitive to motif asymmetry.** With 0.5 μM background glutamate concentration, vesicles containing 3000 or 5000 glutamate molecules were released singly from each of four synapses in motif. The time course of each quantum of glutamate was recorded after vesicular release. Legend indicates release synapse as defined in Figure IV.2A. Results for zero initial glutamate concentration were comparable (data not shown).

IV.3.D Glutamate Diffusion Extent and Decay Rate with Zero Background Glutamate

Starting with zero initial background glutamate concentration, vesicles were released one at a time from the cleft center of each of the four synapses in the motif. Interestingly, motif asymmetry causes variation in mean radial diffusion displacement of glutamate from the point of release among the four synapses following synaptic release (Figure IV.7). The minimum radial diffusion distance of approximately 0.21 μm reflects the absence of glutamate transporters inside the synapse. The mean radial diffusion distance varied slightly among the four synapses but was virtually the same for the two vesicle sizes. Because the spillover is largely contained within a 4 μm diameter region, boundary effects in a 5 μm cubic model are small when glutamate release occurs near the center of the model. The half-lives of the glutamate quanta was slightly shorter than for the simulations with nonzero background glutamate as expected (Table IV.5). The background

glutamate occupies approximately 15% of the transporters increasing the amount of diffusion required for a glutamate molecule to find an unbound transporter.

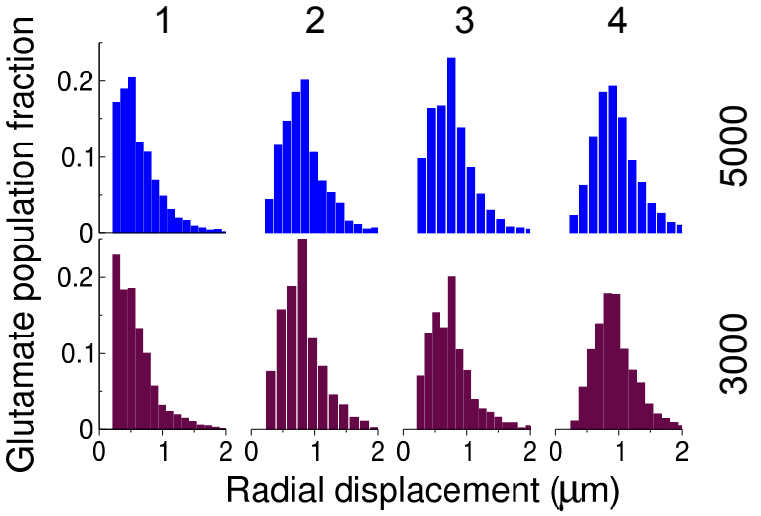

Figure IV.7 **Motif assymetry affects spatial spread of diffusing glutamate.** Motif asymmetry causes variation in mean radial diffusion displacement of glutamate from the point of release among the four synapses following synaptic release. Numbers across top indicate active synapse, and numbers along rights side indicate number of glutamate molecules per vesicle.

Table IV.5 Half-lives of the contents of single vesicles.

Vesicle Size	Zero Background Glu	0.5 μM Background Glu
3000	511 μs	591 μs
5000	596 μs	666 μs

IV.3.E Burst Release Simulations with Zero Background Glutamate Concentration

Simulations of high-frequency activity in the first-generation model consisted of a 100 Hz burst of 5 release events initiated in a single active synapse

(synapse 4 in Figure IV.2A) beginning at time zero. Each vesicle contained 3000 glutamate molecules and was released instantaneously in the center of the synaptic cleft. After each simulation time step of 1 μs the cleft glutamate concentration and state of each receptor on the opposing postsynaptic membrane were recorded for the active synapse (synapse 4) and neighboring synapse (synapse 2). Additionally, the state of each transporter in the model was tracked to evaluate the effect of transporter occupancy on spillover. Simulations were repeated in a model with no transporters to evaluate the effect of glutamate transport on the measured responses. The simulation results from fifty different random number seeds were averaged to reduce stochastic noise. In all simulations the glutamate diffusivity and kinetic rates are room temperature values.

Receptor Activation

Simulation results of high frequency burst release in the first-generation model are consistent with experimental measurements in slice and with previous modeling efforts. AMPA receptors behave like differentiators, sensitive to the rate of change of the glutamate concentration, while NMDA receptors act as integrators and respond to the cumulative amount of glutamate present in the synapse. It is evident from the data that the presence of glutamate transporters is insignificant to receptor activation in the active synapse but profoundly suppresses activation at the neighbor synapse.

Figure IV.8A is a plot of the glutamate concentration in the active synapse following each release. The time courses from each release were aligned at time zero for comparison. In the model with no transporters, the glutamate time course is invariant to subsequent release (Figure IV.8A inset), and the curves are well fit by a double exponential with fast and slow decay time constants of 0.08 ms and 0.66 ms, respectively. In contrast, with transporters present the first release exhibited an accelerated decay phase, i.e. slow decay time constant of 0.43 ms, while subsequent releases resulted in intermediate decay rates. Increased transporter

occupancy around the active synapse explains the slowing of the glutamate time course during the burst (Figure IV.12). Less glutamate binds to transporters and instead diffuses back into the synaptic cleft. Assuming single channel conductances of 10 pS and 45 pS for AMPARs and NMDARs, respectively, the EPSC (Figure IV.8) for a hypothetical voltage-clamp experiment at -70 mV was computed from the receptor activation (Figure IV.9). The EPSC is reasonably fit by a double exponential (green line) with rising and falling decay time constants of 8 ms and 95 ms, respectively. With no transporters the EPSC at the neighbor synapse exhibited a slowly-rising, small inward current with peak amplitude of approximately 5 pA. Virtually no receptor activation was seen at the neighbor synapse with transporters present.

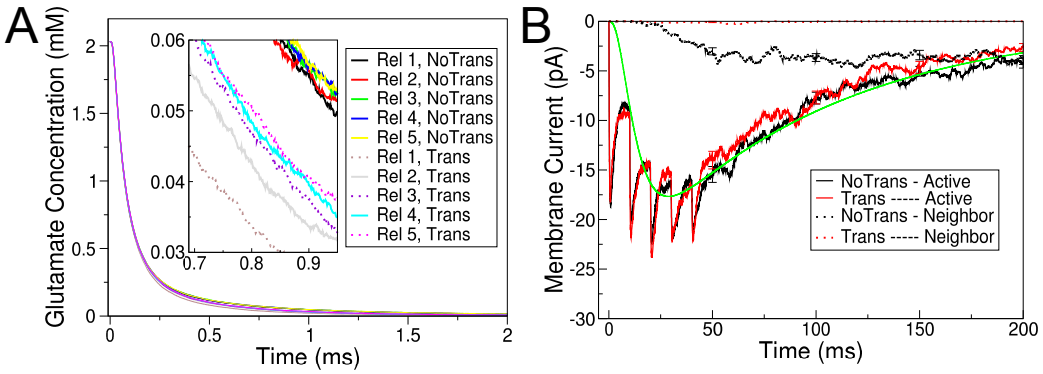

Figure IV.8 **Glutamate transporters dramatically diminish spillover but only marginally affect glutamate time course in the active synapse.** (A) Synaptic cleft glutamate concentration time course in a model with and without transporters. Transporter saturation slows cleft glutamate concentration decay in burst release. Average of 50 seeds. (B) Excitatory postsynaptic potential following burst release. Green curve was tuned to NMDAR-mediated response at the expense of a good fit for high-frequency AMPAR response.

Plotting the fraction of AMPARs and NMDARs in a synapse in the open state illustrates how the different unbinding rates of the two receptor types leads

to dramatically different response profiles (Figure IV.9). Due to a fast glutamate unbinding rate the number of AMPA receptors in the open state returns to zero between releases in the burst (Figure IV.9A). The AMPA response following the first release is well fit by a double exponential with rising and falling phase time constants of 0.15 ms and 2.55 ms, respectively (Figure IV.9A inset). In contrast to the differentiator-like behavior of AMPARs, the NMDAR response is an integration of the glutamate concentration during the burst (Figure IV.9B). The five distinct pulses seen in the AMPAR response are replaced by a slow rise and fall with time constants of 18 ms and 92 ms, respectively, in the NMDAR response at the active synapse. The declining peak response of AMPARs is due to accumulating receptor desensitization (Figure IV.11A). In the model with no transporters the fraction of open NMDA receptors at the neighboring synapse rose slowly and plateaued around the value 0.06, while adding transporters diminished the fraction of open NMDA receptors to near zero. The AMPAR response at the neighbor synapse was virtually nonexistent regardless of transporters (data not shown).

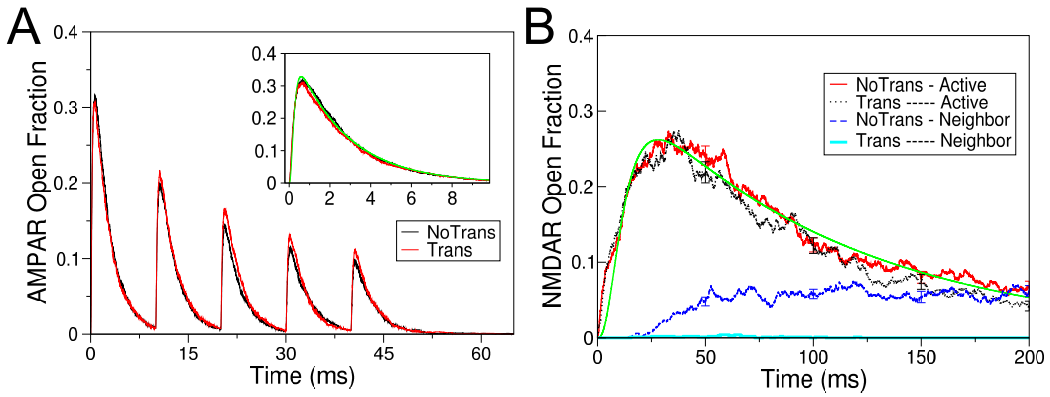

Figure IV.9 **Glutamate transporters dramatically diminish NMDAR-mediated crosstalk but only marginally affect AMPAR or NMDAR activation in the active synapse.** Fraction of (A) AMPARs at the active synapse and (B) NMDARs at the active and neighbor synapse in open state. Average of 50 seeds.

We computed AMPA and NMDA receptor occupancy (the fraction of the receptor population that has two bound glutamate molecules) at the active and neighbor synapses (Figure IV.10). Occupancy increases during the burst, but neither receptor type is saturated after the first release, consistent with previous studies (Oertner et al. (2002), Conti and Lisman (2003)). NMDAR occupancy rises faster than that of AMPARs and approaches saturation (100% occupancy) after the fourth release. These results are consistent with the slower unbinding rate of NMDARs (Kullmann et al. (1999), Franks et al. (2003), Tong and Jahr (1994)). NMDAR occupancy at the neighbor synapse rises quickly to about 25% during the first 50 ms and then, interestingly, continues to rise slowly for the remaining 150 ms of the simulation, reaching a final value of 40% (Figure IV.10B).

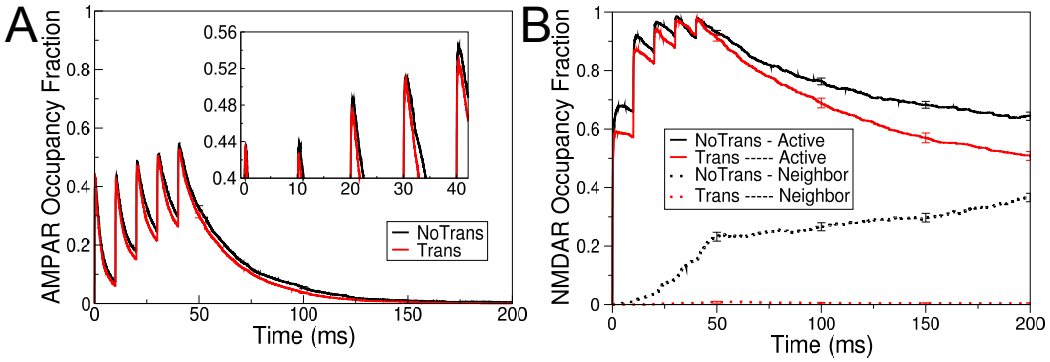

Figure IV.10 **NMDAR occupancy approaches saturation during the burst, while AMPAR occupancy does not.** NMDAR occupancy equals the sum of NMDA receptors in the C2, D, and O states, and AMPAR occupancy equals the sum of AMPA receptors in the C2, C3, C4, C5, and O states (Figure IV.3).

Transporters limit glutamate diffusion

The spatial extent of the diffusing cloud of glutamate following vesicular release was measured for each release in the 100 Hz burst. Since glutamate is irreversibly-bound by transporters in the T2 state (Figure IV.3), the location of

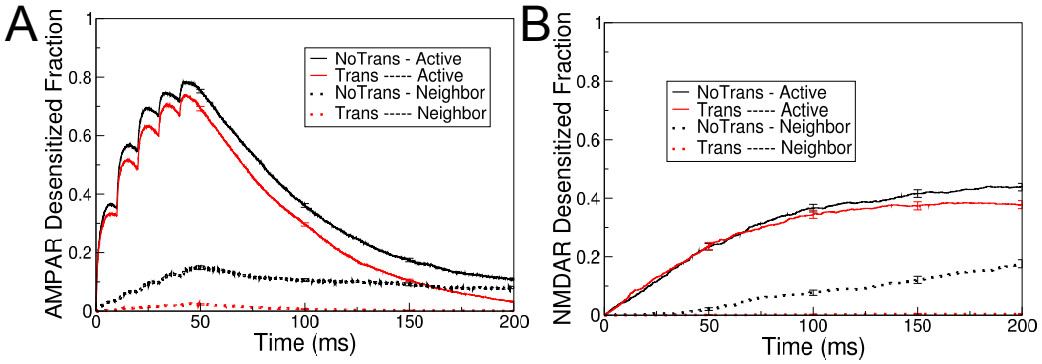

Figure IV.11 **Desensitization during burst release for (A) AMPARs and (B) NMDARs.** Increasing AMPAR desensitization is responsible for declining fraction of open AMPARs in the burst. Interestingly, AMPAR desensitization rises and decays quickly, while NMDAR rises slowly and plateaus.

a transporter in this state represents how far a particular glutamate molecule diffused before being transported. By calculating the distance from the active synapse to each transporter in the T2 state, the radial diffusion distance of each glutamate molecule can be plotted (Figure IV.12). This plot does not represent an instantaneous picture of the distribution of T2 states at a point in time, but rather represent the cumulative distribution of radial positions recorded each time a transporter transitioned into the T2 state. Thus, they are a picture of the final resting place of the vesicle contents after each release. In each subsequent release in the burst less of the glutamate is captured by transporters near the release site (Figure IV.12). Quantifying this effect, the mean radial diffusion distances for the five releases in microns are 0.937, 0.947, 0.977, 0.9899, and 0.982, respectively.

To understand why glutamate diffused farther after each subsequent release in the burst the transporter occupancy (the fraction of transporters with bound glutamate) was measured. Using the same .06 μm bins, the peak transporter occupancy in the model at each time step is plotted in Figure IV.13A, where occupancy denotes a transporter in either the T1 or T2 state. Peak trans-

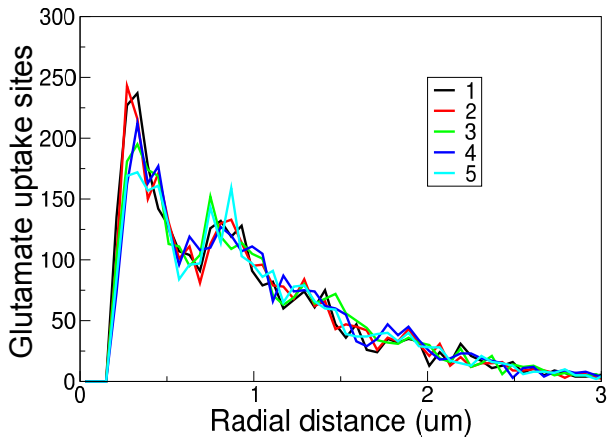

Figure IV.12 **Distribution of glutamate transporters in T2 state reveals that glutamate diffuses farther each release in burst.** Number of transporters in T2 state binned in .06 μm shells centered on release point. In each subsequent release in the burst less of the glutamate is captured by transporters near the release site. Legend indicates release event in the burst. Transporters are excluded from the region less than .192 μm from the release point by motif design.

porter occupancy occurs approximately 1 ms after each release. Figure IV.13B is a plot of the fraction of glutamate-bound transporters in each bin 1 millisecond after each release as a function of radial distance from the release point. Transporter occupancy near the release site increases with each subsequent release in the burst, but the rate of change of peak occupancy decreases during the burst. The fraction of glutamate-bound transporters during peak transporter occupancy seems to be converging to 0.7 suggesting 100 Hz burst release of 3000 glutamate particles can not saturate transporters. But the observation that transporter occupancy increases during the burst supports the idea that unavailability of unbound transporters increases glutamate diffusion distance as described earlier.

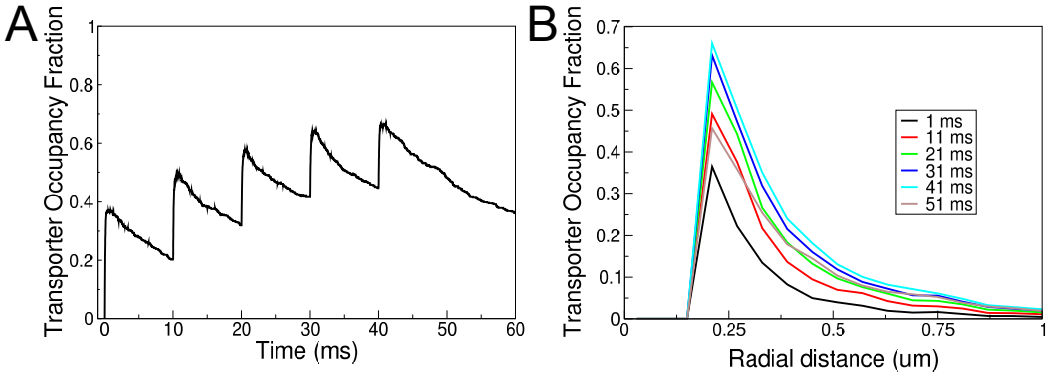

Figure IV.13 **Burst release of glutamate exhibits accumulating transporter occupancy near release site.** (A) Peak transporter occupancy in the first-generation model during the burst. (B) Radial profile of transporter occupancy 1 millisecond after each vesicular release. Legend indicates time point during simulation, i.e. plots are snapshots 1 ms after each vesicle release.

IV.3.F Motivation of Second Generation Model

Note that the first-generation model deviates in several ways from known neuropil morphological parameters, which motivated the development of an improved second-generation model as described in Section IV.4. Most importantly, this first-generation model has an elevated geometric tortuosity, a low glutamate diffusivity, a uniform distribution of glutamate transporters on glial surface, and a glial surface area density that is one order of magnitude too high. These discrepancies and others are addressed in Section IV.4.

IV.4 Second Generation 3D Model

In this section, the construction of the second-generation model is described in detail. Two variants of the improved model are used for spillover simulations with zero background glutamate. One variant has uniform surface density of glutamate transporters on glial surfaces. The other variant has variable surface

density of transporters with a higher density on glial membrane near synapses. It is shown that the mean radial diffusion distance of the glutamate cloud increased during the burst and was slightly larger in the uniform-density model compared to the variable-density model. Little difference in either transporter occupancy or spillover glutamate concentration was seen between models with uniform- and variable-density transporter populations. Glutamate transporters are not saturated with vesicle size of 3000 glutamate even after 100Hz burst in either model. Peak spillover EPSCs were 20% of EPSCs at the active synapse.

IV.4.A Design Improvements Inspired By First generation Model

A second-generation model was constructed by improving the existing first-generation model. The new model consists of an 5 x 5 x 5 μm array of redesigned motifs (Figure IV.15). The motif elements were modified to remove concavities such that the new motif consists of an irregular array of nonuniform convex boxes. The glial surface area density was reduced to 1.5 $\mu m^2/\mu m^3$ (Lehre and Danbolt, 1998) by removing transporters from some surfaces and declaring them neuronal processes. The distribution of glutamate transporters on remaining glial surface was increased to approximately 10000 per square micron and concentrated near synapses (Danbolt, 2001). A transporter surface gradient was instantiated on the model glial membranes with highest transporter density near the synapses and lower density away from the synapse. The surface density of transporters in glial membrane facing synapses (6666 per μm^2) was set approximately 50% higher than the density in glial membrane facing stem dendrites, axons, or other glial (3333 per μm^2) (Chaudhry et al., 1995) as shown in Figure IV.14.

Lowering the glial surface area density also improved the model's reflection of other experimentally measured neuropil morphological parameters. The glial contact rate, which is the percentage of synapses with astrocytic processes at the perimeter, has been observed to be 57% (Ventura and Harris, 1999), not

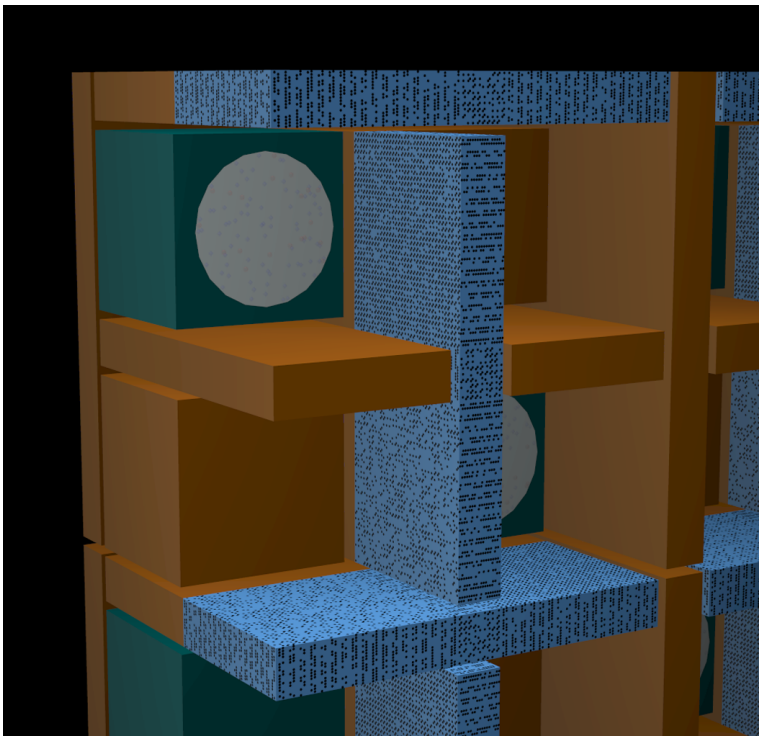

Figure IV.14 **The second-generation neuropil model featured variable transporter surface density on glial membranes.**

100% as in the first-generation model. This morphological change in the model was achieved by positioning the glial membrane opposed to the postsynaptic side of the synapse in 75% of the synapses (Lehre and Rusakov, 2002). Additionally, the percentage of the synapse perimeter opposed to glial membrane, also known as the glial coverage rate, was reduced to 50% (Ventura and Harris (1999), Danbolt (2001)). As expected the decrease in model cell volume attributed to glial processes increased the percentage of nearest-neighbor synapses (NNSs) that are separated by neuronal processes. Ventura and Harris (1999) observed that only one-third of NNSs were separated by astrocytes, unlike the first-generation model in which glial processes intervene between all NNSs. The volume fraction of axons, dendrites, boutons, spines, and glia from experimental measurements in mouse hippocampus (CA1, stratum radiatum) are 30%, 28%, 12%, 5%, 4%, respectively (Chklovskii

et al., 2002). While no distinction is made in the model between axons, dendrites, boutons, and spines, per se, the volume percentage of glia and of neurons in the second-generation model (8% and 75% respectively) are acceptable. The net effect of these structural changes was to reduce the geometric tortuosity to 1.345. The apparent diffusivity in the ECS was set to 0.53 $\mu m^2/ms$ as required to match a total tortuosity of 1.6 (see Section IV.3.A).[4]

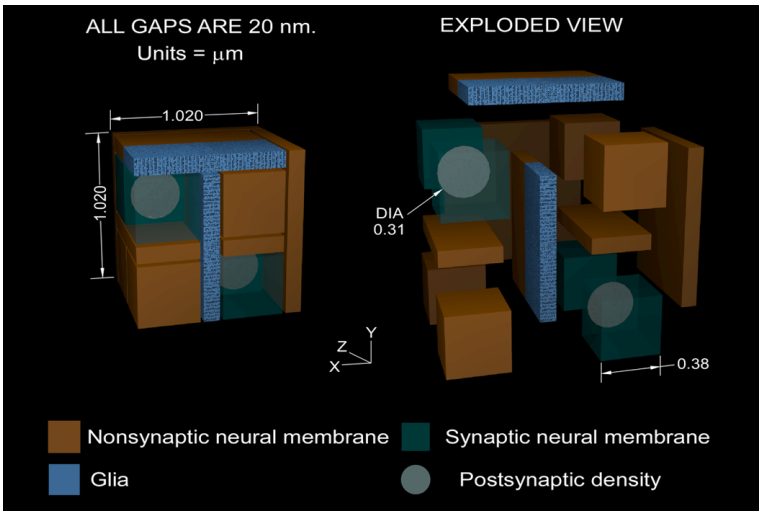

Figure IV.15 **Motif is representative of average hippocampal neuropil morphology as reported in literature.**

IV.4.B Spillover Simulations

The same release protocol used in the spillover simulations for the first-generation model were repeated with the second-generation model. Simulations of high-frequency activity consisted of a 100 Hz burst of 5 release events initiated in a single active synapse beginning at time zero. Each vesicle contained 3000 glutamate molecules and was released instantaneously in the center of the synaptic cleft

[4]The simulation time step of 1 μs when D = 0.25 $\mu m^2/ms$ had to be reduced to 0.1 μs when D = 0.53 $\mu m^2/ms$ to maintain computational accuracy. The cleft narrowing around each synapse was also removed for this reason; therefore, in the second-generation model the synaptic cleft is uniformly 20 nm in width.

(Figure IV.16). Each motif had exactly two synapses so that the synaptic density is 2 per cubic micron(Bushong et al., 2002). After each simulation time step of 0.1 μs the cleft glutamate concentration and state of each receptor in the PSD were recorded in three synapses as illustrated in Figure IV.20. Synapse 0 is the active synapse where glutamate release occurs, while glutamate spillover and crosstalk are measured in synapses 1 and 2. Data is presented for synapse 1 only, since the two synapses experienced similar amounts of spillover. Finally, the state of each transporter in the model was recorded at each time step to monitor transporter occupancy. Simulations were repeated in a model with uniform transporter density on all glial surfaces for comparison. In all simulations the glutamate diffusivity and kinetic rates are room temperature values.

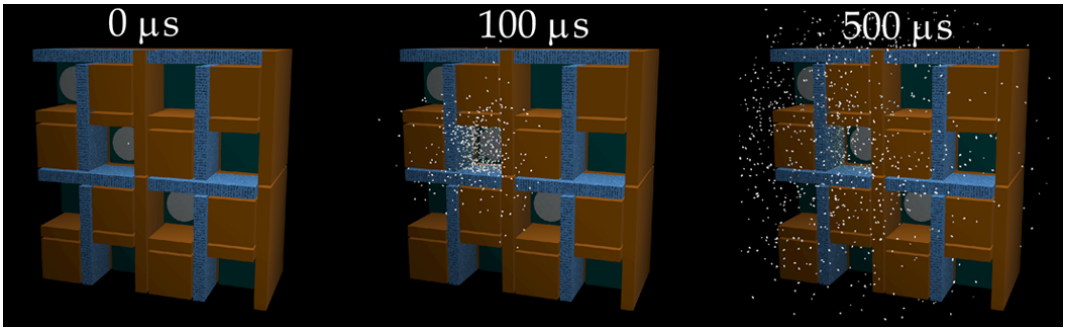

Figure IV.16 **Snapshot renderings of MCell simulation demonstrates diffusing glutamate cloud.** Release of one vesicle of glutamate from the active synapse demonstrates spillover into the extrasynaptic space. Surrounding motifs are hidden from view.

Figure IV.17A is a plot of the fraction of glutamate-bound transporters in each bin at the moment of each release as a function of radial distance from the release point for both the uniform-density (UNI) and variable-density (VAR) models. As seen before with the first-generation model, transporter occupancy near the release site increases with each subsequent release in the burst, but the rate of change of peak occupancy decreases during the burst. The fraction of glutamate-bound transporters during peak transporter occupancy after the fifth

release is less than 0.7 in both models (Figure IV.19B). It does not appear that the transporters can be saturated under these release conditions. The transporter occupancy profile looks similar for the two model types, and Figure IV.17B quantifies the difference. Less than 10% difference in transporter occupancy levels is seen between the uniform-density and variable-density model.

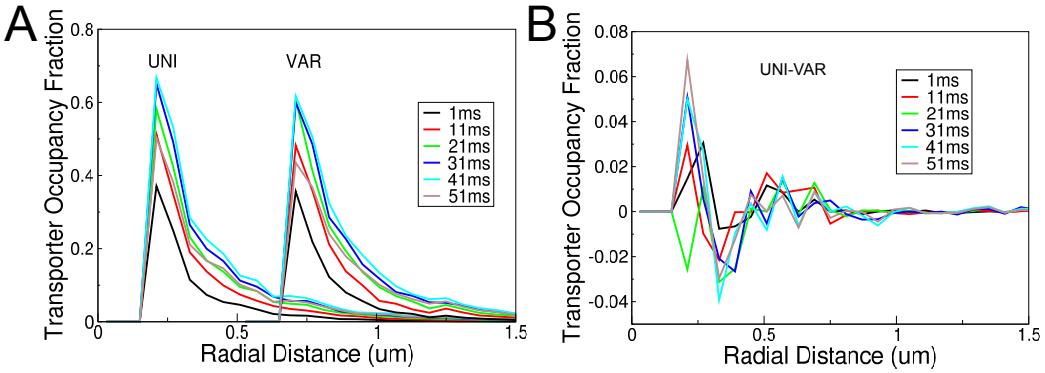

Figure IV.17 **Uniform- and Variable-density glutamate transporter models differ little in transporter occupancy.** Comparison of Transporter Occupancy in Uniform- (UNI) and Variable-Density (VAR) Models. (A) Transporter saturation never reaches 100%. Average of 3 seeds. (B) Little difference in occupancy is seen between the uniform- and variable-density models. In both plots legend indicates time point during simulation.

With such a small difference in transporter occupancy levels, we do not expect to see a big difference in spillover glutamate concentrations between the two models. Both peak glutamate concentration and time course were similar in the two models (Figure IV.18A) as was spillover activation of NMDA and AMPA receptors leading to similar resultant EPSCs (Figure IV.18B). These EPSCs are slightly larger but the same order of magnitude as those predicted from spillover in the first-generation model (Figure IV.8B); in fact, they look like an average of EPSCs at the active and neighbor synapses for the first-generation model.

Peak glutamate concentration increases slightly during the burst in Figure

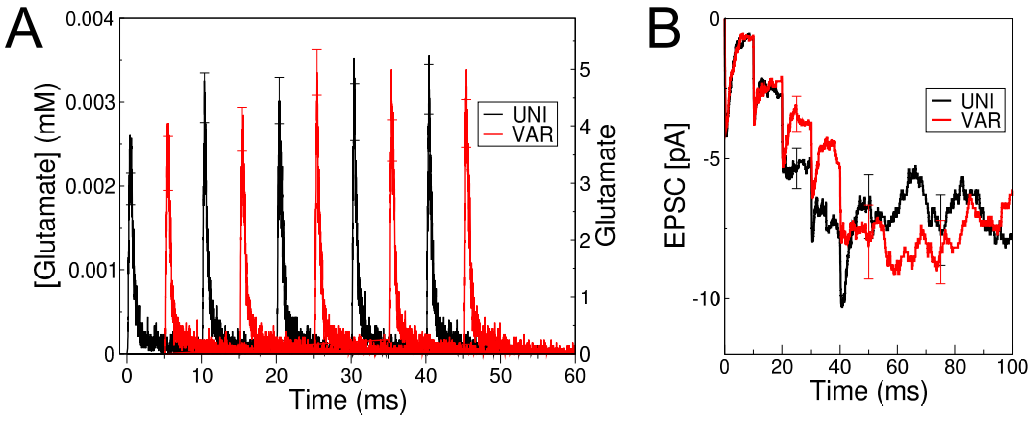

Figure IV.18 **Spillover is not significantly different in Uniform- and Variable-density glutamate transporter models.** (A) Plot of glutamate concentration measured at synapse 1 (see Figure IV.20) in both the uniform-density (UNI) model and variable-density (VAR) model (time-shifted by 5 ms). Glutamate spillover is small in both models. (B) Spillover EPSCs at the neighbor synapse 1 are similar in the two models and similar to the spillover EPSC seen in the first-generation model (Figure IV.8B).

IV.18A. This is consistent with an increase in the mean radial diffusion distance of each vesicle cloud during the burst. As expected from previous results in the first-generation mode, the mean radial diffusion distance is correlated with the peak transporter occupancy during vesicle release (Figure IV.19B). Figure IV.19A illustrates three aspects of glutamate diffusion in the two models. First, glutamate diffuses farther in the uniform-density model than in the variable-density model. Second, the diffusion distance increases during the burst for both models. Third, these trends are probably insignificant, since the differences in diffusion distance are small, i.e. less than 5%.

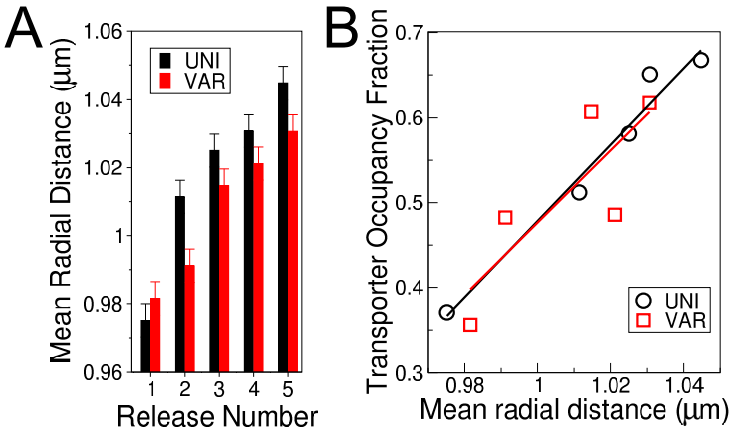

Figure IV.19 **Uniform- and Variable-density glutamate transporter models differ little in mean radial glutamate diffusion distance.** (A) Mean radial diffusion distance increases during burst and is slightly larger, on average, in the uniform-density (UNI) model than in the variable-density (VAR) model. (B) Peak transporter occupancy is positively correlated with mean radial glutamate diffusion distance. Best fit lines are shown.

IV.5 Acknowledgements

This chapter is, in part, in preparation for submission for publication. The dissertation author was primary investigator and author and the co-authors, Frederic D. Broccard, Thomas M. Bartol and Terrence J. Sejnowski, supervised and directed the research which forms the basis of this chapter.

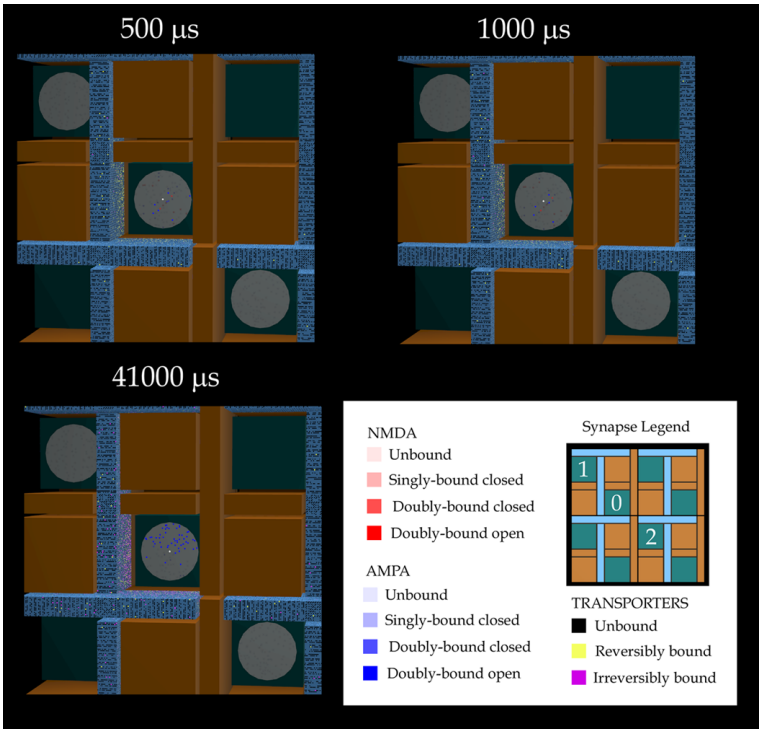

Figure IV.20 **Rendering of transporter occupancy and NMDAR crosstalk in variable-density neuropil model.** Peak glutamate spillover and transporter occupancy occur 500 μs and 1000 μs after vesicle release, respectively. The active synapse is labeled '0' in the legend and two neighbor synapses are also indicated by '1' and '2'. Note synapses '0' and '1' are separated by more glial membrane than are synapses '0' and '2'. Also note the large number of bound transporters surrounding the active synapse. Transporter occupancy is high after fifth release (41000 μs).

A

Supplementary Material

A.1 Steady-State Analysis of Glutamate Transporter and Receptor Kinetic Models

In this section we analyze the effect of a steady, nonzero background glutamate concentration on the distribution of kinetic model states for the glutamate transporter and receptors. The results are compared to Monte Carlo simulation results in Table IV.4.

A.1.A Steady-State Glutamate Transporter Distribution and Transport Rate

The transport of glutamate from the extracellular space into glial cells is modeled with the following reaction equation, where GLU represents extracellular glutamate and GLU' is glutamate that has been transported into glial cells.

$$T0 + GLU \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} T1 \xrightarrow{k_2} T2 \xrightarrow{k_3} T0 + GLU' \tag{A.1}$$

The unbound transporter, T0, binds one molecule of glutamate, GLU, to become the T1 conformation. The T1 conformation transitions to T2 conformation which destroys the bound glutamate molecule and resets to T0 state. Note that conser-

vation of species applies as follows.

$$[T0] + [T1] + [T2] = [T_{total}] \tag{A.2}$$

$[T_{total}]$ (the concentration of glutamate transporters in the model) is constant. The volume concentration of transporter in each state is calculated by dividing the number of transporters in that state by the total extracellular volume (V_{extra}).

$$[T0] = \frac{N_{T0}}{V_{extra}},\ [T1] = \frac{N_{T1}}{V_{extra}},\ [T2] = \frac{N_{T2}}{V_{extra}},\ [T_{total}] = \frac{N_{total}}{V_{extra}} \tag{A.3}$$

At steady-state conditions the rate of synthesis of T1 is equal to the rate of destruction (Equation A.4). The same statement is true for T2 (Equation A.5).

$$k_1[T0][GLU] = k_{-1}[T1] + k_2[T1] \tag{A.4}$$

$$k_2[T1] = k_3[T2] \tag{A.5}$$

By combining Equations A.2, A.4, and A.5 the concentration of T1 can be written as a function of $[GLU]$ and $[T_{total}]$.

$$[T1] = \frac{\left(\frac{k_3}{k_2+k_3}\right)[GLU][T_{total}]}{\left(\frac{k_3}{k_2+k_3}\right)\left(\frac{k_{-1}+k_2}{k_1}\right) + [GLU]} \tag{A.6}$$

The rate of glutamate transport, R, at steady-state is

$$R = k_2[T1] = k_3[T2]. \tag{A.7}$$

Combine equations A.6 and A.7 to yield a new expression for R.

$$R = \frac{\left(\frac{k_2k_3}{k_2+k_3}\right)[GLU][T_{total}]}{k_J + [GLU]} \tag{A.8}$$

where

$$k_J = \left(\frac{k_3}{k_2+k_3}\right)\left(\frac{k_{-1}+k_2}{k_1}\right) \tag{A.9}$$

Let $R_{max} = R$ when $[GLU] \gg k_J$.

$$R_{max} = \left(\frac{k_2k_3}{k_2+k_3}\right)[T_{total}] \tag{A.10}$$

The final expression for the steady-state rate of glutamate transport for a given density of transporters as a function of glutamate concentration becomes

$$R = \frac{R_{max}[GLU]}{k_J + [GLU]} \tag{A.11}$$

The distribution of glutamate transporters at steady-state expressed as a function of transport rate becomes

$$\frac{T0}{T_{total}} = 1 - \frac{T1}{T_{total}} - \frac{T2}{T_{total}} \tag{A.12}$$

$$\frac{T1}{T_{total}} = \frac{R}{k_2 T_{total}} \tag{A.13}$$

$$\frac{T2}{T_{total}} = \frac{R}{k_3 T_{total}} \tag{A.14}$$

A.1.B Steady-state AMPA and NMDA Receptor Distribution

Here we derive the steady-state distribution of AMPARs and NMDARs as a function of glutamate concentration for the kinetic models given in Figure IV.3. Six unique steady-state rate equations can be written based on this diagram. At steady-state the rate of synthesis of each AMPA conformation is equal to its rate of destruction. We write this equality for C1, C2, C3, C4, C5, and O.

$$k_1[C0][GLU] + k_{-4}[C2] + k_{-2}[C3] = (k_{-1} + k_4[GLU] + k_2)[C1] \tag{A.15}$$

$$k_4[C1][GLU] + k_{-6}[C4] + k_{-7}[O] = (k_7 + k_{-4} + k_6)[C2] \tag{A.16}$$

$$k_2[C1] + k_{-5}[C4] = (k_{-2} + k_5[GLU])[C3] \tag{A.17}$$

$$k_5[C3][GLU] + k_6[C2] + k_{-8}[C5] = (k_{-5} + k_{-6} + k_8)[C4] \tag{A.18}$$

$$k_9[O] + k_8[C4] = (k_{-8} + k_{-9})[C5] \tag{A.19}$$

$$k_{-9}[C5] + k_7[C2] = (k_{-7} + k_9)[O] \tag{A.20}$$

The addition of a conservation of species expression,

$$[C0] + [C1] + [C2] + [C3] + [C4] + [C5] + [O] = [AMPA_{total}], \tag{A.21}$$

where $[AMPA_{total}]$ (the concentration of AMPARs in the model) is a constant, represents a seventh equation defining the distribution of AMPA receptors for a given steady-state glutamate concentration, $[GLU]$, and binding rate set, k_i. Since the unknown concentrations number seven, the system of equations can be solved. The system of equations can be written in the form

$$Ax = B. \tag{A.22}$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ k_1[GLU] & \alpha & k_{-4} & k_{-2} & 0 & 0 & 0 \\ 0 & k_4[GLU] & \beta & 0 & k_{-6} & 0 & k_{-7} \\ 0 & k_2 & 0 & \gamma & k_{-5} & 0 & 0 \\ 0 & 0 & k_6 & k_5[GLU] & \delta & k_{-8} & 0 \\ 0 & 0 & 0 & 0 & k_8 & -(k_{-8}+k_{-9}) & k_9 \\ 0 & 0 & k_7 & 0 & 0 & k_{-9} & -(k_{-7}+k_9) \end{bmatrix} \tag{A.23}$$

$$\begin{aligned} \alpha &= -(k_{-1}+k_2+k_4[GLU]) \quad \beta = -(k_{-4}+k_6+k_7) \\ \gamma &= -(k_{-2}+k_5[GLU]) \quad \delta = -(k_{-5}+k_{-6}+k_8) \end{aligned} \tag{A.24}$$

$$B = \begin{bmatrix} [AMPA_{total}] \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad x = \begin{bmatrix} [\mathrm{C0}] \\ [\mathrm{C1}] \\ [\mathrm{C2}] \\ [\mathrm{C3}] \\ [\mathrm{C4}] \\ [\mathrm{C5}] \\ [\mathrm{O}] \end{bmatrix}. \tag{A.25}$$

The solution to this problem involves a simple matrix inversion

$$x = A^{-1}B. \tag{A.26}$$

In a similar manner, we consider the NMDA receptor kinetic model also given in Figure IV.3. Note that the state transition from O to C1 in the NMDA receptor model is erroneous and should be removed. This error was discovered after the simulations were run and was corrected before the second-generation model simulations. Comparison of analytical predictions of NMDA receptor state distributions with 0.5 μM background concentration predicts that the error will overestimate the number of receptors in the unbound state (C0) by 5% and underestimate the number in the desensitized state (D) by 5%. Four unique steady-state rate equations can be written based on this model. At steady-state the rate of synthesis of each NMDA conformation is equal to its rate of destruction. We write this equality for C1, C2, D, and O.

$$k_1[C0][GLU] + k_4[O] + k_{-2}[C2] = k_{-1}[C1] + k_2[C1][GLU] \quad \text{(A.27)}$$

$$k_2[C1][GLU] + k_{-3}[D] + k_{-5}[O] = (k_{-2} + k_3 + k_5)[C2] \quad \text{(A.28)}$$

$$k_3[C2] = k_{-3}[D] \quad \text{(A.29)}$$

$$k_5[C2] = (k_{-5} + k_4)[O] \quad \text{(A.30)}$$

The addition of a conservation of species expression,

$$[C0] + [C1] + [C2] + [D] + [O] = [NMDA_{total}], \quad \text{(A.31)}$$

where $[NMDA_{total}]$ (the concentration of NMDARs in the model) is a constant, represents a fifth equation defining the distribution of NMDA receptors for a given steady-state glutamate concentration, $[GLU]$, and binding rate set, k_i. Since the unknown concentrations number five, the system of equations can be solved. The system of equations can also be written in the same form as Equation A.22 where

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ k_1[GLU] & -k_2[GLU]-k_{-1} & k_{-2} & 0 & k_4 \\ 0 & k_2[GLU] & -(k_{-2}+k_3+k_5) & k_{-3} & k_{-5} \\ 0 & 0 & k_3 & -k_{-3} & 0 \\ 0 & 0 & k_5 & 0 & -(k_{-5}+k_4) \end{bmatrix} \quad \text{(A.32)}$$

$$B = \begin{bmatrix} [NMDA_{total}] \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad x = \begin{bmatrix} [\mathrm{C0}] \\ [\mathrm{C1}] \\ [\mathrm{C2}] \\ [\mathrm{D}] \\ [\mathrm{O}] \end{bmatrix}. \tag{A.33}$$

As before, the solution to this problem involves a simple matrix inversion as shown in Equation A.26.

A.2 Tortuosity

The effective diffusivity D_e of glutamate in the ECS of the neuropil is lower than that in aqueous solution D for two reasons: (1) glutamate molecules interact with macromolecules and the cell membrane, and (2) the convoluted tangle of neuronal and glial processes increases the diffusion distance between any two neuropil locations (Rusakov and Kullmann, 1998a). The first effect is called the viscous tortuosity, λ_v, and the latter the geometric tortuosity, λ_g.[1] The diffusion of glutamate in the neuropil is impeded by the tortuous path a molecule must take to circumnavigate the multitude of glial and neuronal processes in its path. As a result of this complex geometry, the effective diffusivity D_e is lowered. Viscous tortuosity captures diffusional hindrance in the ECS not accounted for by the geometrical tortuosity, while in fact neither the magnitude nor the molecular mechanism of ECS viscosity is known.

The effective diffusion constant is calculated as

$$D_e = \frac{D}{\lambda_T^2} = \frac{D}{(\lambda_v \lambda_g)^2}$$

where λ_T is the total tortuosity. Defining the apparent diffusion D_a as the viscous

[1] Both the viscous and geometric tortuosity of open space is identically equal to one.

component of diffusivity, then by definition

$$D_a = \frac{D}{\lambda_v^2},$$

and therefore

$$D_a = \frac{D\lambda_g^2}{\lambda_T^2}.$$

This formulation is used to calculate an apparent diffusivity that yields a desired effective diffusivity given a specified model geometry.[2] In other words, given a target total tortuosity and measured geometric tortuosity, what apparent diffusivity is required? Note the diffusivity D of glutamate in water at 25 °C was reported as approximately 0.75 $\mu m^2/ms$ (Longsworth, 1953).

The effective diffusivity of glutamate in the cerebellar glomerulus has been experimentally determined to be approximately one-third that of free aqueous value, i.e. approximately 0.25 $\mu m^2/ms$ (Nielsen et al., 2004) Interestingly, this value matches that found for diffusion of acetylcholine in the frog neuromuscular junction (Stiles et al., 1996).

The total tortuosity of CA1 region of hippocampal slices in normoxic (normal oxygen) conditions has consistently been reported as being in the range 1.5 - 1.7 (Nicholson and Phillips (1981), McBain et al. (1990), Sykova (1997), Mazel et al. (1998)) with recent reports assuming a value of 1.6 (Hrabetová and Nicholson (2000), Hrabetová and Nicholson (2004)). Investigation into the nature of geometrical tortuosity revealed that the diffusional hindrance of regular arrays of convex polyhedra is solely a function of the extracellular volume fraction (EVF, α) and independent of polyhedral shape (Tao and Nicholson, 2004): $\lambda_g = \sqrt{(3-\alpha)/2}$. Assuming a volume fraction of 0.2, as is consistent with recent reports for hippocampus stratum radiatum (Hrabetová and Nicholson, 2004), a regular array of convex polyhedra would have a geometric tortuosity of approximately 1.18. What aspect of neuropil increases the tortuosity above this value? In ischemic tissue, it has been argued that dead-end pores contribute to diffusional

[2]The diffusion constant used in MCell is the apparent diffusivity.

hindrance. Experimental results show that adding 70 kDa dextran to ischemic tissue slices lowers the EVF and decreases the measured tortuosity (Hrabetová and Nicholson, 2000). Presumably, during ischemia swelling cells form blockages creating dead-end pores; adding dextran blocks these dead-end pores, causing tortuosity and EVF to drop (Hrabetová and Nicholson, 2004). However, no change was seen in tortuosity or EVF when dextran was added to normoxic hippocampal slices. Furthermore, no direct evidence exists for the presence of dead-end pores in ischemic tissue, not-to-mention normal neuropil.

A.3 First-Generation Model Detail Drawings

The spillover simulations described in Chapter IV utilized a simplified model of neural tissue. The model was constructed as an array of canonical motifs (Figure A.1 and Figure IV.2) designed to match experimentally measured values of the hippocampal neuropil microstructure.

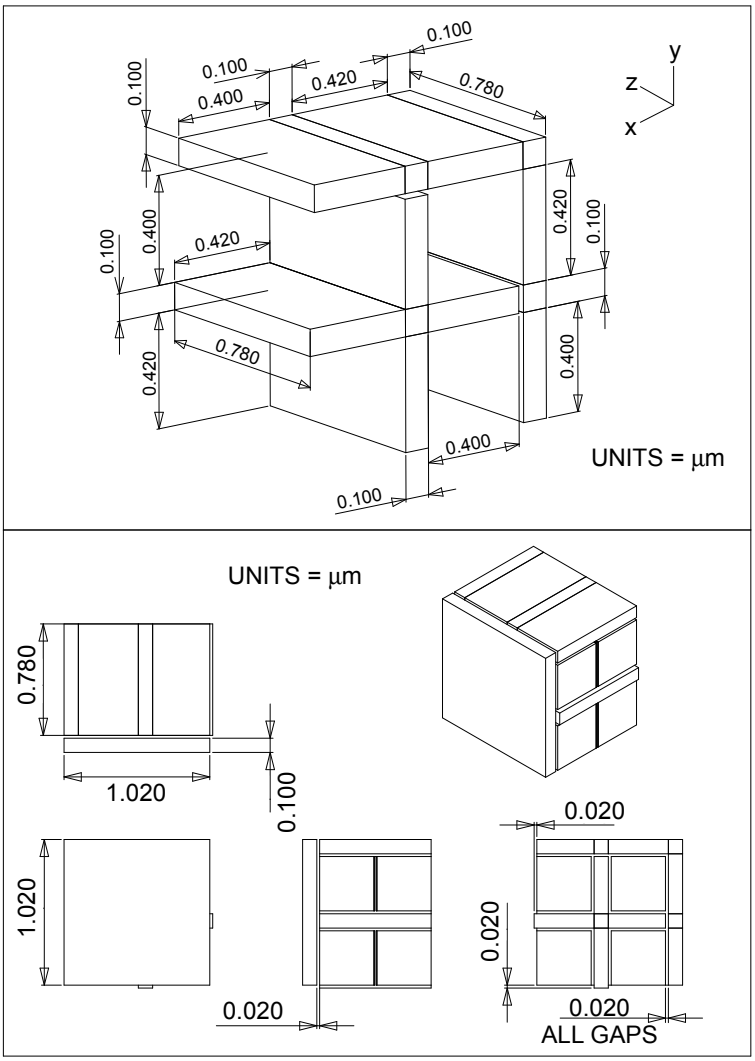

Figure A.1 Schematics of First Generation Model

B

Streamlining the Reconstruction Pipeline

In Section II.2 we described changes to CONTOUR_TILER to further automate the reconstruction. The goal was to accelerate the reconstruction process by adapting CONTOUR_TILER to our specific application - the reconstruction of rat hippocampal neuropil. Here we review a complete list of all thirty-seven software steps in the CA1 hippocampus reconstruction pipeline with a focus on simplifying the work flow. Upon review many improvements can be made to simplify and accelerate this operational sequence. We have aggregated the modifications into three groups each building on the previous group involving progressively more aggressive and invasive changes to the pipeline.

- **Pipeline Improvement Phase 1** B-spline sampling is improved so CONTOUR_TILER does not create sharp, thin protrusions. For simple visualization Blender replaces DReAMM and some manual mesh manipulation. Various file conversions are ameliorated.

- **Pipeline Improvement Phase 2** Changes described in Phase 1 plus CONTOUR_TILER is hacked. Note that the iteration described in this example pipeline was not feasible in original reconstruction nor would be in Phase

1 pipeline because of labor intensive manual steps that would be inside the loop, e.g. use of meshstitch, mesh_separate, and meshclip.

- **Pipeline Improvement Phase 3** Improvements described in Phase 2 plus RECONSTRUCT3D is hacked. This pipeline would be the same as Phase 2 but much faster. Both involve iteration from RECONSTRUCT3D to Blender multiple times, making changes to the contour metadata. Having RECONSTRUCT3D hacked could greatly accelerate the process, since it would facilitate the tagging of metadata to contours. Alternatively, further streamline the reconstruction process by building segmentation tools into Blender, possibly incorporating C++ libraries into Blender's python interface with swig wrappers.

In all example pipelines the first step represents segmentation of the EM images and annotation of the contours (Section II.1.B). The second step is a combination of contour splining, sampling, and mesh surface generation from the contours (Sections II.1.C and II.1.D). Recovery of the extracellular space (Section II.1.F) is included as a MESHMORPH step. The last step is always to perform biochemical simulations with MCELL. All other steps were required to improve the quality of the surface meshes (Section II.1.E). The format of each item is 'software program - description'.

B.1 Original CA1 Reconstruction Pipeline

- RECONSTRUCT3D - create and edit contours
- reconstruct2contourtiler - spline and sample contours and create surface meshes with CONTOUR_TILER
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX

- mesh_stitch - stitch spine meshes to dendrite shaft mesh and stitch astrocyte mesh pieces together
- remove_duplicate_vertices - identify vertices with identical locations and remove one
- MESHALYZER - check integrity of meshes
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- FILTERMESH - remesh to remove shark fins
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- mesh_separate - remove bubbles
- mesh_merge - keep nonbubble meshes
- mesh2irit - convert mesh to IRIT format
- subtract_mesh - create pockets using IRIT CSG
- irit2mesh - stage 1 mesh recovery
- meshheal - stage 2 mesh recovery
- MESHALYZER - did anything go horribly wrong?
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- meshclip - manually remove faces with high aspect ratios
- mesh_renumber - renumber after manual step

- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- mesh2stl - prepare mesh data for netgen
- netgen - remesh data
- netgen2mesh - recover mesh format
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- MESHALYZER - measure mesh statistics
- meshmorph - recover extracellular space
- mesh2dx/mesh2groupdx - convert mesh format to dx format
- DReAMM - view meshes in DX
- Blender - create MCell regions from meshes
- mesh2mcell - convert single mesh to mdl format for mcell sim to generate group DX
- MCell - run diffusion simulation

B.2 Pipeline Improvement Phase 1

B-spline sampling is improved so CONTOUR_TILER does not create shark fins. Blender replaces DReAMM and some manual mesh manipulation. Various file conversions are ameliorated

- RECONSTRUCT3D - create and edit contours

- reconstruct1contourtiler - spline and sample contours and create surface meshes with CONTOUR_TILER
- Blender - inspect meshes
- mesh_stitch - stitch spine meshes to dendrite shaft mesh and stitch astrocyte mesh pieces together
- remove_duplicate_vertices - identify vertices with identical locations and remove one
- MESHALYZER - check integrity of meshes
- Blender - inspect meshes
- mesh_separate - remove bubbles
- mesh_merge - keep nonbubble meshes
- mesh2irit - convert mesh to IRIT format
- subtract_mesh - create pockets using IRIT CSG
- irit2mesh - stage 1 mesh recovery
- meshheal - stage 2 mesh recovery
- MESHALYZER - did anything go horribly wrong?
- Blender - inspect meshes
- meshclip - manually remove faces with high aspect ratios
- mesh_renumber - renumber after manual step
- Blender - inspect meshes
- mesh2stl - prepare mesh data for netgen

- netgen - remesh data
- netgen2mesh - recover mesh format
- Blender - inspect meshes
- MESHALYZER - measure mesh statistics
- meshmorph - recover extracellular space
- Blender - inspect meshes
- Blender - create MCell regions from meshes
- mesh2mcell - convert single mesh to mdl format for mcell sim to generate group DX
- MCell - run diffusion simulation

B.3 Pipeline Improvement Phase 2

Improvements described in Phase 1 plus CONTOUR_TILER is hacked. Note that the iteration described in this example pipeline is not feasible in previous example pipelines because of labor-intensive manual steps that would be inside the loop, e.g. use of mesh-stitch, mesh_separate, and meshclip.

- RECONSTRUCT3D - create and edit contours
- reconstruct1contourtiler - spline and sample contours and create surface meshes with CONTOUR_TILER
- MESHALYZER - check integrity of meshes
- Blender - inspect meshes
- Iterate first four steps until all requisite contour metadata is added yielding error-free meshes

- netgen - remesh data
- Blender - inspect meshes
- meshmorph - recover extracellular space
- Blender - inspect meshes
- MESHALYZER - measure mesh statistics
- Blender - create MCell regions from meshes
- MCell - run diffusion simulation

B.4 Pipeline Improvement Phase 3

Improvements described in Phase 2 plus RECONSTRUCT3D is hacked. This pipeline would be the same as Phase 2 just much faster. Both involve iteration from RECONSTRUCT3D to Blender multiple times, making changes to the contour metadata. Having RECONSTRUCT3D hacked could greatly accelerate the process, since it would facilitate the tagging of metadata to contours.

C

Review of Neurotransmitter Spillover

Experimental investigations of glutamate spillover in hippocampal slices offer three lines of evidence for synaptic crosstalk. First, a lower coefficient of variation for NMDAR-mediated EPSCS compared to AMPAR-mediated currents suggests a higher quantal content is sensed by NMDARs than by AMPARs. The higher quantal content is interpreted as low-concentration glutamate from spillover detected by high-affinity NMDARs. Second, the slow decay of the NMDAR EPSC is taken as evidence for a prolonged low concentration of glutamate since the EPSC time course is modulated by weak competitive antagonists to NMDARs. Finally, glutamate uncaging experiments have shown that no NMDAR EPSC is recorded by glutamate release beyond 1 μM from dendritic spines implying evidence for glutamate diffusion in the slice and an upper bound on spillover distance.

The consensus so far in the published results from computational models is that spillover is minimal. Modeling investigations have consistently found that significant spillover following a single vesicle release is unlikely. The glutamate concentration is minuscule 1 μm away from site of single vesicle release (Wahl et al., 1996). Spillover from one vesicle activated zero AMPARs (0.8% of max open probability) and 2 NMDARS (3.9% of max open probability) 0.5 μm away from

release site even with no glutamate transporters (Franks et al., 2002). Kullmann et al. (1999) calculated that 15% of NMDARs will open 0.5 μm away from single release location. Finally, Franks et al. (2003) found that receptors were essentially unaffected by release events greater than 300 nm away. The high density of transporters and especially the neuropil tortuosity inhibit spillover, such that synapses operate independently, except perhaps if significant multivesicular release occurs.

C.1 Experimental Evidence for Spillover

Recent in vitro experimental results in hippocampus suggest that at CA1 pyramidal cells the released glutamate is not confined to the synaptic cleft, but instead diffuses out of the synapse into the extrasynaptic, extracellular space (ECS), an event termed "spillover". In the densely packed neuropil it is thought that glutamate can spillover and activate postsynaptic receptors on neighboring synapses, deemed "crosstalk". The presence of neuronal and glial transporters were thought to prevent crosstalk by rapidly binding glutamate, however, this assumption and the independence of individual synapses as signal channels has been called into question.

Strictly speaking there has been no direct evidence put forth by the scientific community for the existence or absence of crosstalk between Schaffer collateral synapses in hippocampus *in vivo*. And while no direct in vivo observation of extrasynaptic glutamate spillover in hippocampus has been put forth, it is generally accepted that glutamate spills out of the synapse onto astrocytes where it is rapidly taken up by electrogenic membrane-bound glutamate transporters as recorded by transporter currents (Diamond, 2001). Several *in vitro* studies have investigated the sensitivity of NMDA receptors (NMDARs) to spillover and the role glutamate transporters play in preventing crosstalk.

One line of evidence for in vitro observations of glutamate spillover is based on differences in the quantal content (number of vesicular release events)

sensed by NMDARs and AMPARs. Assuming a binomial model of transmitter release at each synapse, a smaller coefficient of variation of NMDAR-mediated component of EPSC compared to the AMPAR-mediated component implies a larger quantal content is being sensed by the NMDA receptors than the AMPA receptors (Appendix C.4).

If the receptors are colocalized at synapses, then NMDARs might be detecting vesicular release events by spillover because of the higher NMDAR binding affinity to glutamate. Asztely et al. (1997) measured the trial-to-trial variability of NMDAR-mediated and AMPAR-mediated population EPSCs elicited in CA1 cells by repeated stimulation of Schaffer collaterals in guinea pig hippocampal slices. They found that adding the glutamate uptake blocker dihydrokainate (DHK) to the bath increased the ratio of the NMDAR- to AMPAR-sensed quantal content significantly. Pankratov and Krishtal (2003) found similar results in acute rat hippocampal slices at room temperature when 4-aminopyridine (glutamate release enhancer) was added to the bath. Both groups interpret their results as evidence for glutamate spillover and synaptic crosstalk under conditions of high extracellular glutamate concentration. An alternative interpretation, which does not rely on intersynaptic glutamate diffusion, is that AMPA receptors are non-functional or absent at a proportion of synapses ('silent synapse'), although they might be recruited by postsynaptic induction of long-term potentiation (Liao et al., 1995). Although both phenomena may coexist, one observation that is difficult to reconcile with the 'latent AMPA receptor cluster' hypothesis is that, if the experiments are repeated at physiological temperature (as opposed to room temperature), the discrepancy in behavior of the AMPA and NMDA receptor- mediated components is much smaller (Asztely et al., 1997). The authors hypothesize that enhanced glutamate clearance at the physiological temperature reduced the extent of spillover and cite the Q_{10} of 3 (Wadiche et al., 1995) for the glutamate transporter as support for their argument. In support of such a role for glutamate uptake, pharmacological blockade of transporters partially reverses the effect of raising the recording

temperature (Asztely et al., 1997). Intersynaptic crosstalk in vitro may thus be, at least in part, an artifact of the subnormal recording temperature at which most experiments are carried out.

A second line of evidence for in vitro observations of glutamate spillover concern prolongations of the NMDAR-mediated EPSC time course while at the same time the AMPAR-mediated EPSC is unaffected. Based on investigations with a low-affinity competitive antagonist of NMDAR in rat hippocampal slices at room and physiological temperature, spillover between Schaffer collateral synapses after single stimuli has been observed in the absence of glutamate transporter blocker (Diamond, 2001). The competitive antagonist D-AA preferentially blocked slower components of the NMDAR EPSC in pyramidal cells, suggesting that the NMDARs underlying those components were activated by lower glutamate concentration than those contributing to the peak of the EPSC. Reducing extrasynaptic glutamate transport with TBOA and voltage-clamp of the postsynaptic neuron at a positive potential enhanced activation of those receptors encountering the lowest glutamate concentration, suggesting that they were activated by glutamate spillover. Arnth-Jensen et al. (2002) report prolongation of the NMDAR EPSC under in vitro conditions of high glutamate concentration. The time course of the NMDA EPSC was significantly delayed by blocking glutamate uptake by adding TBOA to the bath and either (1) increasing the number of active release sites and number of vesicles released at single synapse by increasing Ca^{2+}/Mg^{2+} ratio or (2) increasing the spatial density of activated synapses by increasing stimulation intensity. Interestingly, in organotypic slice cultures TBOA had little effect on NMDAR EPSC time course following a single spike. Finally, the authors report prolongation of the NMDAR EPSC even in the absence of uptake inhibitor following low-intensity, high-frequency stimulation (100Hz burst of 5-10 releases every 6 seconds). Similar experiments using the transmitter release enhancer 4-AP (Lozovaya et al., 1999), D-AA (Grebenyuk et al., 2004), and low-affinity NMDAR/AMPAR antagonist DGG (Biró and Nusser, 2005) have yielded similar results and conclu-

sions. In all cases experimental manipulation to increase extracellular glutamate concentration leads to a delay in the NMDA EPSC time course decay which can be reversed by a low-affinity NMDAR antagonist, suggesting that the prolongation of the EPSC is due to sustained low glutamate concentration.

Experiments using burst stimulation to enhance glutamate release did not use glutamate transporter blockers and found that spillover effects on NMDA receptors are enhanced by increasing stimulus intensity and frequency (Lozovaya et al. (1999), Grebenyuk et al. (2004), Arnth-Jensen et al. (2002), but see Diamond (2001)). Note that the potential impact of glutamate spillover plateaus as measured by charge transfer into the postsynaptic neuron via NMDAR activation which saturates after 7 releases in a 200Hz burst (Lozovaya et al., 1999). What is minimum stimulus intensity that results in spillover in vitro? This is an interesting question without an obvious answer. Diamond (2001) found that a single low-intensity pulse (30 pA peak EPSC amplitude) of the Schaffer Collaterals without glutamate transporter blocker in the bath was sufficient to cause spillover at CA1 synapses as evidenced by a decrease in the half decay time by the low-affinity competitive NMDA antagonist D-AA. Because the EPSC peak amplitude was not significantly affected, presumably D-AA is outcompeting the glutamate at binding to the the NMDARS during periods of low glutamate concentration caused by spillover. However, Arnth-Jensen et al. (2002) evoked NMDAR EPSCs of similar magnitude in CA1 pyramidal neurons in organotypic slice culture and found that blocking glutamate transporters with TBOA had no significant effect on the time course of the EPSC. If the NMDARs are experiencing prolonged, low concentration glutamate transients and glutamate transporters are effective at preventing spillover, where is the glutamate originating and why doesn't blocking transporters have a noticeable effect?

Are extrasynaptic NMDARs involved? Recent work published by Lozovaya et al. (2004) has focused on extrasynaptic NMDARs with a different subunit composition than synaptic NMDARs. They propose that the increase in the

transmitter release by a short train of 200Hz stimulation results in the involvement of predominantly extrasynaptic 'slow' NR2B and 'superslow' NR2D receptors in the NMDA EPSC. Pharmacologically blocking these receptors accelerates the decay of the NMDA EPSC. However, immunohistochemical localization of NR2D-containing receptors is restricted to the oriens layer of CA1 and stratum lucidum of CA3 and no NR2D-immunoreactivity was seen in stratum radiatum of CA1(Thompson et al., 2002). However, functional extrasynaptic NMDA receptors have been reported and their significance needs to be investigated further (Stocca and Vicini (1998), Rumbaugh and Vicini (1999), Tovar and Westbrook (1999), Sattler et al. (2000), Hardingham et al. (2002), Li et al. (2002), Aarts et al. (2002)).

How far does glutamate diffuse after synaptic vesicular release? The spatial spread of activation of NMDARs in CA1 has been estimated in vitro by 2-photon uncaging of MNI-glutamate at several points along the line connecting the distal tips of two neighboring spines separated by a distance of 2 μm (Noguchi et al., 2005). The uncaging protocol was expected to induce currents with amplitudes about two to three times as large as those of miniature EPSCs. Whole-cell patch-clamp recording and two-photon calcium imaging revealed that NMDAR activation was absent when glutamate was uncaged more than 1 μm from a spine head under the experimental conditions implying feasibility of neurotransmitter spillover an upper bound on glutamate diffusion extent in vitro. Additionally, it was found that photolysis of MNI-glutamate at the dendritic shaft induced only small increases in intracellular calcium concentration compared with those that were apparent with uncaging at the spine tip. This indicates that NMDARs were present in relatively small numbers in the dendritic shaft compared with the spine head.

C.2 Previous Models of Spillover

Published papers investigating extrasynaptic glutamate spillover in hippocampus by computer simulation have predominantly utilized finite element analysis (FEA) of differential equation models of glutamate diffusion. Hippocampal neuropil is modeled as a central synapse surrounded by an isotropic porous medium described by an extracellular volume fraction, α, and total tortuosity, λ[1]. Each element of the model outside the synapse has an assumed concentration of glutamate transporter and a calculated concentration of glutamate. The synaptic cleft is modeled as a thin disk of specified diameter, and glutamate release from the synapse center occurs either instantaneously or at a controlled release rate. Model results consistently predict that AMPAR activation by glutamate spillover is negligible due to the receptor's low glutamate affinity and rapid desensitization. The focus for spillover investigations has therefore been on the potential role NMDARs play in synaptic crosstalk. Computational models of glutamate release in the hippocampal neuropil have served to characterize how model parameters affect NMDAR activation by glutamate spillover.

Accounting for variations among the models in number of glutamate molecules per vesicle, intersynaptic distance, and glutamate transporter density in the neuropil[2], the predicted effect of glutamate transporter density on spillover is generally consistent among the models. Intuitively, glutamate transporters should limit spillover, and the models confirm this notion. In the following discussion the focus will be on normalized spillover activity which is the ratio of activity at a remote synapse due to spillover to activity at an active synapse where release occurred. The normalized spillover activity will be quoted in the main text with the spillover value following in parentheses. Even with no transporters present in the model normalized peak spillover glutamate concentration is predicted to be less than 1% (30 μM) (Lehre and Rusakov, 2002), but, surprisingly, normalized

[1]Typical values of α and λ are 0.2 and 1.4, respectively.

[2]Parameter ranges: vesicle size size [4700-5000 glutamate], intersynaptic distance [465-1100 nm], transporter density [5-500 μM].

peak open probability for an NMDAR is approximately 50% (Franks et al. (2002), Rusakov and Kullmann (1998a), but see Barbour (2001)). Adding transporters to the model halved both the peak glutamate concentration and NMDAR open probability from spillover (Barbour (2001), Barbour and Häusser (1997), Rusakov and Kullmann (1998a), Lehre and Rusakov (2002)).

The effects of synaptic morphology and the glutamate release profile on spillover have been investigated. Spillover is relatively insensitive to the total tortuosity and extracellular volume fraction. Varying the tortuosity from 1.3 to 1.9 and the volume fraction from 0.1 to 0.7 is predicted to have a negligible effect on open probability of NMDARs. The degree of postsynaptic glial ensheathment influences not only the amplitude but also the time course of spillover in simulation (Rusakov, 2001). More ensheathment of the postsynaptic side generates glutamate concentrations that would lead to a roughly linear decrease in peak number of doubly-bound NMDARs due to spillover in the postsynaptic direction. Surprisingly, the amplitude of response of NMDARs based on glutamate concentrations on the presynaptic side would be approximately invariant to glial ensheathment but the time course is delayed. Multivesicular release increases spillover NMDAR peak open probability in a supralinear fashion; simultaneous release of four vesicles causes a ten-fold increase in NMDAR peak open probability due to spillover compared to spillover activation following release of a single vesicle (Barbour (2001), but see Rusakov (2001)). Barbour and Hausser (1997) simulated simultaneous single-vesicle release from multiple synapses arranged in a hexagonal close packing arrangement with a density of 1 synapse per cubic micron. With the number of released vesicles ranging from 5 to 238, the spillover glutamate profile at a synapse centered in the release volume of neuropil with transporters present had a peak amplitude that increased sublinearly with release number and exhibited prolonged rising and falling phase of the time course as number of released vesicles increased. An informative upper bound on the peak spillover glutamate concentration can be calculated by assuming simultaneous release from all synapses except the test

synapse with no transporters; the average glutamate concentration at the test synapse in that case is approximately 38 μM (Barbour and Häusser, 1997). Furthermore, looking at the fraction of doubly-bound NMDARs at the test synapse, it was predicted that concentrated, simultaneous release from many surrounding synapses is required for substantial crosstalk. For example, for 40% of NMDAR population to be doubly-bound, 86 synaptic releases is required; for 60% of NMDAR population to be doubly-bound, 238 synaptic releases is required (Barbour and Häusser, 1997). Increasing the diffusion constant of glutamate from 0.05 to 0.75 μm^2/ms is predicted to lower the normalized peak open probability of NMDARs at 465 nm from the release point from approximately 0.67 to 0.1 in the presence of 100 μM concentration of glutamate transporters (Rusakov and Kullmann, 1998a). The simulated addition of 0.6 μM background glutamate concentration has a mild effect on NMDAR response, raising the normalized peak open probability of NMDARs at 465 nm from the release point by approximately 0.05 at all diffusivity values.

In addition to the FEA models described above, Monte Carlo models have been used to investigate the effects of model parameters on glutamate spillover. One advantage of using FEA compared to other mathematical approaches is reduced computational load. However, realistic, complex boundary conditions are unwieldy in FEA, and furthermore, the stochasticity inherent in the system is lost. For these reasons, Monte Carlo models with explicit representation of individual receptors, transporters, glutamate molecules, and neuropil membrane geometry have also been used to investigate spillover. In this paradigm, the postsynaptic surface is populated with a specified number of AMPA and NMDA receptors, while astrocyte membrane is populated with a specified surface density of glutamate transporters. Comparison between Monte Carlo and FEA model-predicted effect of glutamate transporter density and multivesicular release on spillover reveals good agreement (Franks et al., 2002). Franks et al. (2002) compared spillover in a three-dimensional array of cubes representation of neuropil to spillover in an extended synaptic cleft

disk model to evaluate the effect of the tortuous neuropil on crosstalk. Independent of transporter density and the number of vesicles released, the model predicts lower crosstalk in the neuropil model where the convoluted extracellular space acts as a neurotransmitter sink. Simulations that include perisynaptic glutamate transporters around the remote synapse predict the peak synaptic glutamate concentration due to spillover will decrease by 50% as will the NMDAR peak open probability (Diamond, 2001). Additional computational experiments indicate that neuronal transporters alone are insufficient to affect spillover activation of extrasynaptic NMDARs at the active synapse (Diamond, 2001).

C.3 Outstanding Questions

So far in vitro experiments and computational models have worked to characterize the effects of neuropil morphology parameters and stimulation patterns on spillover. The principal difficulty in comparing the results from physiology experiments on spillover with computational model predictions is that the former measures NMDAR activity over the entire dendritic arbor by patch-clamp at the soma, while the latter measures spillover activity at a single synapse. Consequently, the interpretation of somaticly-measured NMDAR EPSCs with regards to spillover is ambiguous since it represents the average activity at all stimulated synapses onto the neuron, both active synapses and spillover synapses. Nevertheless, qualitative comparisons between in vitro experimental observations and computational model predictions are in agreement. AMPA receptors are insensitive to spillover due to their low affinity for glutamate. Glutamate transporter blockers enhance glutamate spillover. Increasing glutamate release by increasing the number of glutamate molecules in each vesicle, promoting multivesicular release, or increasing the spatial density of release cites all exacerbate spillover.

The big question remains: does glutamate spillover occur in CA1 in vivo? The results of previous experiments and simulations provide hints as to which neu-

ropil parameters are most important for spillover and under which physiological conditions is spillover most likely to occur. The probability of spillover is likely to be highest during high-frequency, synchronous firing of CA3 pyramidal neurons as occurs during hippocampal sharp waves (REFERENCE). An accurate spatiotemporal synaptic vesicle release profile in CA1 would be required for a realistic computer simulation of spillover, including the likelihood of multivesicular release. Additionally, experimental evidence is accumulating for glutamate release by astrocytes, implying an additional source of extracellular glutamate that deserves investigation (Parpura et al. (1994), Fellin et al. (2004), Volterra and Steinhäuser (2004), Angulo et al. (2004)). Spillover is highly sensitive to glutamate transporters, so having accurate astrocyte membrane distributions and kinetic models for the glutamate transporter types (e.g. GLT-1 GLAST, and EAAC) is crucial. The existence of NMDAR subtypes with long off-rates such as NR2B compared to NR2A implies the importance of including accurate distributions of the many NMDAR types and associated kinetic models. Extrasynaptic NMDARs would likely be more sensitive to glutamate spillover than synaptic receptors and should be included in an accurate model in the event their functional existence in CA1 is confirmed. Finally, a convincing simulation of spillover would incorporate accurate representations of neuropil membrane physiology, including both pyramidal neurons and astrocytes with their multitude of fine processes.

C.4 Quantal content of EPSC based on coefficient of variation

Here we build on the assumptions and ideas in Asztely et al. (1997) to show that a decrease in the coefficient of variation of NMDAR EPSC with glutamate transport blocked can be interpreted as more glutamate binding to NMDARs. Consider a CA1 pyramidal neuron receiving n synaptic inputs from CA3 neurons. Assume vesicular release at any synapse is a random process, the synapses are in-

dependent, and all synapses have identical probability of vesicular release p during an action potential. Also assume each synapse has the same number of NMDA receptors. The number of synapses that release glutamate during each CA3 action potential is a random number described by the binomial distribution. Assume n is large and p is small, so that the binomial distribution converges to a Poisson distribution, where the mean number of synaptic releases following an action potential is equal to np, which is also equal to the variance in the number of synaptic releases. Suppose the cumulative contribution from each synapse to the NMDAR EPSC in the CA1 pyramidal neuron is measured at the soma. The NMDAR EPSC will be a random number also described by the Poisson distribution with mean, λ, equal to npg, where g is the NMDAR synaptic weight, uniform across all synapses. The coefficient of variation (CV) of the NMDAR EPSC is equal to the standard deviation divided by the mean. Since the variance is equal to the square of standard deviation, and the variance equals the mean for the Poisson distribution then,

$$\frac{1}{CV^2} = \frac{mean^2}{variance} = mean = npg.$$

Suppose we make measurements of NMDAR EPSC with and without glutamate transporter blocker and compute the coefficient of variation in each case, designated CV_1 and CV_2, respectively. Consider the experimental results where the coefficient of variation of NMDAR EPSC drops when glutamate transporter blocker is added to the bath. If

$$CV_1 < CV_2,$$

then

$$\frac{1}{CV_1^2} > \frac{1}{CV_2^2}.$$

After substitution

$$n_1 pg > n_2 pg,$$

so

$$n_1 > n_2.$$

The interpretation of this result is that a larger number of released vesicles was sensed by the NMDA receptors with glutamate transporter antagonists in the bath.

D

Future Work

A 3D reconstruction of a 5 μm cubic region of rat hippocampus was made from serial section transmission electron micrographs (Section II.1). Here details are given as to how synapses in the reconstruction could be populated with synaptic receptors according to the size of the postsynaptic area. An improved kinetic model for NMDA receptors is described. Furthermore, two improved kinetic models for glutamate transporters are derived from literature.

D.1 Rules for adding AMPARs to spines

Takumi et. al. (1999) used postembedding immunogold labeling of AMPA receptors to compare the receptor density in 50-70 nm sections of mossy fiber - CA3 (MF-CA3) synapses and Schaffer collateral-CA1 synapses in adult rat stratum radiatum. The mean number of particles in sections near MF-CA3 PSD diameter was 5.4 pm 0.3 (n=95) for an average PSD diameter of 320 nm. Despite the low correlation between MF-CA3 PSD size and particle number in sections (r=0.42), we will assume that particle number per section varies linearly with PSD length. Since the section thickness is fixed, this implies that PSD area increases linearly with PSD length; therefore, a constant particle surface density can be calculated for the MF-CA3 synapse. The average MF-CA3 synapse has a surface

density of 2.4e-4 particles per square nanometer which equates to 20 particles total in the PSD, assuming 70 nm thick sections. The average number of AMPA receptors at mossy fiber terminal-CA3 pyramidal cell synapse has been estimated to be 100, based on variance analysis of spontaneous miniature EPSCs recorded from the soma of 6 CA3 neurons from adult rat (Jonas et al. (1993), Forti et al. (1997)). We conclude that immunogold labeling identifies 20% of the total AMPA receptor population and assume that this labeling ratio holds true for Schaffer collateral-CA1 (SCC) synapses of all morphological types.

The mean number of particles in sections near SCC PSD diameter was 2.3 *pm* 0.2 (r=0.78, n=107) for an average PSD diameter of 260 nm. The average SCC synapse has a surface density of 1.2e-4 particles per square nanometer. Given the labeling ratio reported above, the actual surface density of AMPA receptors in the 260 nm diameter SCC PSD is 6.2e-4 receptors per square nanometer. Takumi et al. report that they excluded perforated synapses from the analysis and only included continuous PSDs. This restriction of the applicability of the AMPA receptor density calculation in populating the reconstruction is mitigated by considering, the work of Ganeshina et al. (2004a) who also used postembedding immunogold labeling of AMPA receptors to compare the receptor density along the diameter of three synapse types in adult rat stratum radiatum.

Individual synapses can be classified as belonging to one of three groups based on their morphology (Ganeshina et al., 2004b). Segmented, completely-partitioned synapses (SCP) have perforated synaptic junctions with partitions emanating from the PSD that invaginate the presynaptic bouton; the PSD is completely separated into individual segments. Other perforated (OP) synaptic morphologies exhibit segmented PSDs without the dividing protrusion. All other synapses are nonperforated (NP). Ganeshina et al. report an average NP PSD diameter of 190 *pm* 40 nm, smaller than that observed by Takumi et al., and an average OP PSD diameter of 320 *pm* 70 nm. The mean number of particles observed in the NP and OP PSD's was 2.34 *pm* 0.23 and 14.2 *pm* 1.4, respectively,

again much different from the mean of 7 particles reported by Takumi et. al. However, if we pool the NP (n=96) and OP (n=27) PSDs, the average PSD diameter is 230 nm and mean number of gold particles in the population is 5. Thus, the results of Takumi et al. are consistent with the observations of Ganeshina et al. if the synapse population of Takumi et al. consists of NP and OP PSDs. Note that the immunoreactivity of the AMPA receptor antibodies used in the two experiments may be different, and differences in the recorded number of gold particles per PSD would be expected for that reason. Since the third class of synapses containing SCP PSDs is reported to constitute a mere 2% of the entire axospinous synaptic population, we propose to populate synapses in the reconstruction using the AMPA receptor density calculated above for SCC synapses.

Consistent with Takumi et al. (1999) synapses with PSD diameter less than 180 nm will not be populated with AMPA receptors. Racca et al. (2000) saw no indication of a minimal synapse size below which AMPARs are absent but did observe that AMPARs are absent from 10% to 15% of synapses containing NMDAR. The receptors will be randomly located on the PSD, implying a strict intrasynaptic configuration, based on observations of rat stratum radiatum (Ganeshina et al., 2004b). No perisynaptic or extrasynaptic AMPA receptors will be included.

D.2 Rules for adding NMDAR subtypes (NR2A and NR2B)

By uncaging glutamate at individual spines in adult rat CA1, Sobczyk et al. (2005) recorded peak synaptic current from 98 spines at room temperature with holding potential of -70 mV. The mean peak synaptic current was 4.87 pA, and mean spine volume of sample set was 0.08 *pm* 0.03 fL. Consistent results were obtained with whole-cell patch clamp recordings of EPSCs from the soma of cultured hippocampal neurons (8 to 10 days in vitro) at room temperature

(McAllister and Stevens, 2000). The mean peak synaptic current was 4.25 pA (n=11) at a holding potential of -60 mV.

The population of synaptic NMDA receptors in rat hippocampus has been shown to shift from mostly NR1/NR2B heterotetramers at birth to predominantly NR1/NR2A type in the adult animal (Monyer et al., 1994). Therefore, the synaptic currents measured by Sobczyk et al. and McAllister et al. arise from the opening of mostly NR2A-containing receptors (NR2A-NMDARs) and some NR2B-containing receptors (NR2B-NMDARs). The contribution of NR2A-NMDARs was isolated by the application of ifenprodil, an NR2B-specific NMDAR antagonist, to the hippocampal slice (Sobczyk et al., 2005). The NR2A-associated EPSC was 3.2 *pm* 1.1 pA measured with a membrane potential of -70 mV. Given an NR2A-NMDAR ion channel conductance of 47.6 *pm* 1.1 pS (Lin et al., 2004) and an NMDAR reversal potential of 3 mV (Nowak et al., 1984), the number of open NR2A- NMDARs at the peak is approximately 1 (Nimchinsky et al., 2004). By assuming that the synaptic current difference (4.87 - 3.2 pA) is due to current through NR2B-NMDAR ion channels with conductance of 38.4 *pm* 0.8 pS (Mohrmann et al., 2002), the number of open NR2B-NMDARs at the peak is calculated to be approximately 0.6. This implies that in roughly 60% of the sampled spines a single NR2B-NMDAR was open during the peak current flow.

The mean number of NR2A- and NR2B-NMDARs at an average size synapse, i.e. spine volume of 0.08 fL, was calculated by trial-and-error using Monte Carlo simulations of vesicular release in a model synapse. In the MCell modeling environment, vesicles containing 3000 glutamate molecules (Burger et al., 1989) were released into a 20 nm-wide square synapse 340 nm on a side with a 310 nm diameter PSD. The PSD area in the model (.075 μm^2) was within one standard deviation of the mean PSD size sampled by Sobczyk et al.; based on histological measurements by Harris and Stevens (1989) using serial electron microscopy and 3-D reconstruction, a sample population of 100 synapses in adult rat stratum radiatum had mean spine volume of 0.062 *pm* 0.08 fL and mean PSD area of 0.069

pm 0.08 square microns. The PSD surface was populated with a specified number of NR2A-NMDARs and NR2B-NMDARs whose kinetic model and rate constants are described below. The results from three vesicular release sites (PSD center, 75 nm from center, and 150 nm from center) were averaged. It was discovered that in order to have approximately 1 NR2A-NMDAR and 0.6 NR2B-NMDARs open at peak, the PSD had to contain approximately 8 NR2A-NMDARs total and 4 NR2B-NMDARs total as shown in Figure D.1B. Postembedding immunogold labeling of NMDA receptors in CA1 synapses offers a lower bound on the total number of receptors in the synapse. Mean particle numbers of 7 pm 5 (Racca et al., 2000)(n=250, average synapse area of 0.04 pm .023 μm^2) and 5 pm 1 (Ganeshina et al., 2004a)(n=100, Mean synapse area = .04 μm^2) are consistent with the mean of 12 calculated above (but see Takumi et al. (1999)). The peak number of open NR2A- and NR2B-NMDARs in the model were found to be sensitive to the total number of receptors in the PSD based on simulations with 7 NR2A-NMDARS and 5 NR2B-NMDARS or 9 NR2A-NMDARS and 3 NR2B-NMDARS. Simulations with PSDs containing 8 NR2A-NMDARs were repeated with vesicle sizes of 5000 glutamate as shown in Figure D.1A. NR2A activation was seen to be as sensitive to number of glutamate in vesicle as to total receptor number in the PSD, but not so much for the number of open receptors.

Racca et al. (2000) used postembedding immunogold labeling of NMDA receptors to measure the receptor density in 70-80 nm sections of Schaffer collateral-CA1 synapses in adult rat stratum radiatum. The number of immunogold particles per synapse showed a weak positive correlation with PSD area (120 particles per square micron), consistent with immunogold staining by Ganeshina et al. (2004b) and the correlation between EPSC amplitudes and spine size (Sobczyk et al., 2005). Working with 98 spines in adult rat stratum radiatum Sobczyk et al. (2005) used glutamate uncaging and calcium imaging to show that NR2B-containing receptors are preferentially expressed on small spines and the number of NR2A-containing receptors is independent of spine size. However, few large spines were included in

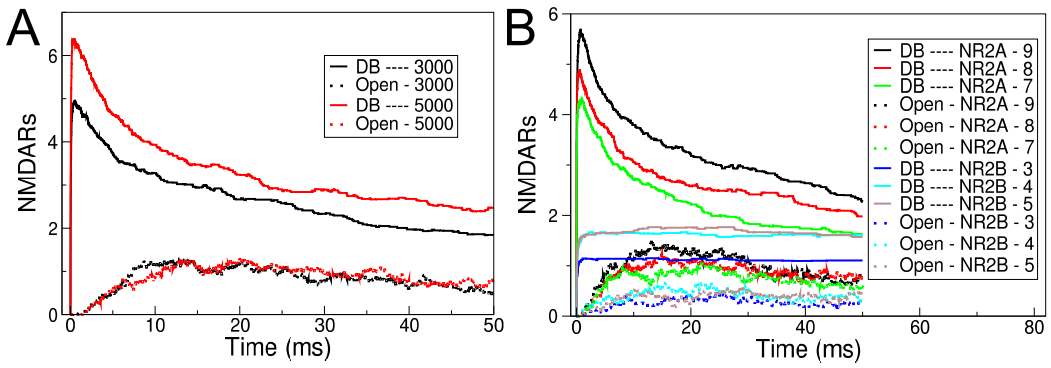

Figure D.1 **Impact of vesicle size and NMDAR subunit distribution on number of open NMDARs.** (A) Effect of number of glutamate in vesicle on NR2A activation. In legend 'DB' represents the number of doubly-bound receptors. (B) Sensitivity of Activation to NR2A/NR2B ratio. In legend 'DB' represents the number of doubly-bound receptors, and the number on right-side indicates the number of each receptor type in the model. Three simulations are plotted with (NR2A,NR2B) combinations of (9,3), (8,4), and (7,5).

the sample set of Sobczyk et al., and Takumi et al. (1999) found no statistically significant correlation between PSD diameter and NMDAR count. We propose to populate all PSD's in the reconstruction with 8 NR2A-NMDARs and 4 NR2B-NMDARs, independent of PSD size.

NMDAR distribution in PSD will be Gaussian aligned to geometric center of the synapse as an approximation of the real receptor distribution which does show a concentration of receptors in the PSD center and fewer towards the edge (Racca et al., 2000). No peri- or extra-synaptic receptors will be included in the model.

D.3 NMDA Receptor Kinetic Models

The same AMPA receptor kinetic model as described in Section IV.3.A will be used in future models. The NMDA receptor kinetic model from before

will be replaced entirely with the kinetic models in Figure D.2 with kinetic rates provided in Table D.1 and Table D.2. The kinetic model for NMDA receptors of NR2A subtype is based on patch clamp measurements of single channel currents of rat NR1A and NR2A cDNA transfected into human embryonic kidney cells at 23 °C (Popescu and Auerbach, 2003). Popescu et al. observed spontaneous, modal transitions in the gating kinetics between three regimes with different open-channel lifetimes which they designated high (H), medium (M), and low (L). In fact, individual channels were observed to cycle through the different modes, but since the cycle time was long compared to the duration of simulation the channel mode will be fixed. Their results indicate that at synapses the L-mode predominates, since the EPSC decay time constant of the L-mode (42 ms) is similar to synaptic aggregate EPSC measurements (50ms). The kinetic model for NMDA receptors of NR2B subtype is based on patch clamp measurements of single channel currents of rat NR1A and NR2B cDNA transfected into human embryonic kidney cells at 23 C (Banke and Traynelis, 2003).

Table D.1 Constants for NMDA Receptors of NR2A (L-Mode) Receptor Kinetic Model.

CUCM	CMCU	CMC1	C1CM	C1D	DC1	C1C2
$M^{-1}s^{-1}$	s^{-1}	$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}	s^{-1}
40e6	60	20e6	120	20	1	120
C2C1	C2C3	C3C2	C3O1	O1C3	O1O2	O2O1
s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}
160	600	2600	2500	2200	3500	660

D.4 Add glutamate receptor subtypes (GLT-1 and GLAST)

Approximately, 10% of the glutamate transported into astrocytes in hippocampus is mediated by GLAST (human homologue is EAAT1). In stratum

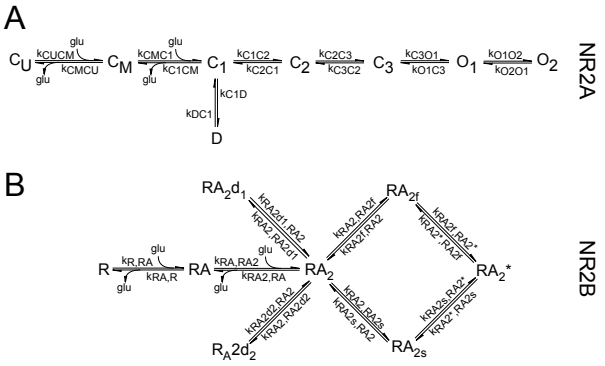

Figure D.2 **Kinetic models for NMDA receptors of (A) NR2A and (B) NR2B subtypes for third generation neuropil model.** (A) CU is the unbound close state, and CM is the single bound closed state. C1, C2, and C3 are doubly-bound closed states. D is a desensitized closed state, and O1 and O2 are open conducting states. The kinetic rates describing transition from unbound state to doubly-bound C1 state are from Lester and Jahr (1992). Conductance of open states 47.6 *pm* 1.1 pS (n = 5) at -60 mV that did not vary with voltage (Lin et al., 2004). (B) R is the unbound close state, and RA and RA2 are the single bound and double bound closed states, respectively. RA2d1 and RA2d2 are desensitized closed states, and RA2f and RA2s are closed intermediate states. RA2* is the open conducting state. Conductance of open state is 38.4 *pm* 0.8 pS (Mohrmann et al., 2002).

radiatum of adult rats, the volumetric concentration of GLAST has been measured as 3200 per cubic micron of neuropil (Lehre and Danbolt, 1998). Lehre and Danbolt estimated a density of 1.4 square microns of astroglial plasma membrane per cubic micron of neuropil using stereological methods and calculated the astrocyte surface density of GLAST to be 2300 per square micron. We will measure the density of astrocyte surface area per unit volume of neuropil in the reconstruction and re-estimate the membrane surface density of GLAST proteins. An upper bound of 25000 transporters per square micron can be calculated based on trans-

Table D.2 Constants for NMDA Receptors of NR2B Receptor Kinetic Model.

RA2f,RA2	RA2*,RA2s	RA2*,RA2f	RA2s,RA2
s^{-1}	s^{-1}	s^{-1}	s^{-1}
182	182	135	135
RA,RA2	RA2,RA	RA2,RA2d2	RA2d2,RA2
$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}
9.5e6	60	70	2.8
RA2,RA2d1	RA2d1,RA2	RA,R	R,RA
s^{-1}	s^{-1}	s^{-1}	$M^{-1}s^{-1}$
45	0.5	29	2*9.5e6
RA2,RA2f	RA2s,RA2*	RA2f,RA2*	RA2,RA2s
s^{-1}	s^{-1}	s^{-1}	s^{-1}
1557	1557	89	89

porter membrane footprint (48 *pm* 5 nm2, human EAAC expressed in Xenopus oocytes; (Eskandari et al., 2000)).

The kinetic model for GLAST glutamate transporters is based on patch clamp measurements at 22 °C of membrane currents in Xenopus oocytes expressing human transporter EAAT1 (Wadiche and Kavanaugh, 1998). The kinetic model is given in Figure D.3 and rate constants are provided in Table D.3.

The other 90% of the glutamate transported into astrocytes in hippocampus is mediated by GLT-1 (human homologue is EAAT2). In stratum radiatum of adult rats, the volumetric concentration of GLT-1 has been measured as 12000 per cubic micron of neuropil (Lehre and Danbolt, 1998). Again, assuming a density of 1.4 square microns of astroglial plasma membrane per cubic micron of neuropil, the astrocyte surface density of GLT-1 is 8500 per square micron, and these estimates will be verified with direct measurements from the reconstruction. A detailed kinetic model for GLT-1 receptors has been derived based on whole-cell patch clamp current measurements at 23 °C of human embryonic kidney cells expressing GLT-1 transporters (Bergles et al., 2002). This model includes explicit binding and unbinding steps for sodium, potassium, hydrogen and glutamate. Here a simplified version of this kinetic model was derived by assuming constant ion concentrations.

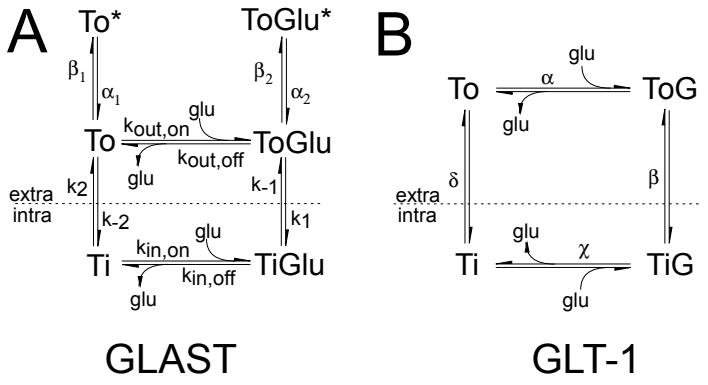

Figure D.3 **Kinetic models for (A) GLAST and (B) GLT-1 glutamate transporters to be used in third-generation neuropil model.** (A) To and Ti are the unbound closed states, and ToGlu and TiGlu are the bound closed states. To* and ToGlu* are anion conducting states, which are not important for the proposed experiments but are included for completeness. In our simulations, $K_{in,on}$ will be set to zero. (B) To is the unbound state, ready to bind glutamate and transition to ToG. The transition from ToG to TiG represents translocation, while unbinding and intracellular release of glutamate is represented by transition to Ti. The reset of the transporter is captured in the model by a return to To state.

Constant intracellular ion concentrations for Na+, K+, H+, and glutamate are assumed to be 10 mM, 140 mM, 7e-5 mM, and 50 μM, respectively (Bergles et al. (2002), Alberts et al. (2002)). Constant extracellular ion concentrations for Na+, K+, and H+ are assumed to be 145 mM, 3 mM, and 4e-5 mM, respectively (Alberts et al. (2002), Longuemare et al. (1999)). The derivation is given in Appendix D.4.A. Figure D.3 illustrates the simplified kinetic model, and the derived kinetic rates are given in Table D.4.

Table D.3 Rate Constants for GLAST transporter kinetic model.

k_1	k_{-1}	k_2	k_{-2}	$k_{out,on}$	$k_{out,off}$
s^{-1}	s^{-1}	s^{-1}	s^{-1}	$M^{-1}s^{-1}$	s^{-1}
16	2.9	885	200	6.80e6	3.06e1
α_1	β_1	α_2	β_2	$k_{in,on}$	$k_{in,off}$
s^{-1}	s^{-1}	s^{-1}	s^{-1}	$M^{-1}s^{-1}$	s^{-1}
8.09e3	100	1.26e3	70	6.80e6	37.2

Table D.4 Rate Constants for simplified GLT-1 transporter kinetic model.

k_α	$k_{-\alpha}$	k_β	$k_{-\beta}$	k_γ	$[glu_i]k_{-\gamma}$	k_δ	$k_{-\delta}$
$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}
5.7e6	2.4e6	4e2	1.2e3	7.6e2	5.8e-1	8.4	2.9e-2

D.4.A Simplification of Glutamate Transporter GLT-1 Kinetic Model

A detailed kinetic model for glutamate transporter GLT-1 has been derived based on whole-cell patch clamp current measurements at 23C of human embryonic kidney cells expressing GLT-1 transporters Bergles et al. (2002). ToNa2H, ToNa3GH, and Tik are anion conducting states, and ToNa2H and ToNa2 are anion leak states. ToNa3GH exhibits fast transient anion conductance. The anion conducting states are included to be thorough but are not important for the proposed experiments.

Differential equations can be derived describing rate of change of original GLT-1 kinetic model states.

$$\begin{aligned}\frac{d}{dt}[ToK] &= k_{-15}[TiK] - k_{15}[ToK] \\ &\quad - k_{16}[ToK] + k_{-16}[K_o][To*] \\ \frac{d}{dt}[To*] &= k_{16}[ToK_o] + k_{-1}[ToNa1]\end{aligned}$$

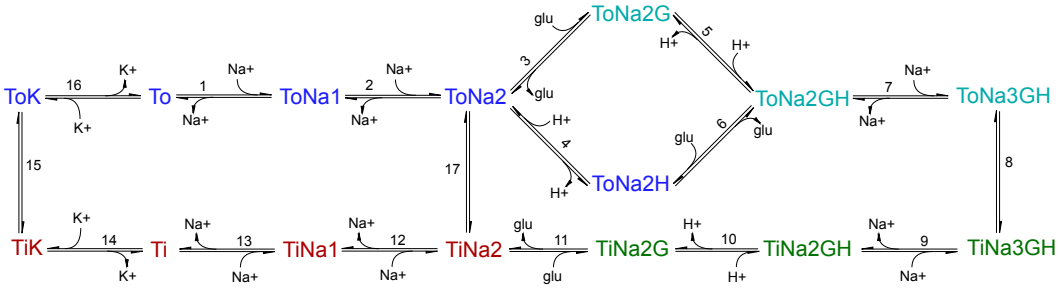

Figure D.4 Original Kinetic Model for Glutamate Transporter GLT-1

Original Kinetic Model for Glutamate Transporter GLT-1 Colors represent corresponding states in this original kinetic model and the simplified kinetic model in Figure D.5.

$$
\begin{aligned}
&- k_1[Na_o][To*] - k_{-16}[K_o][To*] \\
\frac{d}{dt}[ToNa1] &= k_1[To*][Na_o] + k_{-2}[ToNa2] \\
&- k_{-1}[ToNa1] - k_2[Na_o][ToNa1] \\
\frac{d}{dt}[ToNa2] &= k_2[Na_o][ToNa1] - k_{-2}[ToNa2] - k_3[glu_o][ToNa2] \\
&+ k_{-3}[ToNa2G] - k_{-4}[H_o][ToNa2] + k_4[ToNa2H] \\
&- k_{-17}[ToNa2] + k_{17}[TiNa2] \\
\frac{d}{dt}[ToNa2H] &= k_{-4}[H_o][ToNa2] - k_4[ToNa2H] \\
&- k_{-6}[glu_o][ToNa2H] + k_6[ToNa2GH] \\
\frac{d}{dt}[ToNa2G] &= k_3[glu_o][ToNa2] - k_{-3}[ToNa2G] \\
&+ k_{-5}[ToNa2GH] - k_5[H_o][ToNa2G] \\
\frac{d}{dt}[ToNa2GH] &= k_{-6}[glu_o][ToNa2H] - k_6[ToNa2GH] - k_{-5}[ToNa2GH] \\
&+ k_5[H_o][ToNa2G] - k_7[Na_o][ToNa2GH] + k_{-7}[ToNa3GH] \\
\frac{d}{dt}[TiNa3GH] &= k_8[ToNa3GH] - k_{-8}[TiNa3GH] \\
&+ k_{-9}[Na_i][TiNa2GH] - k_9[TiNa3GH]
\end{aligned}
$$

Table D.5 Rate constants for original glutamate transporter GLT-1 kinetic model

	Forward	Units	Backward	Units
1	1.00e4	M-1s-1	1.00e2	s-1
2	1.00e4	M-1s-1	5.00e2	s-1
3	6.00e6	M-1s-1	500	s-1
4	6.00e11	s-1	7.00e2	M-1s-1
5	6.00e11	M-1s-1	7.00e2	s-1
6	6.00e6	s-1	5.00e2	M-1s-1
7	1.00e4	M-1s-1	1.00e3	s-1
8	2.00e3	s-1	1.90e3	s-1
9	1.00e3	M-1s-1	4.00e4	s-1
	Forward	Units	Backward	Units
10	3000	M-1s-1	9.00e10	s-1
11	3000	M-1s-1	1.00e5	s-1
12	1.00e5	M-1s-1	2.00e7	s-1
13	1.00e5	M-1s-1	1.00e8	s-1
14	1.00e6	s-1	1.00e3	M-1s-1
15	4.00e1	s-1	10	s-1
16	2.00e4	s-1	1.00e6	M-1s-1
17	1.40	s-1	1.00e-2	s-1

$$\frac{d}{dt}[TiNa2GH] = k_{-10}[H_i][TiNa2G] - k_{10}[TiNa2GH]$$
$$- k_{-9}[Na_i][TiNa2GH] + k_9[TiNa3GH]$$
$$\frac{d}{dt}[TiNa2G] = k_{10}[TiNa2GH] - k_{-10}[H_i][TiNa2G]$$
$$- k_{11}[TiNa2G] + k_{-11}[glu_i][TiNa2]$$
$$\frac{d}{dt}[TiNa2] = k_{-17}[ToNa2] - k_{17}[TiNa2] + k_{11}[TiNa2G]$$
$$- k_{-11}[glu_i][TiNa2] - k_{12}[TiNa2] + k_{-12}[Na_i][TiNa1]$$
$$\frac{d}{dt}[TiNa1] = k_{-13}[Na_i][Ti*] - k_{13}[TiNa1]$$
$$+ k_{12}[TiNa2] - k_{-12}[Na_i][TiNa1]$$
$$\frac{d}{dt}[Ti*] = k_{-14}[TiK] - k_{14}[K_i][Ti*]$$
$$- k_{-13}[Na_i][Ti*] + k_{13}[TiNa1]$$

$$\frac{d}{dt}[TiK] = k_{15}[ToK] - k_{-15}[TiK]$$
$$- k_{-14}[TiK] + k_{14}[K_i][Ti*]$$

Assuming the extracellular and glial intracellular concentrations of potassium, sodium, and hydrogen are constant, then a simplified kinetic model can be derived as seen below. The mapping between states in the original and simplified kinetic models are color-coded. The task is to solve for the kinetic rates $\alpha, \beta, \gamma, \delta$.

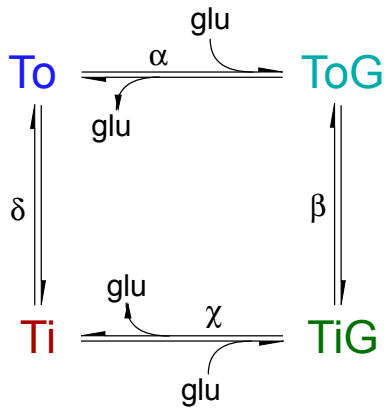

Figure D.5 Simplified GLT-1 Kinetic Model.

Simplified GLT-1 Kinetic Model Colors represent corresponding states in this simplified kinetic model and the original kinetic model in Figure D.4.

Differential equations can be derived describing rate of change of simplified GLT-1 kinetic model states.

$$\frac{d}{dt}[Ti] = k_{-\delta}[To] - k_{\delta}[Ti] - k_{-\gamma}[glu_i][Ti] + k_{\gamma}[TiG]$$
$$\frac{d}{dt}[TiG] = k_{-\gamma}[glu_i][Ti] - k_{\gamma}[TiG] - k_{\beta}[TiG] + k_{-\beta}[ToG]$$
$$\frac{d}{dt}[ToG] = k_{\beta}[TiG] - k_{-\beta}[ToG] - k_{-\alpha}[ToG] + k_{\alpha}[glu_o][To]$$
$$\frac{d}{dt}[To] = k_{-\alpha}[ToG] - k_{\alpha}[glu_o][To] + k_{\delta}[Ti] - k_{-\delta}[To]$$

By definition the simplified GLT-1 kinetic model state are expressible as sums of states in the original GLT-1 kinetic model with corresponding time derivatives.

$$[To] = [ToK] + [To*] + [ToNa1] + [ToNa2] + [ToNa2H]$$
$$[Ti] = [TiK] + [Ti*] + [TiNa1] + [TiNa2]$$
$$[ToG] = [ToNa2G] + [ToNa2GH] + [ToNa3GH]$$
$$[TiG] = [TiNa2G] + [TiNa2GH] + [TiNa3GH]$$
$$\frac{d}{dt}[To] = \frac{d}{dt}[ToK] + \frac{d}{dt}[To*] + \frac{d}{dt}[ToNa1] + \frac{d}{dt}[ToNa2] + \frac{d}{dt}[ToNa2H]$$
$$\frac{d}{dt}[Ti] = \frac{d}{dt}[TiK] + \frac{d}{dt}[Ti*] + \frac{d}{dt}[TiNa1] + \frac{d}{dt}[TiNa2]$$
$$\frac{d}{dt}[ToG] = \frac{d}{dt}[ToNa2G] + \frac{d}{dt}[ToNa2GH] + \frac{d}{dt}[ToNa3GH]$$
$$\frac{d}{dt}[TiG] = \frac{d}{dt}[TiNa2G] + \frac{d}{dt}[TiNa2GH] + \frac{d}{dt}[TiNa3GH]$$

Rate constants for the simplified kinetic model are calculated by substituting the original model differential equations and simplified model differential equations into the time derivatives immediately above.

$$k_{\alpha} = \frac{k_1\, k_{16}\, k_2\, (k_3\, k_4 + [H_o]\, k_{-4}\, k_{-6})\, [Na_o]}{k_{-1}\, k_{-2}\, k_4\, (k_{16} + k_{-16}\, [K_o]) + k_1\, k_{16}\, ((k_2 + k_{-2})\, k_4 + [H_o]\, k_2\, k_{-4})\, [Na_o]}$$
$$k_{-\alpha} = \frac{(k_{-3}\, k_{-5} + [H_o]\, k_5\, k_6)\, k_{-7}}{k_{-5}\, k_{-7} + [H_o]\, k_5\, (k_{-7} + k_7\, [Na_o])}$$
$$k_{\beta} = \frac{[H_i]\, k_{-10}\, k_{-8}\, k_{-9}\, [Na_i]}{k_{10}\, k_9 + [H_i]\, k_{-10}\, (k_9 + k_{-9}\, [Na_i])}$$
$$k_{-\beta} = \frac{[H_o]\, k_5\, k_7\, k_8\, [Na_o]}{k_{-5}\, k_{-7} + [H_o]\, k_5\, (k_{-7} + k_7\, [Na_o])}$$
$$k_{\gamma} = \frac{k_{10}\, k_{11}\, k_9}{k_{10}\, k_9 + [H_i]\, k_{-10}\, (k_9 + k_{-9}\, [Na_i])}$$
$$k_{-\gamma} = \frac{k_{-11}\, k_{-12}\, k_{-13}\, k_{-14}\, [Na_i]^2}{k_{12}\, k_{13}\, (k_{-14} + k_{14}\, [K_i]) + k_{12}\, k_{-13}\, k_{-14}\, [Na_i] + k_{-12}\, k_{-13}\, k_{-14}\, [Na_i]^2}$$
$$k_{\delta} = \frac{k_{12}\, k_{13}\, k_{14}\, k_{-15}\, [K_i] + k_{-12}\, k_{-13}\, k_{-14}\, k_{17}\, [Na_i]^2}{k_{12}\, k_{13}\, (k_{-14} + k_{14}\, [K_i]) + k_{12}\, k_{-13}\, k_{-14}\, [Na_i] + k_{-12}\, k_{-13}\, k_{-14}\, [Na_i]^2}$$
$$k_{-\delta} = \frac{k_4\, (k_{15}\, k_{-16}\, k_{-1}\, k_{-2}\, [K_o] + k_1\, k_{16}\, k_{-17}\, k_2\, [Na_o])}{k_{-1}\, k_{-2}\, k_4\, (k_{16} + k_{-16}\, [K_o]) + k_1\, k_{16}\, ((k_2 + k_{-2})\, k_4 + [H_o]\, k_2\, k_{-4})\, [Na_o]}$$

Constant intracellular ion concentrations for Na+, K+, H+, and glutamate are assumed to be 10 mM Bergles et al. (2002), 140 mM, 7e-5 mM Alberts et al. (2002), and 50 μM, respectively Bergles et al. (2002), cf. Longuemare et al. (1999). Constant extracellular ion concentrations for Na+, K+, and H+ are assumed to be 145 mM, 3 mM Longuemare et al. (1999), and 4e-5 mM Alberts et al. (2002), respectively. The final numerical values for the simplified GLT-1 kinetic model rate constants are given in Table D.6.

Table D.6 Rate Constants for Simplified Glutamate Transporter GLT-1 Kinetic Model

k_α	$k_{-\alpha}$	k_β	$k_{-\beta}$	k_γ	$[glu_i]k_{-\gamma}$	k_δ	$k_{-\delta}$
5.7e6	2.4e6	4e2	1.2e3	7.6e2	5.8e-1	8.4	2.9e-2
$M^{-1}s^{-1}$	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}	s^{-1}

D.5 Add neuronal transporters

Experimental evidence suggests that pyramidal neurons, especially glutamatergic neurons, in hippocampus express glutamate transporter EAAC (EAAT3 in humans) on the soma, dendrites, and dendritic spines (Rothstein et al., 1994). Exclusively postsynaptic, EAAC is excluded from the synaptic membrane and instead is preferentially found in the perisynaptic membrane on the spine (Conti et al. (1998), He et al. (2000)). Importantly, EAAC knockout in rats leads to pathological symptoms including seizures and mild toxicity (Conti et al., 1998). Furthermore, experimental observation in vitro found that glutamate spillover was fully suppressed by neuronal transporters(Diamond, 2001). And so although GLT and GLAST are thought to be the dominant transporters of glutamate in the hippocampus, it would be interesting to include neuronal transporters in the model. Unfortunately, quantitative estimates of the transporter concentration in hippocampus are lacking (Danbolt, 2001). Before EAAC transporters can be incorporated into MCell models estimates of the neuronal surface density are required, and an accurate kinetic model must be developed.

E

Software Encyclopedia

Here is a collection of most of the software tools used in the CA1 hippocampus reconstruction. A brief description is provided along with directions to learn more about the program. If no authoring credit is given, then the tool was written by Tom Bartol or Justin Kinney. Tools that simply convert between file formats are grouped with their parent program.

E.1 RECONSTRUCT3D

RECONSTRUCT3D is a Windows tool for generating contours from EM image stack. The program allows the user to align the images, compensating for translation, rotation, and skew. The user can then manually segment the image stack by tracing the outline of cellular morphology, generating contours. The contours along with any annotations can be exported in XML format. RECONSTRUCT3D was developed in Kristen Harris' lab. A Windows binary executable is available for free download (`synapses.clm.utexas.edu/tools/reconstruct/reconstruct.stm`).

E.2 CONTOUR_TILER

CONTOUR_TILER stitches together contours on adjacent sections with triangular polygons, thus generating a surface mesh from the set of all contours. CONTOUR_TILER is a C++ program developed in Chandrajit Bajaj's lab (Bajaj et al., 1996). The application of CONTOUR_TILER in our reconstructions is described in Section II.1.D with suggested improvements in Section II.2.

poly2mesh

Converts single input poly file to mesh format. Output written to STD-OUT. Written in C.

E.3 reconstruct2contourtiler

reconstruct2contourtiler prepares contours for meshing and then passes the contours to CONTOUR_TILER. First, XML contours are converted into pts and config files. Next, nonrational uniform b-splines are fit to the contours, and then the b-splines are sampled with curve regions of high curvature being more densely sampled than low-curvature regions. Finally, CONTOUR_TILER is executed to mesh the sampled contours. Output files written to specified directory. Written in C++. The following help information is accessible by typing 'reconstruct2contourtiler -h' on the command line.

```
NAME
       reconstruct2contourtiler - generate contour_tiler input files from contours

SYNOPSIS
       reconstruct2contourtiler [options]

DESCRIPTION
       Converts reconstruct contour format to contour_tiler input format.
       All files in input directory are assumed to be
       of the form filename_prefix.section#.
       min_section to max_section is the section range to be converted.
       Section_thickness should be in same scale as x,y contour points.
       x,y and z coordinates of sampled splines will be multipled by scale in output.
```

```
       CAPPING_FLAG=1 to attempt end capping.CAPPING_FLAG=0 to leave ends open.
       DEVIATION_THRESHOLD is the maximum allowed deviation of the spline from raw
       contour points in scaled units. Set
       DEVIATION_THRESHOLD to 0 to disable thresholding.

EXAMPLES
       reconstruct2contourtiler -i ./contours -f myContours -n 10 -x 100 -t .07 -s 1000 -o ./contour_tiler_output -d 2
              Read contours from directory './contours' and write contour_tiler output
              files to directory 'contour_tiler_output'. The input contour files have the
              name scheme 'myContours.#' where # is the section number which varies from
              10 to 100. The distance between contours in the direction of sectioning
              is .070 microns. The contour_tiler output data will be in nanometers as
              dictated by the 1000 scaling. Capping directives will be included in the
              output data. The interpolated contours will not deviate from the input contours
              by more than 2 (nanometers since scaling is 1000).

OPTIONS
       --no_capping
              Do not include directives in the output data to cap the meshes,
              thus creating open surface meshes.
              Default is to cap the meshes at the minimum and maximum sections
              thereby creating closed surface meshes.

       --print_input_as_pts
              Print input points (possibly linearly interpolated) as .pts
              files in output directory.
              Default is to not print input points.

       -n NUM, --min_section=NUM
              The starting section number in the section range.
              Default is '60'.

       -x NUM, --max_section=NUM
              The ending section number in the section range.
              Default is '160'.

       -t NUM, --section_thickness=NUM
              Thickness of the section in microns. Each section
              is assumed to be of identical thickness.
              Default is '0.05'.

       -S NUM, --spline_samples_per_point=NUM
              Sample each spline NUM times between each
              pair of input points.
              Default is '10'.

       -s NUM, --scale=NUM
              The output data will be scaled by NUM. If NUM is
              equal to 1 then the output data is in microns.
              Default is '1' so the output
              data is in nanometers.

       -d NUM, --deviation_threshold=NUM
              The input contours are filtered before output.
              The deviation of the input and output contours
              is constrained to be less than NUM where units
              are input units scaled by --scale value.
              Default is '0'.

       -m NUM, --linear_threshold=NUM
```

```
              Sequential points on the input contours are constrained
              to have separation distance less than NUM. New points
              are inserted by linear interpolation if required.
              Units are input units scaled by --scale value.
              Default is '0' which means
              no threshold is enforced.

       -i DIRECTORY, --input_data_dir=DIRECTORY
              Directory containing input contours.
              Default is current directory.

       -o DIRECTORY, --output_data_dir=DIRECTORY
              Directory where output data will be written.
              Default is current directory.

       -O STRING, --output_script=STRING
              A bash script named STRING will be written
              to automate contour_tiling process.
              Default script name is 'mesh_and_convert.csh'.

       -f STRING, --input_filename_prefix=STRING
              The input contours will be read from
              'input_directory/STRING.min_section' to
              'input_directory/STRING.max_section'.
              Default is 'Volumejosef'.

       -I STRING, --ignore_contour=STRING
              Contours with name STRING will not be processed
              and no output data will be written for these contours.
              Default is no ignored contours.

       -h, --help
              Print reconstruct2contourtiler man page.

Justin Kinney 2008/12/16
```

E.4 DReAMM

DReAMM (`www.mcell.psc.edu/DReAMM/`) is a highly versatile visualization gui program built on PSC DX. Used to visualize meshes and MCell output. Also used to make animations of MCell output. Note that dx format has both ASCII and binary components.

dx2mesh

Converts DX file format to mesh format. Only parses separate positions and connections output from DX. Not generic. Output mesh written to STDOUT.

Written in C.

dx2mesh_2

Converts DX file format to mesh format. Parses larger subset of DX output format from binary files with ASCII header. More generic. Possibly only parses meshes with triangular polygons. Output mesh written to STDOUT. Written in C.

mesh2dx

Convert single input mesh file to dx format. Note nonstandard syntax: 'mesh2dx object.mesh object.dx', but necessary since DX mesh is written in binary format. Written in C.

mesh2groupdx

Convert all mesh files in a directory into a single dx file and associated bin files in standard MCell3 format. In fact, it works by converting mesh files to mdl format, configuring a small simulation in mdl format, and running MCell3 to generate viz output. Output files written to specified directory. Written in C++ and calls bash scripts and MCell3.

E.5 FILTERMESH

Written by Hughes Hoppe in the early 1990s in C++Hoppe et al. (1993), FILTERMESH is used for mesh manipulation and diagnostics and is the source of 'mesh' format. Messages are written to STDERR and output mesh is written to STDOUT. Three major drawbacks to using FILTERMESH are that it (1) can generate intersecting faces and (2) can generate nonmanifold vertices. Additionally, (3) it does not have an obvious way during remeshing of controlling the location of the output mesh relative to the input mesh. In other words the FILTERMESH

algorithm moves convex surfaces inwards towards the process' medial axis and concave surfaces outward. Since the majority of surface meshes in the reconstruction are convex, the volume of the mesh objects drop and the ECS volume fraction increases in concert. These issues are discussed in Section II.1.E.

The following commands are useful and commonly used. Also note that these commands can all be strung together on the command line.

- Filtermesh object.mesh -fillholes N -triangulate

  Fill all holes in object with less than or equal to N open edges and triangulate.

- Filtermesh object.mesh -trisubdiv -trisubdiv

  Execute two successive rounds of face refinement on object, thus increasing the number of faces in object by 16.

- Filtermesh object.mesh -fixvertices

  Fixes nonmanifold vertices by duplicating vertex and assigning some faces to reference the new vertex.

- Filtermesh object.mesh -nfaces N -inscribedc -reduce

  Remesh the object such that the output mesh has approximately N faces and try to enforce uniformity of the inscribed radius of all faces in the output mesh. The resulting output mesh has surprisingly uniform edge lengths and aspect ratios. However, comparing the remeshed surface to the original mesh, we see that concave regions of the mesh get pulled out and convex regions get pulled in.

E.6 IRIT

With the exception of GTS (with which I have no experience), IRIT (`www.cs.technion.ac.il/~irit/`) is the only tool we have for doing constructive solid geometry. The cool thing about IRIT is that it works. One drawback to IRIT is that output meshes are in STL-like format. See Section II.1.E for more details.

intersect_mesh

Perl script that calls IRIT to compute the intersection of two meshes. Output mesh written to specified output file.

subtract_mesh

Perl script that calls IRIT to cut one mesh using another mesh as the cutter. This was very useful for creating pockets in meshes where one cellular process protruded into another cell. Ideally, we would fix CONTOUR_TILER to create pockets and not bubbles in these cases. Output mesh written to specified output file.

mesh2irit

Converts mesh objects into IRIT file format. Output mesh written to STDOUT. Written in C.

irit2mesh

Converts irit output into mesh format. Remember that irit output format is STL-like which requires duplicate vertices to be identified by xyz position. However, during the constructive solid geometry arbitrarily small edges may be created. This makes the determination of equality for pairs of triplets of doubles a delicate procedure. Consequently, the tolerance for declaring two vertices equal is set really low such that the xyz locations must be identical to double precision. The hard part is left for meshheal. Output mesh written to STDOUT. Written in C.

meshheal

Program for merging duplicate vertices in meshes generated by IRIT. The output mesh from irit2mesh will likely have many open edges, i.e. have only one

adjacent face. Meshheal recognizes vertices on open edges, called free vertices, and looks for other free vertices with which to merge. The square of the distance between all pairs of free vertices (N^2 ouch!) is computed in 3D, and the pair of free vertices separated by the shortest distance are merged. To avoid the N^2 algorithm, consider projecting all free vertex locations onto a random vector and then evaluating for merge free vertices that are neighbors after projection. Neighbors on the projection are not necessarily close in 3D, but if vertices are close in 3D then they will be close on the projection. Output mesh written to STDOUT. Written in C++.

E.7 MESHALYZER

Analyzes input mesh file or directory of mesh files and writes results to STDERR. If a directory is passed as input, the cumulative results for the set of input mesh files are aggregated and returned along with results for each single mesh. Written in C++. See Chapter III for more details.

E.8 meshalyzerxxl

Version of MESHALYZER using the STXXL library which is an out-of-core implementation of STL. The full functionality of MESHALYZER has not been added to meshalyzerxxl yet. Execution speed benchmarks have not been run comparing MESHALYZER to meshalyzerxxl. Results written to STDOUT. Written in C++.

E.9 mesh_tools

The most important programs in the mesh_tools suite have been mentioned by name throughout this document, e.g. MESHALYZER, MESHMORPH,

FILTERMESH, etc. Here we describe other tools that are useful for file format conversions and specific mesh manipulation tasks.

contour_plotting

Extracts contours from a single input mesh file or from all mesh files in an input directory. The contours are extracted from the mesh at a user-defined z value by identifying faces with two vertices at the specified z value. Valid input meshes include raw output from CONTOUR_TILER (which will have stratified faces) and output meshes from an appropriate CSG operation using IRIT, e.g. subtract_mesh using a box with one box face at the specified z value. Face vertices at the specified z value are collected and printed in RECONSTRUCT3D contour format to STDOUT. Written in C++.

mesh2mcell

Convert mesh file format to MDL file format. Requires that an MDL object name be chosen. Output mesh written to STDOUT. Written in C.

mesh2off

Convert mesh file format to DEC object file format. Output mesh written to STDOUT. Written in C.

mesh2rib

Convert mesh file format to RenderMan Interface Bytestream format, which is a scene description language. Output mesh written to STDOUT. Written in C.

mesh2ribwireframe

Convert mesh file format to RenderMan Interface Bytestream format. Generates edges as cylinders and vertices as spheres. Scene description language.

Output mesh written to STDOUT. Written in C.

mesh2smesh

Convert mesh file format to TetGen's surface mesh format. Output mesh written to STDOUT. Written in C.

mesh2smf

Convert mesh file format to smf mesh format. QHull? Mesh decimator? Output mesh written to STDOUT. Written in C.

mesh2stl

Convert mesh file format to STL. Output mesh written to STDOUT. Written in C.

mesh2vtk

Convert mesh file format to VTK. Output mesh written to STDOUT. Written in C.

meshclip

Separate one input mesh file into three categories - faces and associated vertices that are (1) fully-inside, (2) fully-outside, and (3) intersect a second input mesh. The faces and associated vertices are written to files - fully_inside.mesh, fully_outside.mesh, and on_edge.mesh, respectively. Vertices are duplicated when shared by faces in different categories. Currently, vertex and face indices are renumbered after categorization. Should add option to preserve original indexing. Note the directions of the face normals in the two input meshes are important. Written in C.

meshfilter

Experimental tool for classifying faces based on normal orientation. Not used much.

meshflip

The normal vector of every face in the input mesh file is flipped by re-arranging the sequence of vertex indices for each face. Output mesh written to STDOUT. Written in C++.

meshmerge

Merge two input mesh files into a single mesh file. No analysis is performed at all. Output mesh written to STDOUT. Written in C.

meshoffset

Offset vertices in input mesh file along average vertex normal. Output mesh written to STDOUT. Written in C.

meshrefine

Given an input mesh file, faces are subdivided so that no edge is longer than a user-defined threshold edge length. If the threshold is negative then each edge is bisected exactly once if edge length is longer than the absolute value of the threshold. New mesh is written to STDOUT. Input mesh must be fully closed, consistently oriented with vertex and face indices sequentially numbered. Written in C++.

meshrefinexxl

Version of meshrefine using the STXXL library which is an out-of-core implementation of STL. This version of meshrefine does not accept edge length

thresholds less than 0. Execution speed benchmarks have not been run comparing meshrefine to meshrefinexxl. Results written to STDOUT. Written in C++.

mesh_renumber

Renumber indices of vertices and faces of input mesh file to be continuous. Output mesh written to STDOUT. Written in C++.

meshscale

Scale input mesh by multiplying vertex locations by x, y, and z user-specified scaling factors. Output mesh written to STDOUT. Written in C.

mesh_separate

Separate input mesh into discontiguous pieces. Writes separate mesh pieces to current directory where each file name is the input mesh file name with an index suffix. The separate pieces are sorted by size so that small indices imply larger meshes. Written in C++.

meshsimplify

Vertices are merged in the input mesh file so that no edge is shorter than user-specified threshold length. Output mesh written to STDOUT. Written in C++.

meshstitch

Used to stitch together a pair of meshes that were created in such a way that both meshes share a set of vertices. Proven applications include stitching spines onto a dendritic shaft and stitching astrocyte pieces together. This program constitutes an ugly workaround for one of the horrible consequence of combining 50 nm neuropil sectioning with CONTOUR_TILER's strict policy of merging overlapping contours. In both of the example applications mentioned before, the offending

contours were meshed separately with the important detail that one contour was duplicated and included in both pieces of the object. This common contour is used to locate where the two meshed pieces can be stitched together. Meshstitch takes as arguments two input mesh file, the z location of the common contour, and the location of the caps that must be removed in the stitching process. Output mesh written to STDOUT. Written in C++.

meshtranslate

Translate input mesh by x, y, and z user-specified translation distances. Output mesh written to STDOUT. Written in C.

remove_duplicate_vertices

Removes vertices from an input mesh file that are not distinguishable at the 1E-10 level in x, y, and z locations. Before using remove_duplicate_vertices, first run MESHALYZER with -p option on the input mesh file to generate meshalyzer_output_file. The meshalyzer_output_file is then scanned for keyphrase "# indistinguishable vertices: vertex " after which follows a list of indistinguishable vertices in input_mesh_file. For each indistinguishable pair of vertices, all instances of the first vertex are replaced with the second vertex. The new mesh file is written to STDOUT. Note that indistinguishable vertices are assumed to come in pairs, and three or more indistinguishable vertices will not be handled appropriately. Written in C++.

simplify_surface

Perl script that calls the DX simplify surface algorithm to remesh an input mesh in DX file format. Note nonstandard syntax: 'simplify_surface object.mesh object.dx', but necessary since DX mesh is written in binary format. The Perl script must be edited to hand to configure the seven arguments to the DX algorithm.

stl2mesh

Convert input STL file into mesh file format. Merges vertices if they are indistinguishable at the 1E-12 level. In other words, the x, y, and z locations all differ by less than one part per trillion. Output mesh written to STDOUT. Written in C.

stlb2stla

Convert input STL binary file to STL ASCII file format. Output data written to STDOUT. Written in C.

synu2mesh

Convert input file in NCMIR's SYNU file format to mesh file format. Note that XVoxTrace can output SYNU format. Output data written to STDOUT. Written in C.

vizvtk

Tcl/Tk program to visualize vtk format meshes. Pain to install VTK (heavy-weight and difficult to install), but has useful mesh manipulation algorithms.

vrml2mesh

Convert input file in Virtual Reality Markup Language file format to mesh file format. Output data written to STDOUT. Written in C.

vtk2mesh

Convert input file in VTK file format to mesh file format. Output data written to STDOUT. Written in C.

vtk2rib

Convert input file in VTK file format to RenderMan Interface Bytestream format. Output data written to STDOUT. Written in C.

vtk_march

Tcl pipeline for using VTK's isosurface extraction algorithm (marching cubes) on an image stack.

E.10 MESHMORPH

Given a directory of input mesh files, MESHMORPH computes the potential energy of the mesh set and then moves vertices to lower the energy of the system. The final state of the mesh set is influenced by user-specified weights which determine the forces (1) between objects, (2) between vertices of the same object, (3) between adjacent faces of edges, and (4) between intersecting faces. The program is intended to compensate for errors in the mesh generation pipeline (including loss of extracellular space due to fixation, segmentation errors, and tiling artifacts as a by-product of course z sampling) by recovering the appropriate extracellular volume fraction in a neuropil mesh set. Morphed versions of the input meshes are written to user-specified output directory along with several diagnostic files. This program more than any other mesh tool requires detailed configuration that is mostly guided by expert knowledge gained from extensive user experience. Written in C++. See Section II.1.F for detailed account of MESHMORPH design and usage and type 'meshmorph -h' on the command line for a lengthy list of program options.

measure_ecw

Measures the distance between objects in an input directory of mesh files. The surface sampling density for measuring inter-object distance has an area equal

to an equilateral triangle of side sampling_length, where sampling_length is user-specified. Inter-object distances are written to STDOUT. Assumes face normals point outwards. Written in C++. As the name implies, the original application was measuring the extracellular width in the CA1 mesh set.

E.11 Netgen

3D volumetric mesh generator. Pros - treats input as geometry and generates beautiful surface meshes on geometry. Cons - volumetric meshes are not Delaunay. Ideally, use Tetgen to generate volumetric mesh from Netgen surface mesh output. Source code can be found at /home/jkinney/src/netgen/. The latest version can always be downloaded from the Netgen CVS repository (`www.hpfem.jku.at/netgen/`); the password is "Dreieckerl". Written in C++.

netgen2mesh

Convert input file in Netgen surface file format to mesh format. Output data written to STDOUT. Written in C.

netgen2smesh

Convert input file in Netgen surface file format to TetGen surface file format. Output data written to STDOUT. Written in C.

E.12 TetGen

3D volumetric mesh generator. Pros - generates beautiful Delaunay volumetric meshes given high-quality surface meshes. Cons - degrades quality of surface mesh under some circumstances and will never generate a high quality surface mesh from a poor quality input surface mesh. Ideally, use Netgen surface mesh as input.

E.13 GNU Triangulated Surface Library

Good = not STL-like. Bad (but unavoidable) = potential for arbitrarily small edges from CSG operations. Should investigate further GTS algorithms, but at least CSG looks promising. Note input format is unusual.

E.14 Blender

Blender is an open-source program that supports many aspects of 3D modeling including parametric design, mesh editing, rendering, and animation. The user interface and ease of use are compelling. The fact that custom scripts are readily created using Python positions Blender as the future hub of mesh generation. However, some strategic planning are warranted for two reasons. First, a new version will be available and soon and may entail fundamental changes to the structure of Blender. Second, having our mesh tools as scripts that run as Blender plugins is attractive; however, preserving the current command line batch processing functionality of the tools is important as well. Ideally, we can design an architecture whereby both needs are fulfilled.

References

Aarts, M., Liu, Y., Liu, L., Besshoh, S., Arundine, M., Gurd, J. W., Wang, Y. T., Salter, M. W., and Tymianski, M., 2002: Treatment of ischemic brain damage by perturbing nmda receptor- psd-95 protein interactions. *Science*, **298**(5594), 846–850.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and P, W., editors, 2002: *Molecular Biology of the Cell*. Garland Publishing, 4th edition.

Angulo, M. C., Kozlov, A. S., Charpak, S., and Audinat, E., 2004: Glutamate released from glial cells synchronizes neuronal activity in the hippocampus. *J Neurosci*, **24**(31), 6920–6927.

Arnth-Jensen, N., Jabaudon, D., and Scanziani, M., 2002: Cooperation between independent hippocampal synapses is controlled by glutamate uptake. *Nat Neurosci*, **5**(4), 325–331.

Arriza, J. L., Fairman, W. A., Wadiche, J. I., Murdoch, G. H., Kavanaugh, M. P., and Amara, S. G., 1994: Functional comparisons of three glutamate transporter subtypes cloned from human motor cortex. *J Neurosci*, **14**(9), 5559–5569.

Asztely, F., Erdemli, G., and Kullmann, D., 1997: Extrasynaptic glutamate spillover in the hippocampus: dependence on temperature and the role of active glutamate uptake. *Neuron*, **18**(2), 281–293.

Bajaj, C. L., Coyle, E. J., and Lin, K.-N., 1996: Arbitrary topology shape reconstruction from planar cross sections. *Graph. Models Image Process.*, **58**(6), 524–543. ISSN 1077-3169.

Banke, T. G., and Traynelis, S. F., 2003: Activation of nr1/nr2b nmda receptors. *Nat Neurosci*, **6**(2), 144–152.

Barbour, B., 2001: An evaluation of synapse independence. *J Neurosci*, **21**(20), 7969–7984.

Barbour, B., and Häusser, M., 1997: Intersynaptic diffusion of neurotransmitter. *Trends Neurosci*, **20**(9), 377–384.

Bellinger, S. C., Miyazawa, G., and Steinmetz, P. N., 2008: Submyelin potassium accumulation may functionally block subsets of local axons during deep brain stimulation: a modeling study. *J Neural Eng*, **5**, 263–274.

Bergles, D. E., Tzingounis, A. V., and Jahr, C. E., 2002: Comparison of coupled and uncoupled currents during glutamate uptake by glt-1 transporters. *J Neurosci*, **22**(23), 10153–10162.

Biró, A. A., and Nusser, Z., 2005: Synapse independence breaks down during highly synchronous network activity in the rat hippocampus. *Eur J Neurosci*, **22**(5), 1257–1262.

Bourne, J. N., and Harris, K. M., 2008: Balancing structure and function at hippocampal dendritic spines. *Annu. Rev. Neurosci.*, **31**, 47–67.

Bouvier, M., Szatkowski, M., Amato, A., and Attwell, D., 1992: The glial cell glutamate uptake carrier countertransports ph-changing anions. *Nature*, **360**(6403), 471–474.

Burger, P. M., Mehl, E., Cameron, P. L., Maycox, P. R., Baumert, M., Lottspeich, F., De Camilli, P., and Jahn, R., 1989: Synaptic vesicles immunoisolated from rat cerebral cortex contain high levels of glutamate. *Neuron*, **3**(6), 715–720.

Bushong, E. A., Martone, M. E., Jones, Y. Z., and Ellisman, M. H., 2002: Protoplasmic astrocytes in ca1 stratum radiatum occupy separate anatomical domains. *J Neurosci*, **22**(1), 183–192.

Buzsáki, G., 2002: Theta oscillations in the hippocampus. *Neuron*, **33**(3), 325–340.

Chaudhry, F. A., Lehre, K. P., van Lookeren Campagne, M., Ottersen, O. P., Danbolt, N. C., and Storm-Mathisen, J., 1995: Glutamate transporters in glial plasma membranes: highly differentiated localizations revealed by quantitative ultrastructural immunocytochemistry. *Neuron*, **15**(3), 711–720.

Chen, S., and Diamond, J. S., 2002: Synaptically released glutamate activates extrasynaptic NMDA receptors on cells in the ganglion cell layer of rat retina. *J. Neurosci.*, **22**, 2165–2173.

Chklovskii, D. B., Schikorski, T., and Stevens, C. F., 2002: Wiring optimization in cortical circuits. *Neuron*, **34**(3), 341–347.

Clements, J. D., Lester, R. A., Tong, G., Jahr, C. E., and Westbrook, G. L., 1992: The time course of glutamate in the synaptic cleft. *Science*, **258**(5087), 1498–1501.

Coggan, J. S., Bartol, T. M., Esquenazi, E., Stiles, J. R., Lamont, S., Martone, M. E., Berg, D. K., Ellisman, M. H., and Sejnowski, T. J., 2005: Evidence for ectopic neurotransmission at a neuronal synapse. *Science*, **309**(5733), 446–451.

Conti, F., DeBiasi, S., Minelli, A., Rothstein, J. D., and Melone, M., 1998: Eaac1, a high-affinity glutamate tranporter, is localized to astrocytes and gabaergic neurons besides pyramidal cells in the rat cerebral cortex. *Cereb Cortex*, **8**(2), 108–116.

Conti, R., and Lisman, J., 2003: The high variance of ampa receptor- and nmda receptor-mediated responses at single hippocampal synapses: evidence for multiquantal release. *Proc Natl Acad Sci U S A*, **100**(8), 4885–4890.

Danbolt, N. C., 2001: Glutamate uptake. *Prog Neurobiol*, **65**(1), 1–105.

Diamond, J., 2001: Neuronal glutamate transporters limit activation of nmda receptors by neurotransmitter spillover on ca1 pyramidal cells. *J Neurosci*, **21**(21), 8328–8338.

Diamond, J. S., and Jahr, C. E., 1997: Transporters buffer synaptically released glutamate on a submillisecond time scale. *J Neurosci*, **17**(12), 4672–4687.

Eskandari, S., Kreman, M., Kavanaugh, M. P., Wright, E. M., and Zampighi, G. A., 2000: Pentameric assembly of a neuronal glutamate transporter. *Proc Natl Acad Sci U S A*, **97**(15), 8641–8646.

Fellin, T., Pascual, O., Gobbo, S., Pozzan, T., Haydon, P. G., and Carmignoto, G., 2004: Neuronal synchrony mediated by astrocytic glutamate through activation of extrasynaptic nmda receptors. *Neuron*, **43**(5), 729–743.

Fiala, J. C., 2005: Reconstruct: a free editor for serial section microscopy. *J Microsc*, **218**, 52–61.

Forti, L., Bossi, M., Bergamaschi, A., Villa, A., and Malgaroli, A., 1997: Loose-patch recordings of single quanta at individual hippocampal synapses. *Nature*, **388**(6645), 874–878.

Fox, C. H., Johnson, F. B., Whiting, J., Roller, P. P., and Blum, F., 1985: Formaldehyde fixation. *J. Histochem. Cytochem.*, **33**, 845–853.

Franks, K. M., Bartol, T. M., and Sejnowski, T. J., 2002: A monte carlo model reveals independent signaling at central glutamatergic synapses. *Biophys J*, **83**(5), 2333–2348.

Franks, K. M., Stevens, C. F., and Sejnowski, T. J., 2003: Independent sources of quantal variability at single glutamatergic synapses. *J Neurosci*, **23**(8), 3186–3195.

Ganeshina, O., Berry, R. W., Petralia, R. S., Nicholson, D. A., and Geinisman, Y., 2004a: Differences in the expression of ampa and nmda receptors between axospinous perforated and nonperforated synapses are related to the configuration and size of postsynaptic densities. *J Comp Neurol*, **468**(1), 86–95.

Ganeshina, O., Berry, R. W., Petralia, R. S., Nicholson, D. A., and Geinisman, Y., 2004b: Synapses with a segmented, completely partitioned postsynaptic density express more ampa receptors than other axospinous synaptic junctions. *Neuroscience*, **125**(3), 615–623.

Grebenyuk, S. E., Lozovaya, N. A., Tsintsadze, T. S., and Krishtal, O. A., 2004: Post-synaptic n-methyl-d-aspartate signalling in hippocampal neurons of rat: spillover increases the impact of each spike in a short burst discharge. *Neurosci Lett*, **361**(1-3), 60–63.

Hamberger, A., and Nyström, B., 1984: Extra- and intracellular amino acids in the hippocampus during development of hepatic encephalopathy. *Neurochem Res*, **9**(9), 1181–1192.

Hardingham, G. E., Fukunaga, Y., and Bading, H., 2002: Extrasynaptic nmdars oppose synaptic nmdars by triggering creb shut-off and cell death pathways. *Nat Neurosci*, **5**(5), 405–414.

Harris, K. M., Perry, E., Bourne, J., Feinberg, M., Ostroff, L., and Hurlburt, J., 2006: Uniform serial sectioning for transmission electron microscopy. *J. Neurosci.*, **26**, 12101–12103.

Harris, K. M., and Stevens, J. K., 1989: Dendritic spines of ca 1 pyramidal cells in the rat hippocampus: serial electron microscopy with reference to their biophysical characteristics. *J Neurosci*, **9**(8), 2982–2997.

Hazell, A. S., Butterworth, R. F., and Hakim, A. M., 1993: Cerebral vulnerability is associated with selective increase in extracellular glutamate concentration in experimental thiamine deficiency. *J Neurochem*, **61**(3), 1155–1158.

He, Y., Janssen, W. G., Rothstein, J. D., and Morrison, J. H., 2000: Differential synaptic localization of the glutamate transporter eaac1 and glutamate receptor subunit glur2 in the rat hippocampus. *J Comp Neurol*, **418**(3), 255–269.

Herman, M. A., and Jahr, C. E., 2007: Extracellular glutamate concentration in hippocampal slice. *J. Neurosci.*, **27**, 9736–9741.

Hochmuth, R. M., Evans, C. A., Wiles, H. C., and McCown, J. T., 1983: Mechanical measurement of red cell membrane thickness. *Science*, **220**, 101–102.

Hodgkin, A. L., and Huxley, A. F., 1952: A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, **117**, 500–544.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1993: Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 19–26. ACM, New York, NY, USA.

Hrabetová, S., and Nicholson, C., 2000: Dextran decreases extracellular tortuosity in thick-slice ischemia model. *J Cereb Blood Flow Metab*, **20**(9), 1306–1310.

Hrabetová, S., and Nicholson, C., 2004: Contribution of dead-space microdomains to tortuosity of brain extracellular space. *Neurochem Int*, **45**(4), 467–477.

Ingber, D. E., Dike, L., Hansen, L., Karp, S., Liley, H., Maniotis, A., McNamee, H., Mooney, D., Plopper, G., Sims, J., and Ingber, D. E., 1994: Cellular tensegrity: exploring how mechanical changes in the cytoskeleton regulate cell growth, migration, and tissue pattern during morphogenesis. *Int. Rev. Cytol.*, **150**, 173–224.

Jensen, F. E., and Harris, K. M., 1989: Preservation of neuronal ultrastructure in hippocampal slices using rapid microwave-enhanced fixation. *J. Neurosci. Methods*, **29**, 217–230.

Jonas, P., Major, G., and Sakmann, B., 1993: Quantal components of unitary epscs at the mossy fibre synapse on ca3 pyramidal cells of rat hippocampus. *J Physiol*, **472**, 615–663.

Kirkpatrick, S., 1997: Optimization by simulated annealing: Quantitative studies. *J Stat Phys*, **34**, 975–986.

Kirov, S. A., Sorra, K. E., and Harris, K. M., 1999: Slices have more synapses than perfusion-fixed hippocampus from both young and mature rats. *J. Neurosci.*, **19**, 2876–2886.

Kullmann, D. M., Min, M. Y., Asztely, F., and Rusakov, D. A., 1999: Extracellular glutamate diffusion determines the occupancy of glutamate receptors at ca1 synapses in the hippocampus. *Philos Trans R Soc Lond B Biol Sci*, **354**(1381), 395–402.

Latefi, N. S., and Colman, D. R., 2007: The CNS synapse revisited: gaps, adhesive welds, and borders. *Neurochem. Res.*, **32**, 303–310.

Lehre, K. P., and Danbolt, N. C., 1998: The number of glutamate transporter subtype molecules at glutamatergic synapses: chemical and stereological quantification in young adult rat brain. *J Neurosci*, **18**(21), 8751–8757.

Lehre, K. P., and Rusakov, D. A., 2002: Asymmetry of glia near central synapses favors presynaptically directed glutamate escape. *Biophys J*, **83**(1), 125–134.

Li, B., Chen, N., Luo, T., Otsu, Y., Murphy, T. H., and Raymond, L. A., 2002: Differential regulation of synaptic and extra-synaptic nmda receptors. *Nat Neurosci*, **5**(9), 833–834.

Liao, D., Hessler, N. A., and Malinow, R., 1995: Activation of postsynaptically silent synapses during pairing-induced ltp in ca1 region of hippocampal slice. *Nature*, **375**(6530), 400–404.

Lin, Y., Skeberdis, V. A., Francesconi, A., Bennett, M. V., and Zukin, R. S., 2004: Postsynaptic density protein-95 regulates nmda channel gating and surface expression. *J Neurosci*, **24**(45), 10138–10148.

Lis, L. J., McAlister, M., Fuller, N., Rand, R. P., and Parsegian, V. A., 1982: Measurement of the lateral compressibility of several phospholipid bilayers. *Biophys. J.*, **37**, 667–672.

Longsworth, L. G., 1953: Diffusion measurements at 25 degrees celcius of aqueous solutions of amino acids, peptides and sugars. *J Am Chem Soc*, **75**(22), 5705–5709.

Longuemare, M. C., Rose, C. R., Farrell, K., Ransom, B. R., Waxman, S. G., and Swanson, R. A., 1999: K(+)-induced reversal of astrocyte glutamate uptake is limited by compensatory changes in intracellular na+. *Neuroscience*, **93**(1), 285–292.

Lorensen, W. E., and Cline, H. E., 1987: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 163–169. ACM, New York, NY, USA.

Lozovaya, N. A., Kopanitsa, M. V., Boychuk, Y. A., and Krishtal, O. A., 1999: Enhancement of glutamate release uncovers spillover-mediated transmission by n-methyl-d-aspartate receptors in the rat hippocampus. *Neuroscience*, **91**(4), 1321–1330.

Mainen, Z. F., Maletic-Savatic, M., Shi, S. H., Hayashi, Y., Malinow, R., and Svoboda, K., 1999: Two-photon imaging in living brain slices. *Methods*, **18**, 231–239.

Mazel, T., Simonová, Z., and Syková, E., 1998: Diffusion heterogeneity and anisotropy in rat hippocampus. *Neuroreport*, **9**(7), 1299–1304.

McAllister, A. K., and Stevens, C. F., 2000: Nonsaturation of ampa and nmda receptors at hippocampal synapses. *Proc Natl Acad Sci U S A*, **97**(11), 6173–6178.

McBain, C. J., Traynelis, S. F., and Dingledine, R., 1990: Regional variation of extracellular space in the hippocampus. *Science*, **249**(4969), 674–677.

Mohrmann, R., Köhr, G., Hatt, H., Sprengel, R., and Gottmann, K., 2002: Deletion of the c-terminal domain of the nr2b subunit alters channel properties

and synaptic targeting of n-methyl-d-aspartate receptors in nascent neocortical synapses. *J Neurosci Res*, **68**(3), 265–275.

Monyer, H., Burnashev, N., Laurie, D. J., Sakmann, B., and Seeburg, P. H., 1994: Developmental and regional expression in the rat brain and functional properties of four nmda receptors. *Neuron*, **12**(3), 529–540.

Moody, W. J., Futamachi, K. J., and Prince, D. A., 1974: Extracellular potassium activity during epileptogenesis. *Exp. Neurol.*, **42**, 248–263.

Nicholson, C., and Phillips, J. M., 1981: Ion diffusion modified by tortuosity and volume fraction in the extracellular microenvironment of the rat cerebellum. *J Physiol*, **321**, 225–257.

Nielsen, T. A., DiGregorio, D. A., and Silver, R. A., 2004: Modulation of glutamate mobility reveals the mechanism underlying slow-rising ampar epscs and the diffusion coefficient in the synaptic cleft. *Neuron*, **42**(5), 757–771.

Nimchinsky, E. A., Yasuda, R., Oertner, T. G., and Svoboda, K., 2004: The number of glutamate receptors opened by synaptic stimulation in single hippocampal spines. *J Neurosci*, **24**(8), 2054–2064.

Noguchi, J., Matsuzaki, M., Ellis-Davies, G. C., and Kasai, H., 2005: Spine-neck geometry determines nmda receptor-dependent ca2+ signaling in dendrites. *Neuron*, **46**(4), 609–622.

Nowak, L., Bregestovski, P., Ascher, P., Herbet, A., and Prochiantz, A., 1984: Magnesium gates glutamate-activated channels in mouse central neurones. *Nature*, **307**(5950), 462–465.

Oertner, T. G., Sabatini, B. L., Nimchinsky, E. A., and Svoboda, K., 2002: Facilitation at single synapses probed with optical quantal analysis. *Nat Neurosci*, **5**(7), 657–664.

Ohno, N., Terada, N., Saitoh, S., and Ohno, S., 2007: Extracellular space in mouse cerebellar cortex revealed by in vivo cryotechnique. *J. Comp. Neurol.*, **505**, 292–301.

Otis, T. S., and Jahr, C. E., 1998: Anion currents and predicted glutamate flux through a neuronal glutamate transporter. *J Neurosci*, **18**(18), 7099–7110.

Parpura, V., Basarsky, T. A., Liu, F., Jeftinija, K., Jeftinija, S., and Haydon, P. G., 1994: Glutamate-mediated astrocyte-neuron signalling. *Nature*, **369**(6483), 744–747.

Patneau, D. K., and Mayer, M. L., 1990: Structure-activity relationships for amino acid transmitter candidates acting at N-methyl-D-aspartate and quisqualate receptors. *J. Neurosci.*, **10**, 2385–2399.

Petrak, L. J., Harris, K. M., and Kirov, S. A., 2005: Synaptogenesis on mature hippocampal dendrites occurs via filopodia and immature spines during blocked synaptic transmission. *J. Comp. Neurol.*, **484**, 183–190.

Popescu, G., and Auerbach, A., 2003: Modal gating of nmda receptors and the shape of their synaptic response. *Nat Neurosci*, **6**(5), 476–483.

Racca, C., Stephenson, F. A., Streit, P., Roberts, J. D., and Somogyi, P., 2000: Nmda receptor content of synapses in stratum radiatum of the hippocampal ca1 area. *J Neurosci*, **20**(7), 2512–2522.

Rothstein, J. D., Martin, L., Levey, A. I., Dykes-Hoberg, M., Jin, L., Wu, D., Nash, N., and Kuncl, R. W., 1994: Localization of neuronal and glial glutamate transporters. *Neuron*, **13**(3), 713–725.

Rumbaugh, G., and Vicini, S., 1999: Distinct synaptic and extrasynaptic nmda receptors in developing cerebellar granule neurons. *J Neurosci*, **19**(24), 10603–10610.

Rusakov, D. A., 2001: The role of perisynaptic glial sheaths in glutamate spillover and extracellular ca(2+) depletion. *Biophys J*, **81**(4), 1947–1959.

Rusakov, D. A., and Kullmann, D. M., 1998a: Extrasynaptic glutamate diffusion in the hippocampus: ultrastructural constraints, uptake, and receptor activation. *J Neurosci*, **18**(9), 3158–3170.

Rusakov, D. A., and Kullmann, D. M., 1998b: Geometric and viscous components of the tortuosity of the extracellular space in the brain. *Proc Natl Acad Sci U S A*, **95**(15), 8975–8980.

Rusakov, D. A., Kullmann, D. M., and Stewart, M. G., 1999: Hippocampal synapses: do they talk to their neighbours? *Trends Neurosci*, **22**(9), 382–388.

Sabouri-Ghomi, M., Wu, Y., Hahn, K., and Danuser, G., 2008: Visualizing and quantifying adhesive signals. *Curr. Opin. Cell Biol.*, **20**, 541–550.

Sattler, R., Xiong, Z., Lu, W. Y., MacDonald, J. F., and Tymianski, M., 2000: Distinct roles of synaptic and extrasynaptic nmda receptors in excitotoxicity. *J Neurosci*, **20**(1), 22–33.

Savtchenko, L. P., and Rusakov, D. A., 2004: Glutamate escape from a tortuous synaptic cleft of the hippocampal mossy fibre synapse. *Neurochem. Int.*, **45**, 479–484.

Scanziani, M., Salin, P. A., Vogt, K. E., Malenka, R. C., and Nicoll, R. A., 1997: Use-dependent increases in glutamate concentration activate presynaptic metabotropic glutamate receptors. *Nature*, **385**, 630–634.

Schoberl, J., 1997: NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Comput Vis Sci*, **1**, 41–52.

Schuz, A., and Palm, G., 1989: Density of neurons and synapses in the cerebral cortex of the mouse. *J. Comp. Neurol.*, **286**, 442–455.

Sobczyk, A., Scheuss, V., and Svoboda, K., 2005: Nmda receptor subunit-dependent [ca2+] signaling in individual hippocampal dendritic spines. *J Neurosci*, **25**(26), 6037–6046.

Spruston, N., Jonas, P., and Sakmann, B., 1995: Dendritic glutamate receptor channels in rat hippocampal ca3 and ca1 pyramidal neurons. *J Physiol*, **482 (Pt 2)**, 325–352.

Stiles, J., and Bartol, T. J., 2001: *Computational Neuroscience Realistic Modeling for Experimentalists*, chapter Monte Carlo methods for simulating realistic synaptic microphysiology, 681–731. CRC Press.

Stiles, J. R., Van Helden, D., Bartol, T. M., Salpeter, E. E., and Salpeter, M. M., 1996: Miniature endplate current rise times less than 100 microseconds from improved dual recordings can be modeled with passive acetylcholine diffusion from a synaptic vesicle. *Proc Natl Acad Sci U S A*, **93**(12), 5747–5752.

Stocca, G., and Vicini, S., 1998: Increased contribution of nr2a subunit to synaptic nmda receptors in developing rat cortical neurons. *J Physiol*, **507 (Pt 1)**, 13–24.

Sykova, E., 1997: The extracellular space in the cns: Its regulation, volume and geometry in normal and pathological neuronal function. *Neuroscientist*, **3**(1), 28–41.

Sykov, E., and Nicholson, C., 2008: Diffusion in brain extracellular space. *Physiol. Rev.*, **88**, 1277–1340.

Takahashi, M., Sarantis, M., and Attwell, D., 1996: Postsynaptic glutamate uptake in rat cerebellar purkinje cells. *J Physiol*, **497 (Pt 2)**, 523–530.

Takumi, Y., Ramírez-León, V., Laake, P., Rinvik, E., and Ottersen, O. P., 1999: Different modes of expression of ampa and nmda receptors in hippocampal synapses. *Nat Neurosci*, **2**(7), 618–624.

Tao, L., and Nicholson, C., 2004: Maximum geometrical hindrance to diffusion in brain extracellular space surrounding uniformly spaced convex cells. *J Theor Biol*, **229**(1), 59–68.

Thompson, C. L., Drewery, D. L., Atkins, H. D., Stephenson, F. A., and Chazot, P. L., 2002: Immunohistochemical localization of n-methyl-d-aspartate receptor subunits in the adult murine hippocampal formation: evidence for a unique role of the nr2d subunit. *Brain Res Mol Brain Res*, **102**(1-2), 55–61.

Thorne, R. G., and Nicholson, C., 2006: In vivo diffusion analysis with quantum dots and dextrans predicts the width of brain extracellular space. *Proc. Natl. Acad. Sci. U.S.A.*, **103**, 5567–5572.

Tong, G., and Jahr, C. E., 1994: Block of glutamate transporters potentiates postsynaptic excitation. *Neuron*, **13**(5), 1195–1203.

Tovar, K. R., and Westbrook, G. L., 1999: The incorporation of nmda receptors with a distinct subunit composition at nascent hippocampal synapses in vitro. *J Neurosci*, **19**(10), 4180–4188.

van Harreveld, A., Crowell, J., and Malhotra, S. K., 1965: A Study Of Extracellular Space In Central Nervous Tssue By Freeze-Substitution. *J. Cell Biol.*, **25**, 117–137.

Ventura, R., and Harris, K. M., 1999: Three-dimensional relationships between hippocampal synapses and astrocytes. *J Neurosci*, **19**(16), 6897–6906.

Volterra, A., and Meldolesi, J., 2005: Astrocytes, from brain glue to communication elements: the revolution continues. *Nat Rev Neurosci*, **6**(8), 626–640.

Volterra, A., and Steinhäuser, C., 2004: Glial modulation of synaptic transmission in the hippocampus. *Glia*, **47**(3), 249–257.

Wadiche, J. I., Arriza, J. L., Amara, S. G., and Kavanaugh, M. P., 1995: Kinetics of a human glutamate transporter. *Neuron*, **14**(5), 1019–1027.

Wadiche, J. I., and Kavanaugh, M. P., 1998: Macroscopic and microscopic properties of a cloned glutamate transporter/chloride channel. *J Neurosci*, **18**(19), 7650–7661.

Wahl, L. M., Pouzat, C., and Stratford, K. J., 1996: Monte carlo simulation of fast excitatory synaptic transmission at a hippocampal synapse. *J Neurophysiol*, **75**(2), 597–608.

Wolfe, J., and Bryant, G., 1999: Freezing, drying, and/or vitrification of membrane- solute-water systems. *Cryobiology*, **39**, 103–129.

Xu, H., Othman, S. F., and Magin, R. L., 2008: Monitoring tissue engineering using magnetic resonance imaging. *J. Biosci. Bioeng.*, **106**, 515–527.

Zhang, S., Takeda, Y., Hagioka, S., Takata, K., Aoe, H., Nakatsuka, H., Yokoyama, M., and Morita, K., 2005: Measurement of gaba and glutamate in vivo levels with high sensitivity and frequency. *Brain Res Brain Res Protoc*, **14**(2), 61–66.