Major Project - CO499 : Implementing 3D regions defined by Arbitrary Crosssections with discontinuous boundary elements, on Neuroconstruct

Sumukha R M , Abhishek Reddy Y N, Job J

12CO67 ,12CO03 ,12CO34

18/04/2016

National Institute of Technology Karnataka, Surathkal

Karnataka, India - 575025 (www.nitk.ac.in)

To

Basavaraj Talawar

Department of Computer Science and Engineering,

National Institute of Technology Karnataka, Surathkal

Certificate

This is to certify that the project entitled **Implementing 3D regions defined by Arbitrary Crosssections with discontinuous boundary elements, on Neuroconstruct** has been completed by,

Sumukha R M (12CO67),
Abhishek Reddy Y N (12CO03),
Job J (12CO34)

final year students of the **Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal**, in the month of April 2016, during the Even Semester of the academic year 2015 - 2016, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Place : NITK Surathkal

Date : 18/04/2016 **Signature of Instructor**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Major Project -II

End Semester Evaluation Report, April 2016

Course code : CO499

Course Title: Major Project -II

Title: *Implementing 3D regions defined by Arbitrary Crosssections with discontinuous boundary elements, on Neuroconstruct*

Project Group:

Name of the Student	Register No.	Signature with Date
Sumukha R M	12CO67	
Abhishek Reddy Y N	12CO03	
Job J	12CO34	

Place:

Date: Name and Signature of Project Guide

List of Figures

Contents

Abstract

In this project, we analyse biologically significant images of arbitrary Cross sections, like Spinal cord cross section, and we implement this functionality in NeuroConstruct which is a tool to develop various neuronal models. This arbitrary cross sectioned model is itself a new neuronal contribution we make at the end of our project. .

Keywords:

Neuroconstruct, Arbitrary crosssection, Neuron, Classes, Region, Model

1 Introduction

The complex geometrcal patterns formed by the various components of the brain play a huge role in its information processing capabilities. Even at a neuron level the shape of the dendric tree determines many details of the electrical activity, and those of the synapses determine how signals are integrated. Thus the signal processing carried out by a neuron is determined both by its own morphology and the 3d structure of the network in which it is embedded. Understanding how these low level features of the neurons give rise to higher level features of the brain requires the help of advanced network models. Research has supported a relationship between the morphological and functional properties of neurons. For instance, the accordance between the morphology and the functional classes of cat retinal ganglion cells has been studied to show the relationship between neuron shape and function. Orientation sensitivity and dendritic branching patterns are a few other common characteristics of neurons that researchers have noted as having an effect on neuron function. nueroConstruct allows us to model many of these morphologies and network models and analyse the electrical characteristics and the associated functionalities.It is implemented in Java and generates script files for a number of widely used neuronal simulation platforms. Understanding how complex brain structures and the myriad of underlying mechanisms interact to produce higher-level functions will require the help of network models with biologically realistic features. Models that use compartmental neurons, Hodgkin-Huxley type membrane conductances, and semirealistic synaptic connectivity have been used to explore the potential mechanisms underlying synchronous activity , cortical oscillations , hippocampal memory, and temporal coding . They have also provided insights into potential causes of epileptiform activity in dentate gyrus and cortex. However, virtually all such models uti-

lize simplified synaptic connectivity, featuring abstract neurons connected in either one dimension or two-dimensional layered structures. The development of more biologically realistic network models that include explicit 3D information would allow direct comparison of the model structure with anatomical measurements. Such network models would also allow direct comparison of the spatiotemporal properties of simulated neural activity with experimental measurements using multielectrode recordings or two-photon imaging of activity in blocks of tissue . Moreover, they could be extended to simulate volume signaling and brain metabolism. While the development of such 3D network models is theoretically possible with current simulators such as NEURON and GENESIS, and some preliminary attempts have been made, considerable technical difficulties remain. These include a requirement for algorithms that can create the highly nonuniform 3D synaptic connectivity observed in biological networks, a method for verifying connectivity and routines for analyzing network behavior. Indeed, the absence of such tools has prevented the development and use of more biologically realistic 3D network models. A more integrated understanding of brain function will require closer interaction between experimental and theoretical neuroscientists. At present, communication between these groups, and even between individual theoreticians, is hampered by poor accessibility and interoperability of models. Although single-cell and network models are available on public databases, their utility as research tools is often restricted to those familiar with the specialist scripting languages, which are simulator specific. For example, a neuronal model written in NEURON script cannot be used as part of a GENESIS simulation, thereby limiting its interchange and reuse. Although graphical user interfaces (GUIs) have improved the accessibility of single-cell models, networks remain inaccessible to most neuroscientists. We have developed a new software ap-

plication, neuro- Construct, that facilitates the creation and analysis of networks of multicompartmental neurons in 3D space. Automated cell placement and generation of synaptic connectivity, together with 3D visualization, allow the creation and verification of models with greater anatomical realism than achieved previously with script files. A GUI and the automated generation of code for NEURON or GENESIS allow network models to be built, modified, and run without programming, enhancing their accessibility. Model reuse and interchangeability are facilitated through the implementation of a simulator-independent model description based on NeuroML standards. We describe and test the functionality of neuroConstruct and extend a 1D model of the granule cell layer of the cerebellar cortex to 3D, thereby providing an example of behavior, previously observed in vivo, that could not be captured in the original 1D model. Conductance-based neuronal network models can help us understand how synaptic and cellular mechanisms underlie brain function. However, these complex models are difficult to develop and are inaccessible to most neuroscientists. Moreover, even the most biologically realistic network models disregard many 3D anatomical features of the brain. Here, we describe a new software application, neuroConstruct, that facilitates the creation, visualization, and analysis of networks of multicompartmental neurons in 3D space A graphical user interface allows model generation and modification without programming. Models within neuroConstruct are based on new simulator-independent NeuroML standards, allowing automatic generation of code for NEURON or GENESIS simulators. neuroConstruct was tested by reproducing published models and its simulator independence verified by comparing the same model on two simulators. We show how more anatomically realistic network models can be created and their properties compared with experimental measurements by extending a published 1D cerebellar granule cell layer model

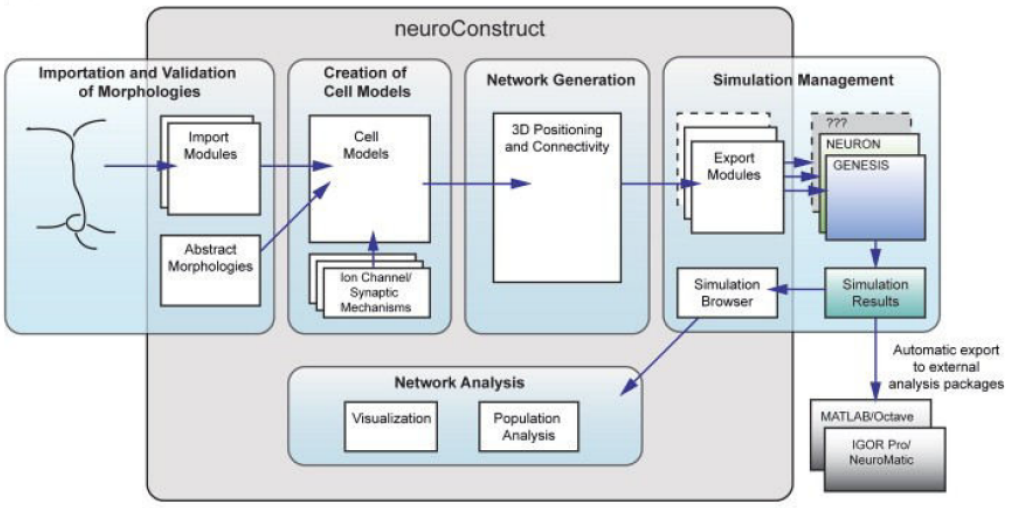

Figure 1: Analysing neuronal models using NeuroConstruct

to 3D.

neuroConstructs functionality can be grouped into five main areas

1. Importation and Validation of Morphologies Reconstructed neuronal morphologies, commonly used in conductance-based neuronal models, can be imported into neuroConstruct in various formats and automatically checked for errors. More abstract morphologies with a smaller number of compartments can also be created manually

2. Creation of Simulator-Independent Conductance-Based Cell Models Modeling of detailed cellular mechanisms, such as the conductance changes produced by voltage- and ligandgated ion channels, is essential for reproducing the complex behavior of real neurons. Cell mechanisms can be defined in neuroConstruct in a simulator-independent format and cell models created by specifying the complement and density of these on the cell membrane (

3. Network Generation Once cell models have been created in neuroConstruct, they can be placed within a region of 3D space at a specified density. Layered structures, such as the cortex, can be created from stacks of contiguous regions. Once the cells are arranged, synaptic connections can be generated according to specified sets of rules

4. Simulation Management Network simulations are carried out by automatically generating script files for the simulator packages NEURON or GENESIS and the results stored in text files.

5. Network Analysis Simulations can be loaded back into neuroConstruct for visualization and analysis. For more specialized analyses, script files are created that allow data to be imported into two common numerical analysis packages.

2 Literature Survey

1. Cell Mechanisms

   Neuronal signaling is mediated by a variety of subcellular, membrane, and synaptic mechanisms. Models of cellular mechanisms can be simple, such as a synaptic conductance waveform, or more complex, like Hodgkin-Huxley type formulations of voltage-gated conductances, which depend on both voltage and time, and their conductance density can be nonuniformly distributed over the cell membrane. Such models form a core part of any conductance based neuronal simulation, but their implementation is one of the more complicated aspects of using existing simulation packages. Although the mathematical framework used to describe such mechanisms (e.g., maximum channel conductance, reversal potential, rate equations) is general and familiar to many neuroscientists, implementation of these in NEURON or GENESIS involves use of a platform-specific programming language. Models of cell mechanisms are implemented in neuro- Construct using a ChannelML-based description, which forms part of level 2 of the NeuroML framework. It consists of an XML file containing the physiological parameters in a structured format that can be validated against a specification, reducing the probability of errors. Information in XML files can easily be transformed into other formats with an XSL (extensible stylesheet language) mapping file (Figure 3). We have created XSL files which map ChannelML descriptions of cell mechanisms onto NMODL format for NEURON and onto the appropriate object in a GENESIS script file.

2. Flood Fill Algorithm

   Flood Fill Algorithm has been extensively used in our project to de-

tect and identify boundaries. Flood fill, also called seed fill, is an algorithm that determines the area connected to a given node in a multi-dimensional array. It is used in the "bucket" fill tool of paint programs to fill connected, similarly-colored areas with a different color, and in games such as Go and Minesweeper for determining which pieces are cleared. When applied on an image to fill a particular bounded area with color, it is also known as boundary fill.

The flood-fill algorithm takes three parameters: a start node, a target color, and a replacement color. The algorithm looks for all nodes in the array that are connected to the start node by a path of the target color and changes them to the replacement color. There are many ways in which the flood-fill algorithm can be structured, but they all make use of a queue or stack data structure, explicitly or implicitly. Depending on whether we consider nodes touching at the corners connected or not, we have two variations: eight-way and four-way respectively.

3. Creation of Cell Models

   Once a cellular morphology has been imported or created in neuroConstruct, groups of sections can be defined to distinguish axons, somata, and dendrites. Subgroups of sections such as proximal, oblique, and apical dendrites can also be defined. The distribution of cellular mechanisms can then be specified for each cell region. For example, a nonuniform channel density can be implemented by varying the conductance density in each group . Ion-concentration mechanisms (e.g., activity-dependent intracellular Ca2+ concentrations) can also be added in this way, as can passive electrical properties (specific membrane capacitance and specific membrane/ axial resistance), allowing spine densities to be simulated without additional compartments. New cell models can be

created from experimental or published data using neuro- Construct by importing/creating cell morphologies and modifying existing ChannelML templates or adding native code. However, the painstaking process of making a detailed multicompartmental cell model from scratch often involves automated optimization of parameters and access to all model variables, which requires coding with a command-line-based simulator or another program.

4. Cell Placement in 3D

   The gross anatomy of a brain region is generated in neuroConstruct by defining 3D regions in which specific cell types are placed. Regions can currently be boxes, spheres, cylinders, or cones, and multiple regions can be used to create composite structures such as the layers found in the cerebellum and cortex. Cells in neuroConstruct are arranged in cell groups, which are created by specifying the cell type, the 3D region in which the cells are found, and the packing pattern used to fill the space. Possible packing patterns include the following: cubic close packing for maximum density in 3D space, evenly spaced packing in 3D with cell body centers aligned, hexagonal planar patterns, single cells placed at a specified location, and cells placed in a one-dimensional line. However, for many brain regions, random cell placement is more realistic (Figure 1C). The number of cells in a specified region can be set, and whether cells should avoid the space occupied by existing cell bodies or can overlap can be specified. This allows cell densities to be matched to experimentally measured values for a particular brain region.

5. Connectivity Algorithms

   Connections can be defined relative to the pre or post synaptic cell. Each

source cell is assigned a number of connections, which can be fixed or variable within set bounds. One or more synaptic mechanisms are associated with the connection, and these can have variable or fixed weights and internal delays. With the morphology-based connection, the target cell can be chosen at random, can be the closest available cell, or the closest cell from a random pool of n possible locations. Maximum and minimum connection lengths can also be set. It is often convenient to calculate the time the AP takes to get from the soma to the synaptic terminal, rather than model the axonal sections explicitly, to reduce computational overhead. AP propagation speed can be specified for cells, and neuroConstruct will calculate the extra synaptic delay from the axonal morphology. For the volume-based connection, an axonal arborization volume is defined and any appropriate target segment falling within this region is a candidate for a connection. Nonuniform connectivity is generated by assigning putative connection locations a connection probability that is a function of radial distance or x, y, z coordinates relative to the source soma. The spatial-dependence function can be defined by the user. A connection is made if a random number (01) is probability; otherwise, another random location is picked until all connections are made.

3 Problem Description

Currently neuroconstruct allows creation of regions which are geometric only, for e.g sphere, cube,cone,Rectangle. We need to modify the tool to allow an arbitrary region defined by a set of points,splines,Random Curves .Motivation for this is that regions as obtained from microscopy images are never geometric. These microscopic images include Spinal Cord scans.

4 Methodology

4.0.1 Image Processing

The first step is preprocessing of crosssection images of spinal cords. This involves 3 stages namely text removal, Closing up the gaps , Applying connected component analysis. Each of these is described below.

(a) Text removal from images

The text letters specifying the regions are coloured in black and they are removed from the image. This forms the first step in image processing, This text forms a large part of noise and it is composed of layers of black pixels. Each of these text represents different cross sections of the spinal cord.

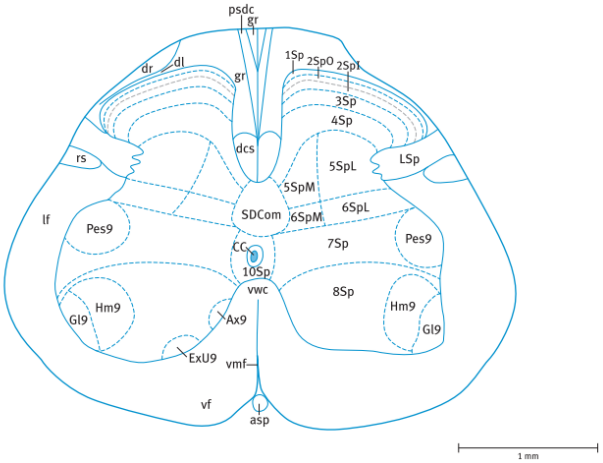

Figure 2: Initial image

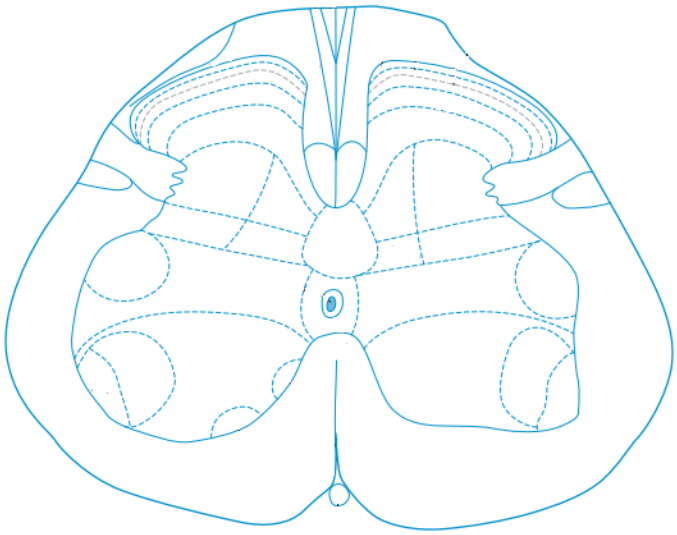

Figure 3: Image after removing text

The black pixels appearing in the first figure have a property that each of the pairwise differences between (R,G,B) triplet exceed no more than 80. This is what differentiates these pixels from the bluish ones which have marked deviation in (R,G,B) values. As this pairwise difference increases, the image becomes more bright. In some other datasets(images) we have got some grey lines as a part of the border, they have been retained but the text has been scrapped off.

(b) Closing up the gaps

Next step is to close the small openings as shown in the above figure. Pixels are examined for their colour and blue pixels are subjected to dilation. This fills up the gaps and results in the image as shown below.

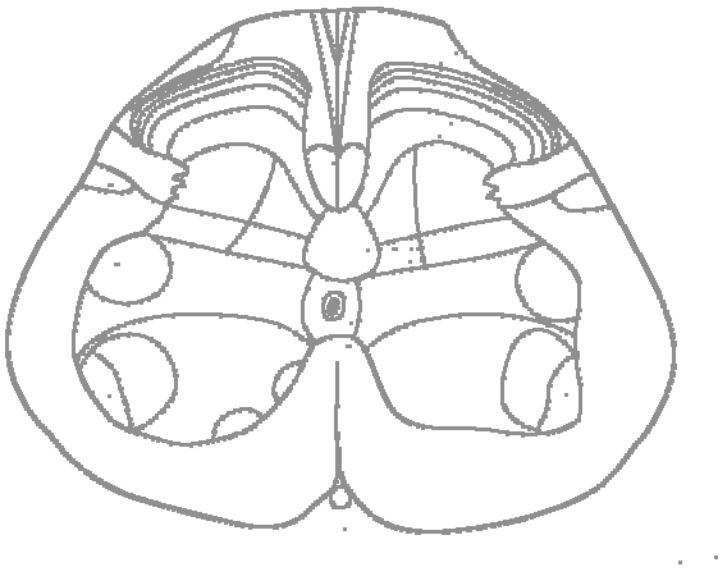

Figure 4: Image after dilation

Erosion is performed on this image to smoothen the borders and to avoid merging of different regions, Depending on the binary file representation, Mask size of erosion has to be decided, In this case it comes out to be 1.

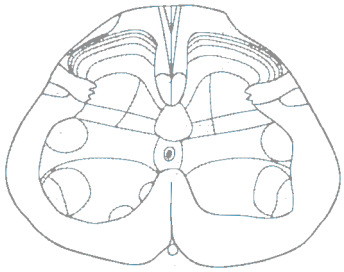

Figure 5: Image after erosion

First we perform square dilation, in which each of the black pixels are extended in 8 surrounding directions. This also results in

thickening by 2 layers of the already closed border. To smoothen this, we scrap off the top and the bottom black pixel layers of the borders.Extending the lines in the direction of their slopes does not work here because it results in formation of ridges which alters the quality of the image.

(c) Applying Floodfill to extract Connected Components

Next step is to colour each of these closed regions with different shades of blue,green, and red. This is achieved by assigning a unique number to each pixel of a closed region and assigning a corresponding negative number to its bundary pixel.In this way, boundaries of each subregions are recorded. We use the standard Breadth First Search to Flood connected components.

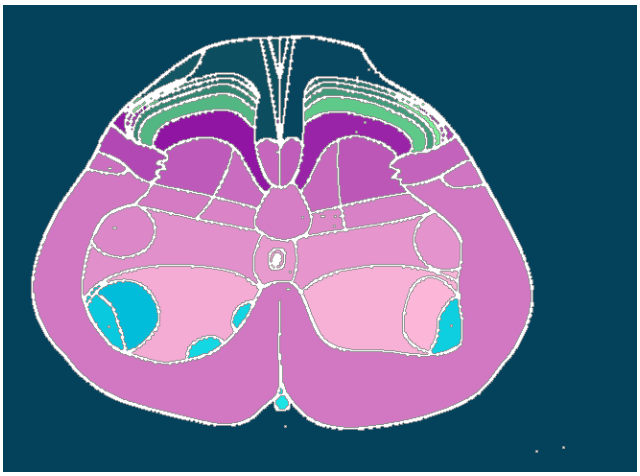

Figure 6: Connected Components of the image

4.1 Visualising these arbitrary Crosssections in 3D and scattering Neurons Randomly over a 3D cube

After the preprocessing step comes the drawing step, and this is implemented in python. There exist no specific algorithms for drawing arbitrary regions, so the conventional algorithm of scanning each pixel for color and extending boundary pixels along z-axis would take O(N*M*f(z)) complexity where (n,m) is the dimension of the image, and z is the interdistance between lower and upper cross sections. f is a linear function of this 'z' assuming bresenham's algorithm is employed for drawing the line. Neurons could be plotted over any given space in this 3D region, and the topview of 3D region would help in better analysis of neurons in the given crosssection.

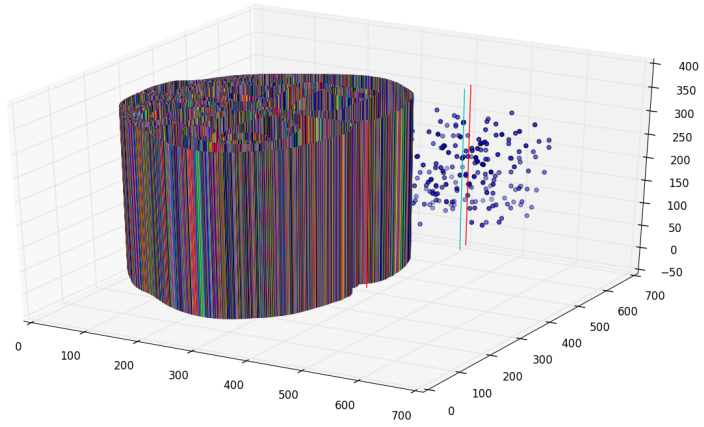

Figure 7: Sideview : All Neurons fall outside

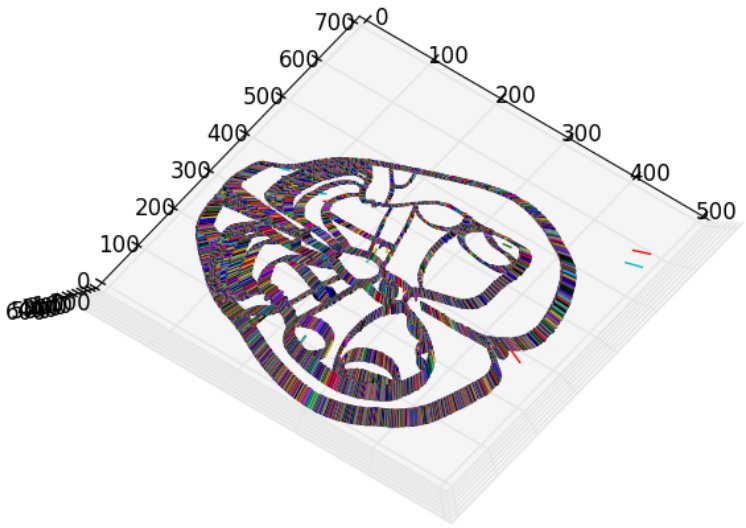

Figure 8: Topview : All Neurons fall outside

4.1.1 Interoperability of classes within NeuroConstruct : with reference to Region modification in NeuroConstruct

(a) Registration of region in RegionTypeHelper:

This class includes basic details about the regions and stores a list the types of all the regions and their descriptions. So each new region has to be registered over in this class. This the class that the GUI calls to give user the choice of which regions are available for the user.

(b) Extending the Region class:

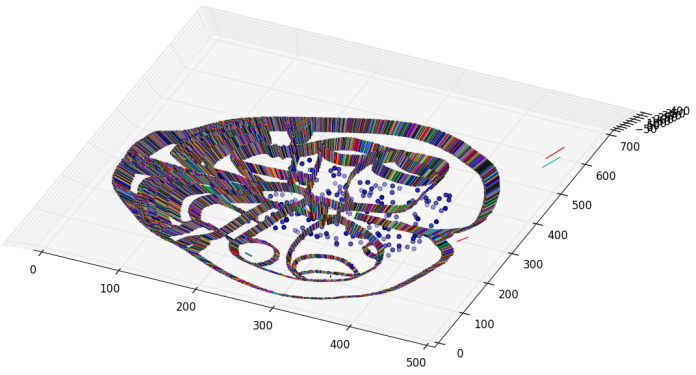

Figure 9: Topview : All Neurons fall inside

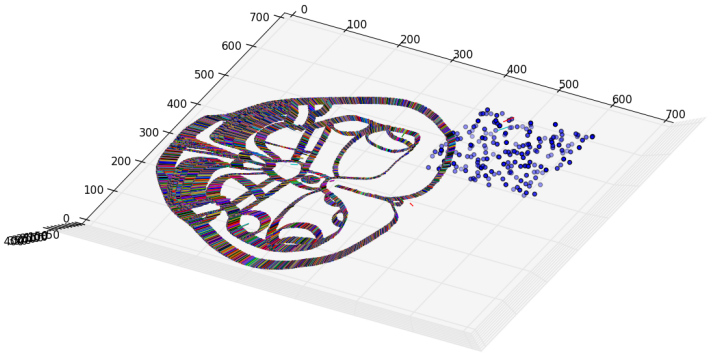

Figure 10: Topview : Some Neurons fall inside

Each type of region has to have a separate class of its own to describe the functionalities relevant to the region. Note that the software expects each region class to perform certain common opera-

tions i.e for eg. find out whether the point specified is inside the region. Hence there is a common Region class that is extended by all the region types. The actual functions used to perform these functionalities is different for each region type, hence the functions in the Region class are overridden by the functions in the subclasses.

(c) Drawing the new region in the 3d canvas

The actual drawing is to be done by the function addPrimitiveForRegion() present in each regionType class. This function in turn calls a function in the 3dutils class of of the J3D folder. The 3d figure has to be an object of the Primitive class. Hence the new class has to be created as a subclass of Primitive. The actual 3d objects are themselves added as children of a TransformGroup whereas each subpart of the 3d shape is represented as an object of Shape3d.

(d) Other Details of Regions in NeuroConstruct.

NeuroConstruct uses an XML format to save details of the regions and other parameters about a neuroConstruct project. However saving certain regions like ConvexHull requires signnificant modification to the .ncx XML file.Gui for accepting parameters of the Region. NeuroConstruct uses a standard Java Swing based GUI for accepting user input. Depending on the region, the GUI may have to be modified significantly.

4.1.2 Integration with NeuroConstruct

6. The following figure explains the class linkage. MainFrame.Java invokes the Python module which takes in the parameters through ArbitraryRegion.java, which is further governed by RegionTypeHelper class. Now if the developer wishes to add more functionality to this arbitrary region, he could do so by embedding that in this python class.

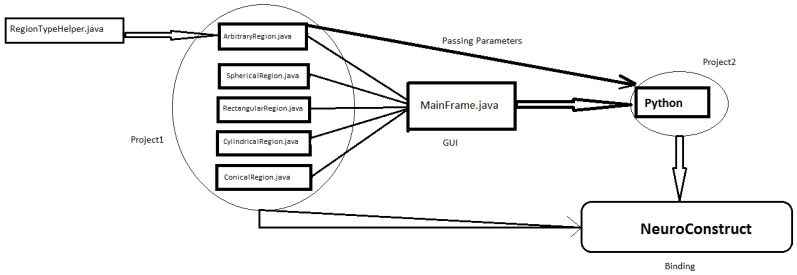

Figure 11: Invoke Operation in NeuroConstruct

5 Conclusion and Future work

Arbitrary regions were processed properly and were drawn using python, this functionality is integrated in to neuroconstruct. One of the most positive aspects of our work is that this arbitrary functionality runs along with neuroconstruct and its interference with other traditional regions of neuroconstruct is null, thus providing a simultaneous view to developers and users. If some extra functionality like data files and algorithms need to be added for arbitrary region, then the user can easily do that without worrying much about the complex aspects of the source code, whose understanding formed the major part of our project. NeuroConstruct uses an assortment of Java libraries and features to perform its functionality. Some of these are legacy systems that have been out of widespread use for quite a while by now. The most obvious example of this is JAVA3D. JAVA3D originally introduced in 1997 had its last stable official release in 1997 and most Java softwares of this scale have moved onto better JAVA3D libraries for their graphics functionality. One of the fundamental diffiulties that we faced in trying to come up with better 3d visualization features was the steep learning curve of the scene graph model. This may prove to be a hindrance for further development of the software Incremental improvements can also be made in certain other areas of NeuroConstruct, especially XML storage. At the moment the entire project specification storage feature in NeuroConstuct uses a basic XML format, that is designed keeping in mind very basic geometries. As more complex geometry features are added, a more functional XML format would be required. Another part of the software in which huge improvements can be made is UI. With the advent of touch based interfaces and interactive computing, the overall functionality of the software

can be hugely enhanced by their addition even to a slight extent.

References

[1] Gleeson, Padraig, Volker Steuber, and R. Angus Silver. "neuroConstruct: a tool for modeling networks of neurons in 3D space." Neuron 54.2 (2007): 219-235.

[2] libNeuroML and PyLEMS: using Python to combine procedural and declarative modeling approaches in computational neurosciences. Michael Vella, Robert C. Cannon, Sharon Crook, Andrew P. Davison, Gautham Ganapathy, Hugh P. C. Robinson, R. Angus Silver and Padraig Gleeson Journal: Frontiers in Neuroinformatics, 2014, Volume 8s

[3] Muller E, Bednar JA, Diesmann M, Gewaltig M-O, Hines M, Davison AP. Python in neuroscience. Frontiers in Neuroinformatics. 2015;9:11. doi:10.3389/fninf.2015.00011.

[4] Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, Farinella M, et al. (2010) NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail. PLoS Comput Biol 6(6): e1000815. doi:10.1371/journal.pcbi.1000815

[5] Interoperability of Neuroscience Modeling Software: Current Status and Future Directions.Neuroinformatics, 2007, Volume 5, Number 2, Page 127 Robert C. Cannon, Marc-Oliver Gewaltig, Padraig Gleeson

[6] Flexible Specification of Data Models for Neuroscience Databases.Philipp L. Rautenberg and Thomas Wachtler Conference: 2011 22nd International Workshop on Database and Expert Systems Applications, Year: 2011,

[7] http://www.java3d.org/

[8] http://mdp-toolkit.sourceforge.net/$_s tatic/Bochum2009TalkPythonMDP.pdf$

[9] www.neuroconstruct.org/

[10] Computational Neuroscience with Python Matlab https://www.youtube.com/watch?v=aGdkaufQekQ