Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Neurocomputing 70 (2006) 327–342

NEUROCOMPUTING

www.elsevier.com/locate/neucom

NeuGen: A tool for the generation of realistic morphology of cortical neurons and neural networks in 3D

J.P. Eberhard*, A. Wanner, G. Wittum

Interdisciplinary Center for Scientific Computing, Simulation in Technology, University of Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany

Received 25 January 2005; received in revised form 12 January 2006; accepted 12 January 2006
Communicated by T. Heskes
Available online 7 July 2006

Abstract

We introduce the software package NeuGen for the efficient generation of anatomically accurate synthetic neurons and neural networks. NeuGen generates non-identical neurons of morphological classes of the cortex, e.g., pyramidal cells and stellate neurons, and synaptically connected neural networks in 3D. It is based on sets of descriptive and iterative rules which represent the axonal and dendritic geometry of neurons by inter-correlating morphological parameters. The generation algorithm stochastically samples parameter values from distribution functions induced by experimental data. The generator is adequate for the geometric modelling and for the construction of the morphology. The generated neurons can be exported into a 3D graphic format for visualization and into multi-compartment files for simulations with the program NEURON. NeuGen is intended for scientists aiming at simulations of realistic networks in 3D. The software includes a graphical user interface and is available at http://neugen.uni-hd.de.
© 2006 Elsevier B.V. All rights reserved.

Keywords: NeuGen; Modelling neural morphology; Stochastic generation; Cortical neural network; 3D geometry

1. Introduction

In the past decade, computer simulations of cellular behaviour of single neurons or small networks of neurons with an accurate dendritic and axonal morphology have become increasingly common. The complex morphology of the neurons is usually taken from experimental data resulting in anatomically precise compartmental models. The complex geometric morphology is important for the understanding of the neuronal integration, see e.g. [8,28,19,16,13,7,23].

However, the reconstruction of anatomically precise compartmental models of neurons from experiments either manually or automatically with the help of reconstruction software is rather tedious and time-consuming. Depending on dye and recording technique such a reconstruction is only feasible for one or a few neurons at a time. Hence a software program that is able to generate three-dimensional (3D) synthetic neuron geometries and neural networks in conformity with experimental findings is an invaluable tool. We present the software package NeuGen for the generation of realistic neurons and neural networks in 3D. NeuGen provides an easy way to construct not only single cells but also complex networks with a large number of neurons. The networks are interconnected by synapses. These synapses are created at axonal locations determined by a function of the distance to the dendrites of all other neurons.

The software L-Neuron introduced in [1,2] is a modelling tool to generate anatomically accurate neuronal analogues. It is oriented towards single-cell generation and is an implementation of the algorithm in [9,5]. While L-Neuron is based on recursive rules using a Lindenmayer-system formalism, i.e., looping rules given by a Lindenmayer-string, NeuGen implements a straightforward algorithm which utilizes forward-stepping rules. The key of the generative method lies in a tail-recursive, thus forward-stepping and not truly

*Corresponding author. Tel.: +49 6221 54 8856; fax: +49 6221 54 8860.
E-mail addresses: eberhard@uni-hd.de (J.P. Eberhard), awanner@ix.urz.uni-heidelberg.de (A. Wanner), wittum@uni-hd.de (G. Wittum).

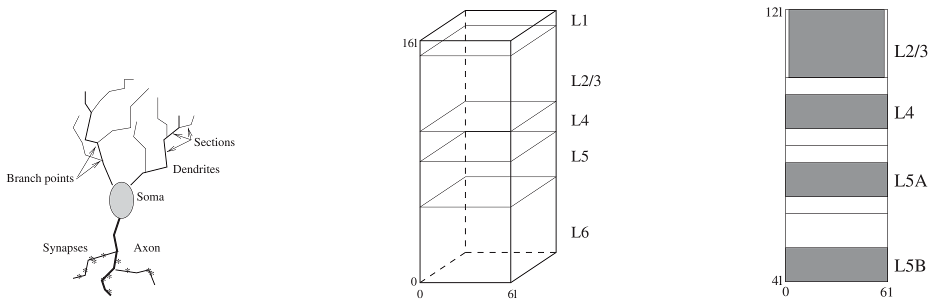


Fig. 1. Left: Sketch of a neuron with its constituents, dendritic sections, and branch points. Middle: Schematic representation of a cortical column with its layers denoted by layer 1 (L1), layer 2/3 (L2/3), layer 4 (L4), layer 5 (L5), and layer 6 (L6), and its dimensions where $l = 100\,\mu\text{m}$. Right: Bounding boxes for the locations of the somata used by NeuGen for the generated cells. The boxes are illustrated by the shaded areas within the cortical column.

recursive, function to generate the sections of a cell. Further each neuron type builds an independent class in NeuGen. The algorithm directly maps anatomical fingerprints of the different neuron types onto a coordinate-based description for the three-dimensional neuron geometry. Therefore, NeuGen uses statistical distributions based on morphological parameters given by realistic data, see e.g. [21,22,6,14], and some basic compartment model elements.

ArborVitae is another software for the reconstruction of networks by algorithmic amplification of morphological data [30]. It generates large-scale, anatomically accurate networks similar to NeuGen. ArborVitae develops brain circuits by given growth rules and subcellular informations whereas NeuGen generates neurons simply by morphological rules. However, the variables used to describe the generated structures, such as soma and dendrite locations, length, diameter and taper of segments are the same. While ArborVitae focuses on generating cells of the hippocampus and other brain regions [30], NeuGen primarily generates cells of the neocortex. NeuGen also does not consider competition in the neuronal generation process as it is done in [32]. The program A-Cell-3D [11] also models detailed three-dimensional neuron morphology which is able to model the calcium dynamics in spines, too. Unlike NeuGen it develops only single cell geometries.

In NeuGen geometric constraints are used to describe the different neuron classes. These constraints are characteristic features as well as morphological rules obtained from experimental findings. These rules correlate the morphological parameters to describe the axonal and dendritic geometry of the neurons. The basic description of a single dendrite, for instance, is given by a list of sections. Each section is represented as a series of connected, cylindrical or frustum-shaped segments, also termed compartments, containing a numerical tag, the spatial Cartesian coordinates, the start radius and the end radius. This structure is used to assemble an accurate description of the dendritic morphology. We additionally characterize the sections by morphological parameters, such as branch point angles and branching depth. Statistical distributions are associated with each of these parameters, i.e., the final value for the parameter is sampled from a distribution and then used to generate the morphology. In the literature, there are a few algorithms available that describe neuronal morphologies purely on the basis of such statistical distributions of geometrical parameters [9,5,17]. Unlike the description applied in NeuGen, the algorithm in [17] only yields information on an ensemble of neurons, i.e., the description is not sufficient to provide an explicit and precise map of the morphology. Whereas the implementation of the algorithms of [9,5] in L-Neuron [1] can provide an explicit representation of complete, individual neurons.

NeuGen is intended to generate neural networks as cortical columns connecting layer 5 (L5) and layer 4 (L4) with layer 2/3 (L2/3) in the cortex; see Fig. 1 (middle) for the layers. These model columns are found in the somatosensory barrel cortex in rodents [22]. In the following we use the classification of neurons in the cortex due to their anatomical shape, i.e., the classes of pyramidal cells, star pyramidal cells, and stellate neurons. NeuGen is based on this classification, that is, NeuGen provides the generation of L2/3 pyramidal cells, L5 A and L5 B pyramidal cells, L4 star pyramidal cells, and L4 stellate neurons. The different cell types are arranged automatically in their associated layers. The generated three-dimensional networks are then represented by an object-oriented data structure and can be applied for visualizations and for numerical studies with the simulation program NEURON [10].

NeuGen is written in ANSI C++ and Java. It runs on Linux, MS Windows and MacOS X platforms, and it is available at http://neugen.uni-hd.de.

2. Methods

The NeuGen neuron generator is intended to produce realizations of neural networks in 3D where the geometry and morphology are modelled as realistic as possible. To this end we use appropriate probability density functions to describe the distributions of the morphological parameters. The morphological data is extracted from the morphometric analysis of the columnar innervation domain, as done e.g. in [28,13,15,21]. The neuroanatomical parameters used by NeuGen are obtained from the literature [15,12,6,14], or they are measured from image files of experimental data.

The configuration is done with the help of three files, called `Param.neu`, `Interna.neu`, and `Output-Options.neu`. The file `Param.neu` includes standard parameters such as the number of generated neurons, whereas `Interna.neu` gives access to enhanced parameters such as the random generator seeds. A default set of configuration parameters for all neuron classes of NeuGen is provided with the software package. In these default configuration files all parameter values are set according to experimental data [14,8,13,15,3,21,28,18,29].

NeuGen reads the list of the neuroanatomical parameters from the configuration files. These values are translated with the help of functionals of the uniform distribution into different statistical distributions. When a neuron is generated, values of the parameters are sampled stochastically from these distributions. The quantitative description of these distribution functions is based on the morphological data given in the literature [28,13,15,21,12,6,14]. In Section 2.3 we describe the functional shape of the probability density functions of the underlying stochastic parameters and properties.

The experimental data that had been analysed for the generation of the cells is given by the following list for the different neuron classes:

For L4 stellate neurons [15,3,21,14], for L2/3 pyramidal cells [15,27], for L4 star pyramidal neurons [3,14,29], for L5 A pyramidal cells [28,13,8], and for L5 B pyramidal cells [28,18].

2.1. Neuroanatomical constraints for a cortical column

NeuGen follows general neuroanatomical constraints for the cells in a cortical column. In the following we list the major experimental results of the morphometric analysis for the cortical column as found in rodents, see [21,22,3,6,14]. These qualitative results are used for the parameterized generation in NeuGen.

- The axonal tree of L4 spiny stellate neurons and the dendritic tree of L2/3 pyramidal cells are assigned to a so-called "home" barrel, see [14]. In particular, the axonal collaterals in a column span all cortical layers with a higher density within L4 and L2/3.
- Spiny stellate neurons are mainly assigned to L4, i.e., their dendritic field is restricted to the borders of the "home" column where the somata of these neurons are located. Stellate cells show an asymmetric dendritic configuration where the dendrites are oriented towards the centre of the barrel.
- The basal dendrites of L2/3 pyramidal cells show a symmetric arrangement around the respective somata. The apical dendrites of L2/3, L5 A and L5 B pyramidal cells form a characteristic terminal tuft in L1. Star pyramidal cells have a prominent thick apical dendrite that terminates in L2/3 without forming a tuft. They are located towards the centre of the barrel with symmetrically arranged basal dendrites.

2.2. Representation of the morphology

An object-oriented framework underlies the design of NeuGen for the description of the neuron morphology. The framework is also adapted to the description and stochastic generation of neuron populations for networks. For the object-oriented representation we consider the soma, the dendrites, the axon, and the synapses as the relevant constituents of a neuron, see Fig. 1 (left). This cell description forms the basis of the generation of neural networks as described in detail in Section 3.

Spines are known to have different shapes depending on the synaptic properties. The detailed shape of a spine is hard to distinguish even at the current limit of optical resolution and thus the spine class in NeuGen is presently an empty class. We provide it as a spaceholder class for future extensions.

2.3. Stochastic generation and distribution functions

We apply a stochastic concept for the generation of realizations of a random neural network. We use random numbers to determine the final values that are necessary for the individual setting and construction of the dendritic and axonal trees of a single neuron as well as for the construction of a network. These random numbers are drawn from different distribution functions. The parameter values for the generation of a single realization are then determined with the help of the configuration values which represent the mean values. Hence the distribution functions are directly linked to the parameter values inside the generation algorithms, see Appendix B for the detailed algorithms. This implies that the generated neurons or networks are unique even if we use the same parameter files but different initial values for the random number generation. The applied distribution functions are based on functionals of the uniform distribution $P_{a,b}(z)$. This simplifies the implementation and causes very fast generations. The underlying uniform distribution function $P_{a,b}(z)$ is defined by

$$P_{a,b}(z) = \begin{cases} 1/b & \text{if } |z-a| \leqslant b/2, \\ 0 & \text{otherwise,} \end{cases}$$

where a and b are given constant parameters (see Fig. 2 for an illustration). With the help of $P_{a,b}(z)$ we form the vector distribution functions $\mathbf{P}_{\theta_1,\theta_2}(z)$ and $\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}(z)$, where θ_1 and θ_2 are given angles, and $\mathbf{e}$ is a given direction. $\mathbf{P}_{\theta_1,\theta_2}(z)$ and $\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}(z)$ generate random vectors on the unit sphere that are equally distributed in a spherical angle element, see also Appendix C. Further we define the vector

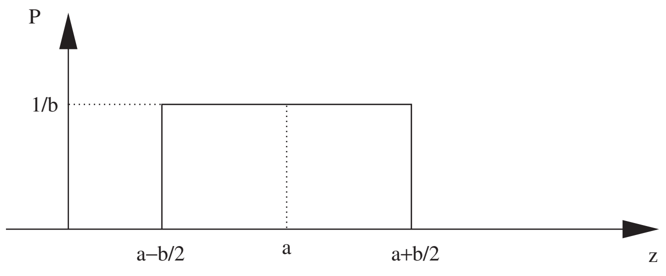


Fig. 2. Illustration of the distribution function $P_{a,b}(z)$ with parameters a and b.

distribution function $\mathbf{P}_{\perp,\mathbf{e}}(z)$ which generates random vectors perpendicular to a given vector $\mathbf{e}$, see Appendix C. These kinds of distribution functions are used by NeuGen extensively for the generation of axonal and dendritic branches. Here, the values for θ_1 and θ_2 vary with respect to the morphological properties of the parameter described by $\mathbf{P}_{\theta_1,\theta_2}$ or $\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}$.

A class called `Randint` provides all classes for the random number generators and random vector generators in NeuGen. The seed values for the random number generators are located in the file `Interna.neu`.

2.4. Parameter values for the generation of neurons and networks

Here, we describe the statistical properties of a neural network embedded in a cortical column of the cortex located between $(0,0,0)$ and $(6l,6l,16l)$, see Fig. 1 (middle). In the following, (x,y,z) denotes a 3D vector and $l = 100\,\mu\text{m}$.

The coordinates of the somata of L4 stellate neurons are drawn from the vector $(P_{3l,6l}(z), P_{3l,6l}(z), P_{9l,l}(z))$. The soma locations of L2/3 pyramidal cells are given by $(P_{2.6l,4.5l}(z), P_{2.6l,4.5l}(z), P_{11l,2l}(z))$. For the L4 star pyramidal cells the coordinates of the somata are drawn from $(P_{2.5l,4l}(z), P_{2.5l,4l}(z), P_{9.5l,l}(z))$ and for the L5 A and L5 B pyramidal cells from $(P_{3l,6l}(z), P_{3l,6l}(z), P_{7l,l}(z))$ and $(P_{3l,6l}(z), P_{3l,6l}(z), P_{4.5l,l}(z))$, respectively. For an illustration, the soma locations of the different neuron types are shown as bounding boxes in Fig. 1 (right) by shaded boxes.

The number of branches of the dendritic and axonal trees are drawn from $P_{a,a}(z)$ for all classes of neurons, where the parameter value a is specified for each branch type and class by the configuration. The parameter for a is called `nbranch_param` for dendrites and `nbranch` for the axon. The basal dendritic parts of all neuron classes are constructed by the same statistical and morphological framework but with different parameter values for each neuron class. The number of branches of the terminal tuft is drawn from $P_{a,a}(z)$ for the apical dendritic trees of L2/3 and L5 pyramidal cells. Here, the value of a can be given for different branch generations separately; typical values are 2 or 3. The parameters `nbranch_param` and `nbranch` are associated with the distribution $P_{1,1}(z)$. Whereas the directions of segments for the axonal and dendritic branches are given by $\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}(z)$. The angles θ_1 and θ_2 are given by the fixed range of the parameter `branch_angle`. These values are set in the configuration by `min` and `max` for `branch_angle`. The latter may vary for different neuron classes as well as for different branching depths within the axonal and dendritic trees. The maximum branchgeneration depth is 10 for dendrites. Therefore, the branch generations are indicated by the keys `gen_0` and subsequent `siblings` in the configuration. The number of segments which is given by an one-dimensional density (per length) is called `nparts_density`. The length for basal branch parts are given by `nparts_density`, `len_param` and the location of the branch point at the parent branch. For apical dendrites, the number of segments grows linearly with `nparts_density`, and the larger the distance between the branch point and the start of the parent branch the smaller the length of the branch.

Taper is regarded for all neuronal branches. At each axonal branch point the starting values of the radii are given by the half of the radii of the last segment before the branch point. To compute the radii of two daughter branches in the dendritic tree NeuGen applies Rall's "power rule" [25,26] with a given power and threshold for the radius. (Two parameters for Rall's rule can be set in the configuration.) The end radii of dendritic and axonal branches are given by the minimal radii defined by the parameter `min`. All parameters responsible for taper may depend on the given cell type, and their values are bounded by the given minimal parameters. The detailed algorithms for constructing dendritic and axonal branch parts are given in Appendix B.

2.5. Synapses and network connectivity

The network connectivity is generated in NeuGen as follows. Axonal synapses are created where a spatial connection between the axon of a neuron and a dendritic part of any other neuron of the network is given. For this purpose, a distance function marks a connection between two neurons when the geometric distance between the neurons is below the given threshold distance called `dist_synapse`. Therefore, the number of generated synapses of a network depends on the `dist_synapse` parameter. The number of synapses for increasing number of cells in the network and two different values of `dist_synapse` are shown in Table 2.

Further, NeuGen provides the possibility to generate synapses for neurons without the criterion of spatial connectivity (in addition to the synapses generated by spatial connections). This is done by distributing synapses over the dendritic trees according to a distribution function, that is, choosing randomly locations for synapses on the dendrites. The distribution depends on the geometric distance from the soma. Such synapses can be created at dendritic locations of L4 stellate cells or at dendritic locations of all cells in the network. To distinguish this type of synapse from the synapses due to spatial connections they are termed `nf_synapses` (non-functional synapses).

2.6. Output and visualization

The generated neurons or neural network can be saved in different file formats. The options for the different outputs have to be set in `OutputOptions.neu` via the configuration. The simplest output is a log file which contains the relevant parameters of the generated neurons or network, see Section 4.1. Further, a HOC file for the visualization with the program NEURON [10] can be written as an output. A model for numerical simulations with NEURON can be written into a model HOC file, too. The implemented model for the channels and electrophysiological parameters is given by a model of the Senselab's Model Database (ModelDB), see [20,31,16]. This HOC file is written in addition to the file which includes the neural network geometry. It is adapted to the generated network and can be used directly with NEURON.

A 3D visualization output can be generated by NeuGen which is an input for the Open Visualization Data Explorer (OpenDX). The data and program files are exported in the native OpenDX format. OpenDX can interactively display and render the neurons or neural network in 3D. The free software is available at http://www.opendx.org.

3. Object-oriented structure of NeuGen

3.1. Object-oriented class design for the morphological description

We briefly describe the object-oriented class design that represents the neurons in NeuGen.

The classes `Net` and `Neuron` construct networks and neurons, respectively, based on the classes for the axon, the dendrites, and the soma, see Fig. 3.

The soma of each neuron is constructed by an instance of the class `Cellipsoid`. It represents an ellipsoid which

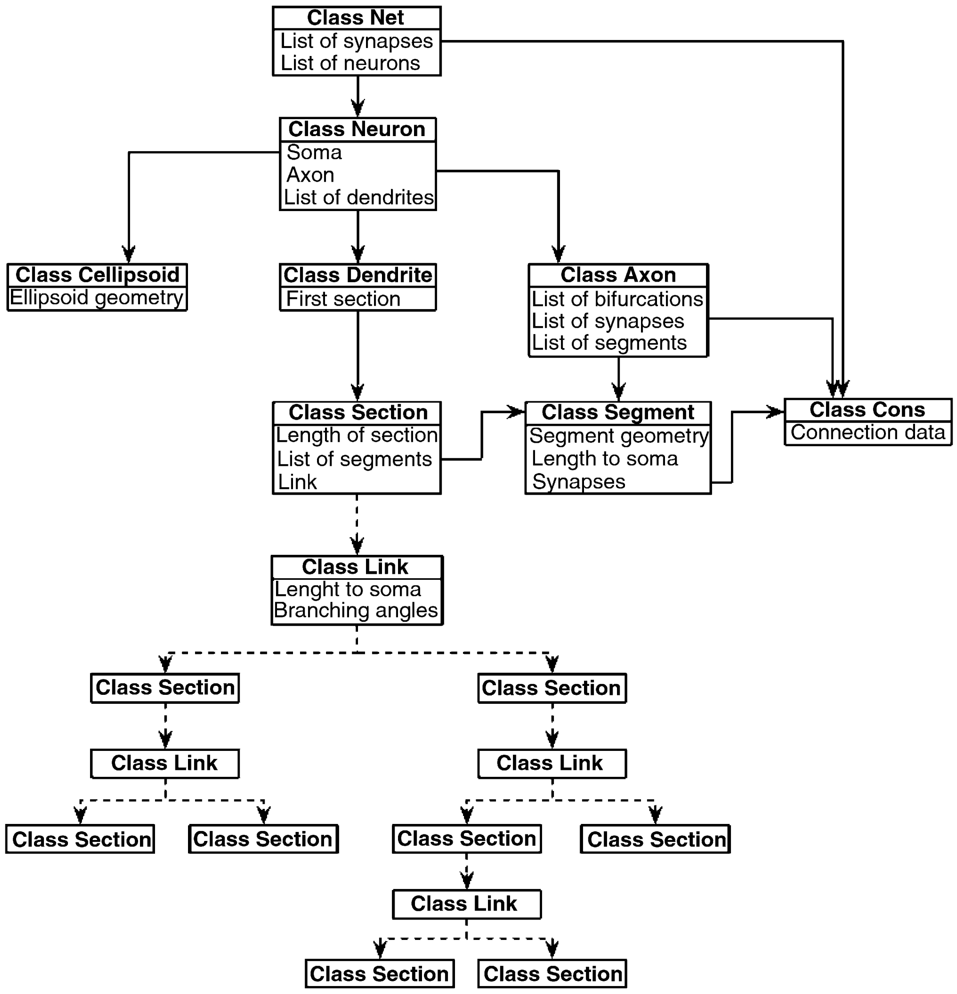


Fig. 3. The hierarchical class structure of NeuGen from the class Net to the class Section. The dashed arrow lines show possible connections in the hierarchy belonging to the class Dendrite.

approximates the shape and volume of the soma as set in the configuration. The ellipsoidal shape given by the lengths of the semi-axes of the ellipsoid may also depend on the type of the neuron.

The basic class for a compartment or segment is called Segment, whose instances are interlinked to form the overall neuronal morphology. This class represents a compartment in the form of a cylinder or frustum, which is defined by the coordinates and radii of the start point and end point. A segment can be a terminal segment, or it can bifurcate to produce two daughter segments. The intermediate angles between the segments are stored for the new daughter branches. Several segments build a section, and two sections are connected by a link which contains the branching angles. The class Section, therefore, contains a list of segments and has a reference to the parent and daughter sections via Link. Objects of the class Section are always connected with the class Link, see Fig. 3, and within a section there are no bifurcations.

The basic class for constructing the axonal arborization is called Axon. The class is designed by the main axonal branch which consists of compartments given by Segment. Beside the main branch, the class Axon creates different branches starting from the main axonal branch. At the moment, the axonal tree is described in the morphological description by a piecewise linear representation using the class Segment.

The base class Dendrite represents a single dendrite with its branches of the dendritic arborization of a neuron. The dendrites always start from the soma and can have several dendritic branches with specified branching depths which may vary due to the neuron type. Unlike the axon, the dendritic tree is given by a hierarchical structure using the class Section and Link, see Fig. 3.

An additional class—called Cons—contains the data structure for storing the neural information of the synapses. This class stores the tag of the dendritic section, the sequential segment number inside this section and the segment tag of the axon which are connected by a synapse. It also stores the sequential numbers of the two neurons being interconnected. The class Cons is used for a list of synaptic connections in the class Net.

3.2. The classes Neuron and Net

The class Neuron builds a single neuron based on the classes for the axon, the dendrites, and the soma. Various subclasses of Neuron provide the relevant classes for the instantiations. The subclasses of Neuron are L4StellateNeuron, StarpyramidalNeuron, L23PyramidalNeuron and L5PyramidalNeuron which is derived into L5APyramidalNeuron and L5BPyramidalNeuron. The parameter values of each subclass are also inherited by the base classes. However, due to the object-oriented design these parameter values can be overwritten. The inheritance diagram for Neuron is shown in Fig. 4. Due to the object-oriented class design a new neuron class can be added to the class hierarchy by copying an existing subclass of the class neuron.

The class Net constructs and connects several neurons to a neural network. The constructor of Net allows to generate a neural net with given numbers for the different neuron classes as follows. The parameter nL4stellate is the number of L4 stellate neurons, nL23pyramidal and nL5Apyramidal and nL5Bpyramidal is the number of L2/3 and L5 A and L5 B pyramidal neurons, respectively, and nstarpyramidal is the number of star pyramidal cells in the generated net, see also Appendix A. The constructor of Net initializes the neural network and uses the various subclasses of Neuron to create a cortical column. Net also supports functions to return electrophysiological values on the constructed neural network to support enhanced simulations in the future. Further, the class Net comprises the routines to generate the output files

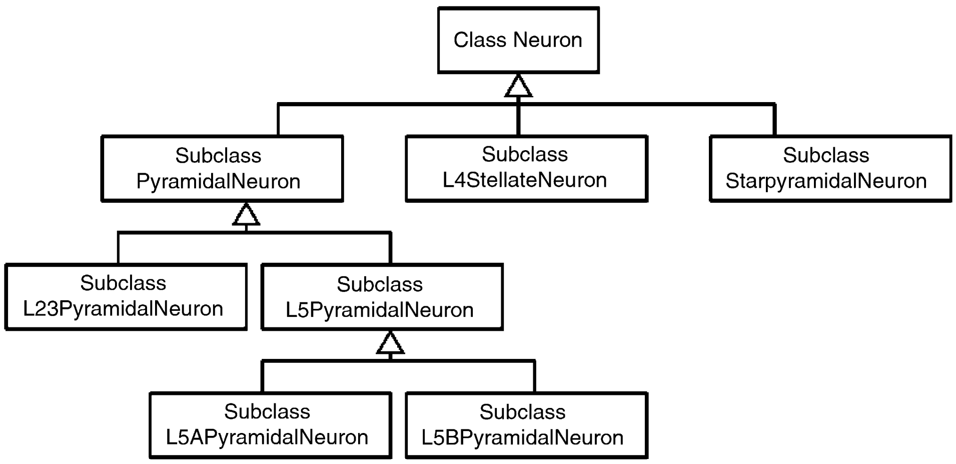

Class Neuron
Subclass PyramidalNeuron
Subclass L4StellateNeuron
Subclass StarpyramidalNeuron
Subclass L23PyramidalNeuron
Subclass L5PyramidalNeuron
Subclass L5APyramidalNeuron
Subclass L5BPyramidalNeuron

Fig. 4. Inheritance diagram for the subclasses of the class Neuron.

(OpenDX, HOC, TIFF) for the visualization of the constructed network by different graphical programs, see Sections 2.6 and 4.2.

3.2.1. *Configuration of NeuGen*

The configuration of NeuGen can be done with the integrated graphical user interface which automatically writes the configuration files. The main configuration file `Param.neu` contains the parameters for the different neuron classes and for the generation of the dendritic and axonal trees. All length values are given in µm and all angles in degree. The configuration parameters of a class are inherited to all subclasses. Thus, parameters given, for instance, for the class `Neuron` are inherited to all its subclasses (see Fig. 4) if they are not changed otherwise. Since the identification of the parameters by their names is clear and the meaning of the parameter names is obvious, the graphical user interface is easy to manage, see Appendix A, where a snapshot of the configuration editor is shown. A list of relevant configuration variables can also be found in Appendix A. Additionally, the configuration file, called `Interna.neu`, has access to the enhanced variables for the different neuron classes which are not part of the file `Param.neu`. These parameters can be changed with the configuration user interface as well.

4. Discussion and results

The presented generation of the neuron morphology with NeuGen is intuitive and brings all the benefits of the object-oriented paradigm, which include portability, extensibility, and re-usability. Building on the two basic classes—`Segment` and `Cellipsoid`—we have developed a seven class model in NeuGen which can be used as a general framework for numerical studies on realistic

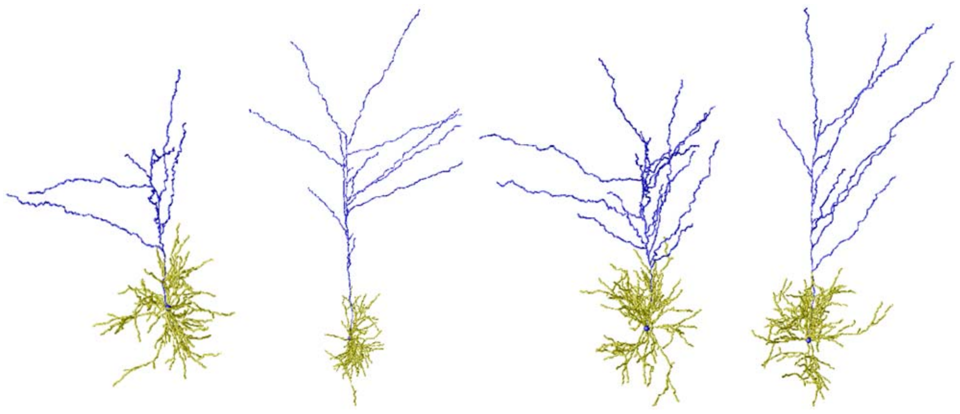


Fig. 5. Four different realizations of L4 stellate neurons with 12 dendrites starting from the soma which are shown in yellow. The axonal tree is coloured in blue. The visualization was done with OpenDX.

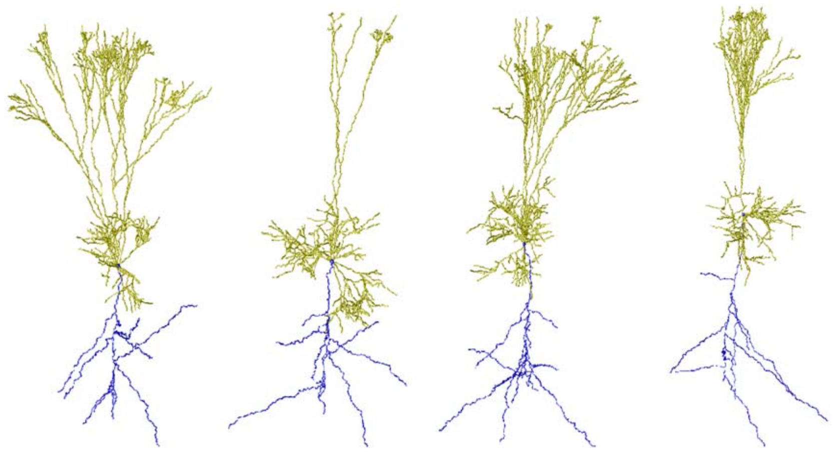


Fig. 6. Four different L2/3 pyramidal cells with eight dendrites starting from the soma (yellow). The cells show the terminal tuft being characteristic for the apical dendrite.

morphology. With the additional classes for the net and the synapses (`Net` and `Cons`) the significant morphological information of a neuron is captured in the attributes and relationships of the object instances of these seven classes.

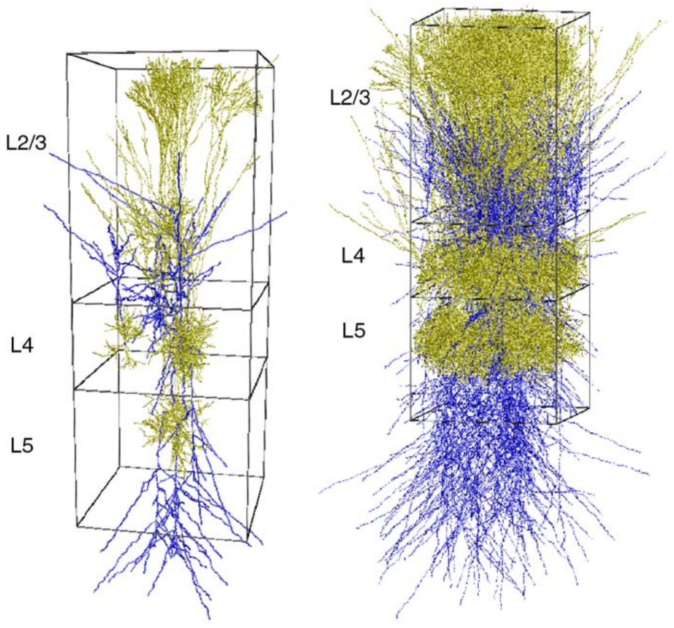


Fig. 7. Left: A generated network for a cortical column with 10 cells. The net consists of three L4 stellate neurons, two L4 star pyramidal cells, three L2/3 pyramidal cells, and two L5 A pyramidal neurons. All neurons have nine dendrites starting from the soma. Right: A generated network for a cortical column with 150 cells. The net consists of 40 L4 stellate neurons, 30 L4 star pyramidal cells, 40 L2/3 pyramidal cells, and 40 L5 A pyramidal neurons. The neurons have nine dendrites starting from the soma. The boundaries of the cortical column are indicated by the boxes.

NeuGen uses parameter distributions derived from anatomical data to construct synthetic neurons of different morphological classes. Within each class, the implemented algorithm produces non-identical neurons. The generated cells strongly resemble the morphological classes of the real neurons from which the morphological parameters were extracted. To illustrate the capabilities of NeuGen, Figs. 5–7 show various results of generated neurons and networks. The plots were visualized with OpenDX. Fig. 5 displays four different results for L4 stellate neurons where the dendrites are shown in yellow and the axon in blue. Fig. 6 shows four different L2/3 pyramidal cells with seven basal dendrites and one apical dendrite that ends in the terminal tuft. Fig. 7 (left) shows a small network of ten cells embedded in a cortical column. The net contains three L4 stellate neurons, two L4 star pyramidal cells, three L2/3 pyramidal cells, and two L5 A pyramidal neurons. Each cell has nine dendrites. Fig. 7 (right) displays a generated network with 150 cells. The network represents a cortical column from layer 1 to layer 5. The boundaries of the column are indicated by the surrounding boxes. The generated neuron morphology shown in the figures is very realistic compared to experimental images, as shown e.g. in [15,28], and bears comparison to cells reconstructed with the aid of reconstruction software [4]. For further illustrations with NeuGen see http://neugen.uni-hd.de.

The long-term goal for the synthetic construction is to create the model such that the experimental data from real neurons and synthesized neurons cannot be statistically distinguished. To this end, in Fig. 8 (left) the branch point distribution $B(x)$ is plotted for the dendritic tree as a function of the distance from the soma. The extracted width of the dendritic tree $W(x)$ is

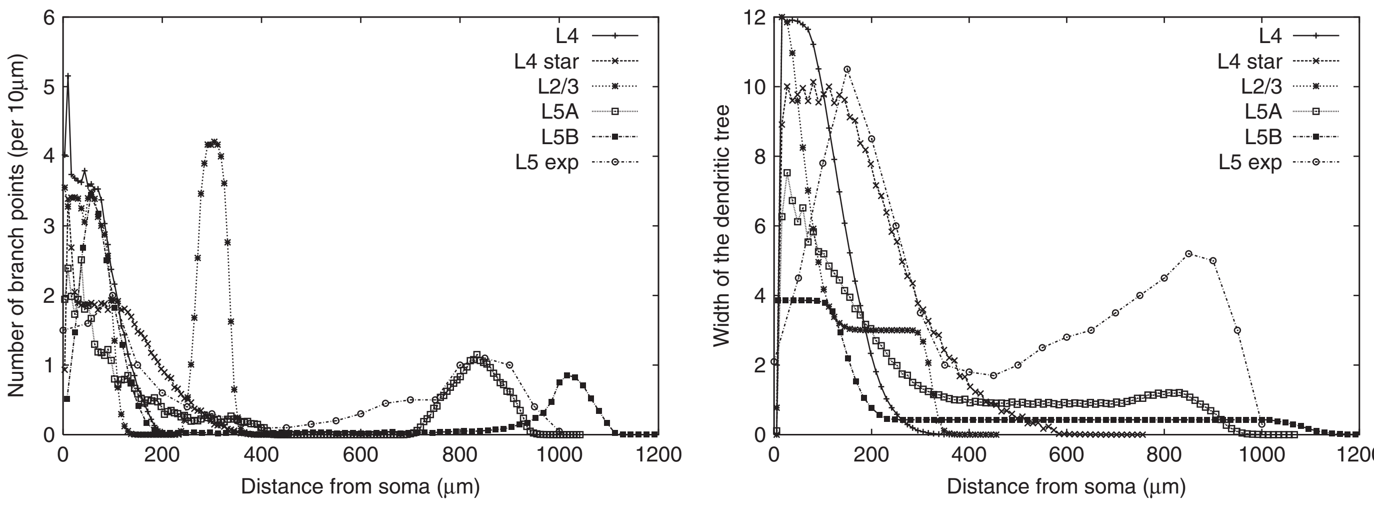


Fig. 8. Dendritic fingerprints extracted for the apical and basal dendrites of 4200 neurons generated by NeuGen. Left: One-dimensional branch point density $B(x)$ for the dendrites as a function of the distance x from the soma for L4 neurons, L2/3 pyramids, and L5 A and L5 B pyramidal cells. $B(x)$ is defined as the number of branch points per 10 μm dendritic length. Right: Width of the dendritic tree $W(x)$ which is calculated as the number of dendritic branches intersecting with a distance level x from the soma. The spatial resolution of the function $W(x)$ is 10 μm. For comparison, the plots include the experimental data for $B(x)$ and $W(x)$ for 29 L5 pyramidal cells extracted from [28] (denoted by L5 exp).

Table 1
Quantification of different axonal and dendritic morphological parameters of generated neurons (gen.) compared with experimental data (exp.)

Neuron type		L4	L4 star	L2/3	L5 A	L5 B
Number of dendrites	Gen.	7	7	7	7	6
	Exp.	6.3 ± 2.3	6.4 ± 1.7	6.6 ± 0.9	6.6 ± 1.7	5.4 ± 0.6
Total dendritic length (μm)	Gen.	3598 ± 341	6522 ± 791	13903 ± 3178	7281 ± 771	9568 ± 1662
	Exp.	2835 ± 901	4960 ± 2017	9500 ± 2000	5945 ± 1495	7853 ± 1583
Total number of dendritic segments	Gen.	760 ± 73	1383 ± 167	2992 ± 693	1553 ± 163	2049 ± 359
	Exp.	–	–	–	–	–
Membrane area of dendrites (μm^2)	Gen.	11748 ± 1065	15162 ± 1256	22856 ± 4927	13559 ± 1386	16868 ± 2632
	Exp.	–	–	–	–	22700 ± 7046
Total axonal length (μm)	Gen.	6411 ± 1913	6394 ± 1869	3338 ± 955	7354 ± 2148	7276 ± 2162
	Exp.	6405 ± 2427	–	2551 ± 253	7422 ± 1802	6582 ± 1724
Number of axonal branches	Gen.	25.6 ± 6.7	23.7 ± 6.1	11.5 ± 3.1	23.3 ± 6.1	23.4 ± 6.1
	Exp.	65 ± 45	–	13.3 ± 2.7	–	–

For the values generated by NeuGen 1000 cells of each neuron type are produced. The experimental values are found in [3,14,15,28,29,18]. All values are means ±SD.

Table 2
Averaged execution times and number of geometric synapses for generated networks with different numbers of neurons

Number of neurons	10	20	50	100	200	500	1000	2000	4000	5000
Number of synapses (2.5 μm)	45	71	379	1312	4863	30424	1.18e5	4.78e5	1.88e6	2.95e6
Number of synapses (1.0 μm)	2	6	26	71	320	1902	7420	3.05e4	1.21e5	1.90e5
Execution time (s)	2.0	2.6	4.3	7.5	13.8	35.4	74	166	379	485
L-Neuron (s)	1.3	2.4	4.8	8.4	12.6	38.4	84	180	387	497

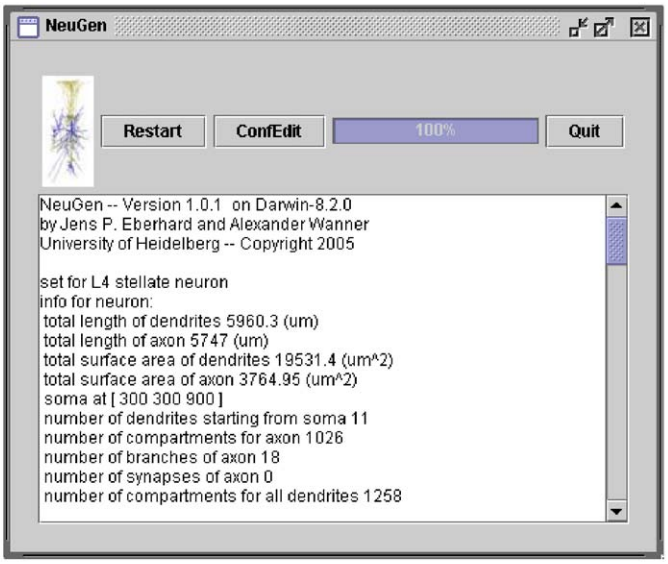
The networks are made up of 35% of L4 stellate neurons and L2/3 pyramidal cells, respectively, and 15% of L5 A and L5 B pyramidal cells, respectively. The threshold distance for generating a synapse is 2.5 and 1.0 μm, respectively. For comparison, the runtimes of the L-Neuron generator are given for randomly generated cells using the Hillman algorithm [1,2].

shown in Fig. 8 (right) as a distribution depending on the soma distance. The plots show the distributions for 4200 generated neurons by NeuGen. The graphs also include the available experimental data for $B(x)$ and $W(x)$ for L5 pyramidal neurons from [28]. The comparison with experimental data, given e.g. by [28], shows that the generated cells yield the characteristic morphological properties, i.e., a proximal plateau for basal dendrites and an increase due to the terminal tuft. However, the increase due to the tuft for the L5 A and L5 B pyramidal cells is smaller in $W(x)$ as the neurons in [28] are thick-tufted cells.

Further, Table 1 includes characteristic axonal and dendritic parameters of neurons. These morphological parameters are computed for 1000 neurons of each cell type by NeuGen and, in addition, the corresponding values given by the experimental data are shown. The comparison between the values of the generated neurons and the experimental values exhibits a good agreement within the given SD intervals. All extracted values match the specified experimental data except for the number of axonal branches. This number is limited by NeuGen's simplified implementation of the class `Axon` at the moment.

Table 2 includes the numbers of synapses due to spatial connections generated for networks with different numbers of neurons. The networks consist of 35 per cent of L4 stellate neurons and L2/3 pyramidal cells, respectively, and 15 per cent of L5 A and L5 B pyramidal cells, respectively. The threshold distance for generating a synapse is 2.5 and 1.0 μm, respectively. The numbers of generated synapses is smaller than found in real cortical columns. Using NeuGen's possibility of `nf_synapses` the number of synapses can be increased so that the total number of existing synapses in the network is similar to numbers found in experiments. The synapses of the type `nf_synapses` can be used then to simulate a stochastic input in single neurons as well as for the simulation of permanent stochastic input in the network.

Table 2 also compares the generation speed in averaged execution times on a personal computer (PC) for the increasing number of neurons in the generated networks. The table shows the runtimes for the generation and interconnection of 10 to 5000 neurons generated on a PC



```
NeuGen -- Version 1.0.1  on Darwin-8.2.0
by Jens P. Eberhard and Alexander Wanner
University of Heidelberg -- Copyright 2005

set for L4 stellate neuron
info for neuron:
 total length of dendrites 5960.3 (um)
 total length of axon 5747 (um)
 total surface area of dendrites 19531.4 (um^2)
 total surface area of axon 3764.95 (um^2)
 soma at [ 300 300 900 ]
 number of dendrites starting from soma 11
 number of compartments for axon 1026
 number of branches of axon 18
 number of synapses of axon 0
 number of compartments for all dendrites 1258
```

Fig. 9. The control user interface of NeuGen with the text output.

with a 2 GHz processor. The runtime increase of NeuGen is nearly linear in the growth of the number of neurons. Compared to the execution times of the L-Neuron generator, also given in Table 2, NeuGen shows a similar runtime behaviour.

4.1. Text output of NeuGen

During the runtime of NeuGen a text output is produced by the program. The text lists essential information about the constructed cells. The type of information for the output is determined by an 'output level' in the configuration for `OutputOptions.neu`. For 'level 0' a minimal output is given, and for 'level 2' NeuGen yields the detailed information about all generated neurons, i.e. the output includes the coordinates of the constructed compartments of the neurons. A sample 'level 0' output of NeuGen for the generation of a single L4 stellate neuron is shown in Fig. 9.

4.2. Optional features

NeuGen has some additional features which can be integrated into the software package by compiling the C++ main program anew. For instance, NeuGen provides a function to save the distribution of the generated branch points for all neurons of a network as a function of the distance. The distance is given by the axonal or dendritic length of the neurons in 3D. NeuGen also provides a function to write the width of the neural tree as a distance distribution (depending on the distance from the soma) for the neurons of the network as well as for a single neuron class. The function counts the branches of each neuron intersecting a sphere with a given radius from the soma. Both functions are able to scan either the dendritic tree, the axonal tree, or both for the selected neurons. The results for the dendritic trees are shown in Fig. 8.

The 3D geometric volume data that is represented by a generated neural network can be exported to a tagged image file format (TIFF). This file format is readable and processable by many graphical software programs.

5. Conclusion and outlook

We present the software NeuGen for the efficient generation of anatomically accurate neurons and neural networks. NeuGen is intended for scientists aiming at simulations of morphologically realistic networks in 3D. It is based on sets of descriptive rules that represent the axonal and dendritic geometry by morphological parameters. The generator stochastically samples parameter values from statistical distributions induced by experimental data. The range of the parameter values is given by the configuration. NeuGen generates non-identical neurons within morphological neuron classes of the cortex and synaptically connected neural networks in 3D. The neuronal structures can be converted into a 3D graphic format for visualization and into files for numerical simulations with the program NEURON.

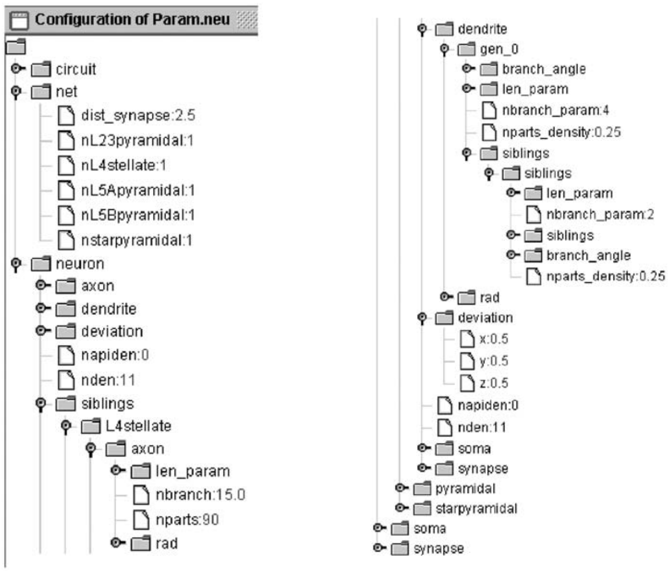
A graphical user interface is included in NeuGen which makes the program control and the assignment of values to the configuration parameters very easy. The current version also has an improved runtime behaviour resulting in an enormous speed-up for the interconnection algorithm and for the generation of the synapses. The interconnection has also been implemented in such a way as only neurons of particular cortical layers are interconnected.

In the future, it will be an important task to validate the stochastic model and the derived generation of neurons and neural networks against available databases of manually-traced neurons in 3D. In a subsequent paper we would like to demonstrate that experimental data and results, such as projection density fields or length density plots for coupled neurons, can be extracted from the synthetic neural networks of the generator.

NeuGen is an on-going software project which will be extended by additional features. It is available at http://neugen.uni-hd.de and runs on Linux, MS Windows and MacOS X platforms. According to NeuGen's class-oriented design it is well suited to support other cell types, e.g. smooth cells, which can be implemented in the future as subclasses of the neuron base class. Dendritic spines can also be implemented easily in the provided spaceholder class. However, due to their small extensions the OpenDX output for spines might be unsatisfactory.

Appendix A. Example for the configuration

A snapshot of the configuration user interface is shown with the configuration window for the file `Param.neu` in Fig. A1. It displays part of the parameters which determine the configuration of a L4 stellate neuron.



```
Configuration of Param.neu
circuit
net
    dist_synapse:2.5
    nL23pyramidal:1
    nL4stellate:1
    nL5Apyramidal:1
    nL5Bpyramidal:1
    nstarpyramidal:1
neuron
    axon
    dendrite
    deviation
    napiden:0
    nden:11
    siblings
        L4stellate
            axon
                len_param
                nbranch:15.0
                nparts:90
                rad
            dendrite
                gen_0
                    branch_angle
                    len_param
                    nbranch_param:4
                    nparts_density:0.25
                    siblings
                        siblings
                            len_param
                            nbranch_param:2
                            siblings
                            branch_angle
                            nparts_density:0.25
                rad
            deviation
                x:0.5
                y:0.5
                z:0.5
            napiden:0
            nden:11
            soma
            synapse
        pyramidal
        starpyramidal
    soma
    synapse
```

Fig. A1. Snapshot of the graphical user interface for the configuration of NeuGen.

A short description of the parameters follows.

Parameter	Short description
`dist_synapse`	Threshold distance for generating synapses
`nL23pyramidal`	Number of generated L2/3 pyramidal cells
`nL4stellate`	Number of generated L4 stellate neurons
`nL5Apyramidal`	Number of generated L5 A pyramidal cells
`nL5Bpyramidal`	Number of generated L5 B pyramidal cells
`nstarpyramidal`	Number of generated L4 star pyramidal cells
`nden`	Number of primary dendrites (basal and apical dendrites)
`napiden`	Number of apical dendrites
`noblique`	Number of oblique dendrites for the apical dendrites
`len_param`	Geometric length vector in multiples of the soma radius
`nbranch`	Number of branches of the axonal tree
`nparts`	Number of segments for the primary trunk of the axon
`gen_0`	First dendritic generation of the dendritic tree
`siblings`	Subsequent generations of the dendritic tree
`nparts_density`	Number of segments per μm with respect to the distance between start and end of a dendritic section
`branch_angle.max`	Maximal branching angle of the selected neuronal part
`branch_angle.min`	Minimal branching angle of the selected neuronal part
`rad.max`	Maximal radius (context-sensitiveparameter)
`rad.min`	Minimal radius (context-sensitive parameter)
`Rall.a`	Value of Rall's power
`Rall.c`	Threshold for the radius of Rall's power rule
`low_branching_limit`	Minimal distance from soma for non-oblique dendritic branches
`top_fluctuation`	Maximal relative fluctuation of the vertical extension for apical dendrites

The starting values for the seeds of the random number generators are set in the configuration file `Interna.neu`. With the aid of the separators `DXfile`, `HOCfile`, and `config_backup` in the file `OutputOptions.neu` it is possible to declare the names of the files to enable the writing of output files for the OpenDX visualization, for the simulation program NEURON, and for the backup of configuration values, respectively.

The initial values and ranges of the parameters are specified for each cell type below. These values are given by experimental data [14,8,13,15,3,21,28,18,29].

Parameter	L4 stellate	L2/3 pyramidal	L5 A pyramidal	L5 B pyramidal	L4 star
soma_position	$(P_{3l,6l}, P_{3l,6l}, P_{9l,l})$	$(P_{2.6l,4.5l}, P_{2.6l,4.5l}, P_{11l,2l})$	$(P_{3l,6l}, P_{3l,6l}, P_{7l,l})$	$(P_{3l,6l}, P_{3l,6l}, P_{4.5l,l})$	$(P_{2.5l,4l}, P_{2.5l,4l}, P_{9.5l,l})$
soma_radius	5 μm	5 μm	5 μm	5 μm	5 μm
dist_synapse	2.5 μm	2.5 μm	2.5 μm	2.5 μm	2.5 μm
nden	7	7	7	6	7
noblique	0	2	2	2	0
axon.len_param	(50, 50, 375) μm	(50, 50, 400) μm	(50, 50, 500) μm	(50, 50, 500) μm	(50, 50, 400) μm
axon.nbranch	26	12	24	24	24
axon.nparts	90	100	100	100	100
axon.rad.max	0.5 μm	0.5 μm	0.5 μm	0.5 μm	0.5 μm
axon.rad.min	0.1μm	0.1 μm	0.1 μm	0.1 μm	0.1 μm
nparts_density	$0.25\,\mu m^{-1}$	$0.25\,\mu m^{-1}$	$0.25\,\mu m^{-1}$	$0.25\,\mu m^{-1}$	$0.25\,\mu m^{-1}$
branch_angle.max	60	60	60	60	60
branch_angle.min	30	30	30	30	30
basal.rad.max	1.5 μm	1.5 μm	1.5 μm	1.5 μm	0.5 μm
basal.rad.min	0.5 μm	0.25 μm	0.25 μm	0.25 μm	0.25 μm
apical.rad.max	0	1.5 μm	1.5 μm	1.5 μm	2.5 μm
apical.rad.min	0	0.25 μm	0.25 μm	0.25 μm	0.5 μm
Rall.c, Rall.a	0.75, 1.5	0.75, 1.5	0.75, 1.5	0.75, 1.5	0.75, 1.5
low_branching_limit	0	20 μm	900 μm	700 μm	0
top_fluctuation	0	0.1	0.1	0.1	0

Appendix B. Algorithms for constructing branch parts

The description and detailed algorithms for constructing dendritic and axonal branch parts are given below. Therein corresponding lines and loops of the description and algorithm are indicated by (l1) to (l12). Further, in order to explain the generation of dendrites and axons we introduce the idea of a string. A string is defined by an unbranched connected chain of segments. For the construction of a string each algorithm uses a virtual random end point to determine the mean direction vector for the segments as well as the number of segments of the string (for dendrites).

In the following we denote by $rand_{x,y}$ a random value drawn from $P_{(x+y)/2,|x-y|}(z)$, by $rand^{e}_{x,y}$ a random vector drawn from $\mathbf{P}_{(x+y)/2,|x-y|,\mathbf{e}}(z)$, and by $\lfloor x \rfloor$ the largest integer value not larger than x.

B.1. Algorithm for constructing dendrites

A dendrite is constructed by generating a first string and, further, generating the branches which may consist of a hierarchical structure of branching strings (see *generateString*). Each branching string generation of this structure has its own defining parameter set for determining the geometrical properties such as angles, number of branches and so on. The radii of the segments of a dendrite are computed as follows. The radii at branch points are given by Rall's power rule (given by the variables Rall.c and Rall.a (l4,l6)). And the segments' radii along the strings are calculated by linear interpolation (l3). If the so calculated radii are below a given minimal value the radii are set to this threshold radius.

The input parameter dendrite_param includes the given configuration parameters for the dendrite.

```
generateBasal(dendrite_param, soma_position, soma_radius)
    set sr = soma_radius
    set dp = dendrite_param
    set gp = dp.gen_0
    set startp = soma_position
    set refl = gp.len_param * sr
    draw rand_direction from $\mathbf{P}_{\pi,2\pi}(z)$
    set refp = startp + $rand_{0.5,1.5}$* | refl | * rand_direction
    set rmin = dp.rad.min
    set rmax = dp.rad.max
    set zmax = dp.columntop
    generateString(gp, startp, refp, rmin, rmax, dp.Rall.c, dp.Rall.a, zmax, 1, dp.noblique, sr, false)

    generateString(genparam, startp, refpoint, minrad, maxrad, c, a, zmax, scale, noblique, soma_r, do_sh)
        if refpoint.z > zmax
            rotate vector (startp - refpoint) at soma so that refpoint.z = zmax                      (11)
        set nseg = $\lfloor$|startp-refpoint| * genparam.nparts_density$\rfloor$
        set limit = $\lfloor$genparam.low_branching_limit * genparam.nparts_density$\rfloor$             (12)
        set branchpoints[0] = $-1$
        for i in {1, ..., noblique}
            set branchpoints[i] = $\lfloor rand_{0,1}$*(limit $-1$)$\rfloor$
        for i in {noblique + 1, ..., noblique + $rand_{0.5,1.5}$* genparam.nbranch_param}
            set branchpoints[i] = $\lfloor$limit + $rand_{0,1}$* (nseg $-$limit $-2$)$\rfloor$
        sort branchpoints in ascending order
        set segment(0).startradius = maxrad
        for i in {1, ..., |branchpoints|}
            for j in {branchpoints[$i-1$] + 1, ..., branchpoints[$i$]}                                (13)
              set sr = segment(j).startradius
              set segment(j).endradius = sr $-$(sr $-$minrad)/(nseg $-$j $-1$)
              set segment(j + 1).startradius = segment(j).endradius
            set segment(branchpoints[i + 1]).startradius =
                                          c * segment(branchpoints[i]).endradius                     (14)
        for i in {1, ..., |branchpoints|}
            set bp = branchpoints[i]
            if i > noblique
              set genparam2 = genparam.siblings
              set scalefactor = scale
            else
              set genparam2 = genparam.oblique.gen_0
              set scalefactor = 1 $-$0.95*i/noblique
            if do_sh and i > noblique
              set scalefactor = scalefactor * (nseg $-$bp $-1$)/(nseg)                              (15)
            set len = |genparam2.len_param| * scalefactor * soma_r
            set min = genparam2.branch_angle.min
            set max = genparam2.branch_angle.max
            set direction = $rand_{min,max}^{segment(bp).direction}$
            set refpoint2 = segment(bp).endpoint + $rand_{0.5,1.5}$* len * direction
            generateString(genparam2, segment(bp).endpoint, refpoint2, minrad,
                         segment(bp).endradius*$(1-c^a)^{1/a}$, c, a, scalefactor,                  (16)
                         zmax, 0, soma_r, do_sh)
```

For the apical dendrites the algorithm implements three more features. First, an upper threshold distance for oblique branches can be given for the first string (variable `gen_param.low_branching_limit` (12)). Second, there is a maximal horizontal distance given by the first string generation (17). Therefore, the terminal tuft is generated by strings which may be above this maximal distance and which are rotated length-preservingly then to match this distance with the z-coordinate of their virtual end points (11). Third, the length of non-oblique strings are the more reduced the closer their starting points to the end point of the parent strings (15). This property defines a scaling factor which the parent strings inherit to their siblings (variable `scalefactor`).

```
generateApical(dendrite_param, soma_position, soma_radius)
    set sr = soma_radius
    set dp = dendrite_param
    set gp = dp.gen_0
    set startp = soma_position
    set refp = startp + $rand_{0.75,1.25}$*gp.len_param * sr
    set refl = gp.len_param * sr
    set zmax = startp + refl*(1 + $rand_{-1,1}$*dp.top_fluctuation)                    (l7)
    set rmin = dp.rad.min
    set rmax = dp.rad.max
    generateString(gp, startp, refp, rmin, rmax, dp.Rall.c, dp.Rall.a,
              zmax, 1, dp.noblique, sr, true)
```

B.2. Algorithm for constructing the axon

The axonal tree is constructed first by generating the first string (l8). Then, branching strings are generated and attached along the first string (l9). The lengths of these branches are limited by the length of the first string (l11). The axonal radii of the segments along the strings are then given by linear interpolation (l12). In addition, the radius of the first segment of a branching string is given by half of the start radius of the segment where the string is being attached (l10). Again, if the radii are below a given minimal value the radii are set to an axonal threshold radius.

The pseudo-code algorithm for constructing axonal branch parts is as follows. The input parameter axon_param includes the given configuration parameters.

```
generateAxon(axon_param, soma_position, soma_radius)
    set segment(0).startradius = axon_param.rad.max
    set sr = segment(0).startradius
    set segment(0).startpoint = soma_position
    set nseg = axon_param.nparts
    set aend = neuron_param.axon.end
    set inc = (aend −soma_position)/nseg
    set dev = axon_param.deviation
    for i in {0,..,nseg −1}                                                          (l8)
        set segment(i).endradius = sr −(sr −axon_param.rad.min)*(i + 1)/nseg
        set segment(i+1).startradius = segment(i).endradius
        draw direction from $\mathbf{P}_{0,2\pi}(z)$
        for j in {1,2,3} set direction[k] = dev[k] * direction[k]
        set segment(i).endpoint = segment(i).startpoint + inc + direction
        set segment(i+1).startpoint = segment(i).endpoint
    set nbranch = $\lfloor rand_{0.5,1.5}$*axon_param.nbranch$\rfloor$
    set last = nseg −1
    for i in {0,...,nbranch − 1}                                                     (l9)
        set bp = $\lfloor$nseg/3*(1 + ($rand_{0,1}$ + i)*2/nbranch) −1$\rfloor$
        set sr = segment(bp).startradius/2                                           (l10)
        set er = axon_param.rad.min
        set nbparts = $\lfloor$nseg*$rand_{0,1}\rfloor$ + 1                                  (l11)
        set segment(last).startradius = sr
        set min = axon_param.branch.angle.min
        set max = axon_param.branch.angle.max
        set binc = |inc| *$rand_{min,max}^{inc/|inc|}$
        for j in {last,...,nbparts + last − 1}                                       (l12)
            set segment(j).endradius = sr −(sr −er)*(j + 1)/nbparts
            draw dev from $\mathbf{P}_{\perp,inc}(z)$
            set segment(j).endpoint = segment(j).startpoint
                              + $rand_{0.5,1.5}$* binc + $rand_{-5,5}$* dev
            set segment(j+1).startpoint = segment(j).endpoint
      set last = nbparts + last
```

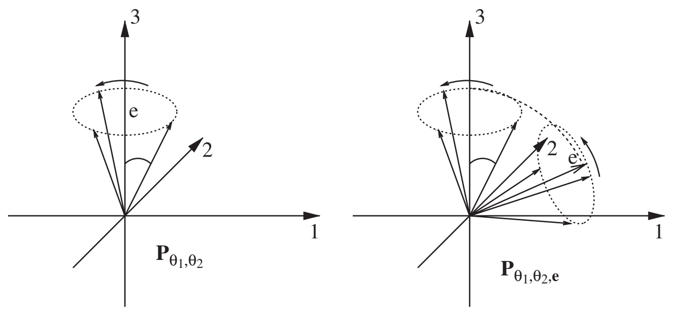

3
e
2
1
$\mathbf{P}_{\theta_1,\theta_2}$
3
e
2
1
$\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}$

Fig. C1. Illustration of the random vector distributions as described in the text.

Appendix C. Random vector distributions

An instance of the ensemble of unit random vectors around the axis $\mathbf{e}_3$ with a given angle θ reads $\mathbf{v} = (1, \theta, \phi)$ using spherical coordinates, and $\mathbf{v} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ using Cartesian coordinates, where the angle ϕ is drawn from the uniform distribution $P_{\pi,2\pi}(z)$. In addition to $P_{a,b}(z)$ we implicitly define the distribution function $\mathbf{P}_{\theta_1,\theta_2}(z)$ by the resulting ensemble of random vectors $\mathbf{v} = (1, \theta, \phi)$, where ϕ is drawn from $P_{\pi,2\pi}(z)$ and θ is drawn from $P_{\theta_1,\theta_2}(z)$. Further, for a direction $\mathbf{e}$ given in spherical coordinates by $\mathbf{e} = (1, \theta_e, \phi_e)$, we define the function $\mathbf{P}_{\theta_1,\theta_2,\mathbf{e}}(z)$ by the resulting ensemble of all random vectors $\tilde{\mathbf{v}} = (1, \tilde{\theta}, \tilde{\phi})$ given by $R_{\mathbf{e}}\mathbf{v}$. The vectors $\mathbf{v}$ are drawn from $\mathbf{P}_{\theta_1,\theta_2}(z)$ and $R_{\mathbf{e}}$ denotes the rotation matrix for rotating the vector $\mathbf{e}_3$ into the direction of $\mathbf{e}$, see Fig. C1. The rotation $R_{\mathbf{e}}$ is then given by $R_{\mathbf{e}} = R_3(\phi_e)R_2(\theta_e)$.The vector distribution function $\mathbf{P}_{\perp,\mathbf{e}}(z)$ is defined by the ensemble of random vectors $\mathbf{v}$ drawn from $\mathbf{P}_{\pi,2\pi}(z)$ which fulfil $\mathbf{v} \perp \mathbf{e}$ for a given direction $\mathbf{e}$.

The rotation matrices for rotating vectors with an angle α around one of the axes of the Cartesian coordinate system are given by (see [24]):

$$R_2(\alpha) := \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix},$$

$$R_3(\alpha) := \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

References

[1] G.A. Ascoli, J.L. Krichmar, L-neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology, Neurocomputing 32–33 (2000) 1003–1011.

[2] G.A. Ascoli, J.L. Krichmar, S.J. Nasuto, S.L. Senft, Generation, description and storage of dendritic morphology data, Philos. Trans. R. Soc. London B 356 (2001) 1131–1145.

[3] M. Brecht, B. Sakmann, Dynamic representation of whisker deflection by synaptic potentials in spiny stellate and pyramidal cells in the barrels and septa of layer 4 rat somatosensory cortex, J. Physiol. 543 (1) (2002) 49–70.

[4] P.J. Broser, R. Schulte, S. Lang, A. Roth, F. Helmchen, J. Waters, B. Sakmann, G. Wittum, Nonlinear anisotropic diffusion filtering of three-dimensional image data from two-photon microscopy, J. Biomed. Opt. 9 (6) (2004) 1253–1264.

[5] R.E. Burke, W.B. Marks, B. Ulfhake, A parsimonious description of motoneurons dendritic morphology using computer simulation, J. Neurosci. 12 (1992) 2403–2416.

[6] D. Feldmeyer, J. Lübke, R.A. Silver, B. Sakmann, Synaptic connections between layer 4 spiny neurone-layer 2/3 pyramidal cell pairs in juvenile rat barrel cortex: physiology and anatomy of interlaminar signalling within a cortical column, J. Physiol. 538 (3) (2002) 803–822.

[7] M. Häusser, B. Mel, Dendrites: bug or feature?, Curr. Opin. Neurobiol. 13 (2003) 372–383.

[8] M. Häusser, N. Spruston, G.J. Stuart, Diversity and dynamics of dendritic signaling, Science 290 (2000) 739–744.

[9] D.E. Hillman, Neuronal shape parameters and substructures as a basis of neuronal form, in: F. Schmitt (Ed.), The Neurosciences, 4th Study Program, MIT Press, Cambridge, 1979, pp. 477–498.

[10] M.L. Hines, N.T. Carnevale, The NEURON simulation environment, Neural Comput. 9 (1997) 1179–1209.

[11] K. Ichikawa, A modeling environment with three-dimensional morphology, a-cell-3d, and ca2+ dynamics in a spine, Neuroinformatics 3 (1) (2005) 49–64.

[12] N. Kalisman, G. Silberberg, H. Markram, Deriving physical connectivity from neuronal morphology, Biol. Cybern. 88 (2003) 210–218.

[13] M.E. Larkum, J.J. Zhu, B. Sakmann, Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons, J. Physiol. 533 (2) (2001) 447–466.

[14] J. Lübke, V. Egger, B. Sakmann, D. Feldmeyer, Columnar organization of dendrites and axons of single and synaptically coupled excitatory spiny neurons in layer 4 of the rat barrel cortex, J. Neurosci. 20 (14) (2000) 5300–5311.

[15] J. Lübke, A. Roth, D. Feldmeyer, B. Sakmann, Morphometric analysis of the columnar innervation domain of neurons connecting layer 4 and layer 2/3 of juvenile rat barrel cortex, Cerebral Cortex 13 (2003) 1051–1063.

[16] Z.F. Mainen, T.J. Sejnowski, Influence of dendritic structure on firing pattern in model neocortical neurons, Nature 382 (1996) 363–366.

[17] H.A. Mallot, F. Giannakopoulos, Population networks: a large-scale framework for modelling cortical neural networks, Biol. Cybern. 75 (1996) 441–452.

[18] I.A. Manns, B. Sakmann, M. Brecht, Sub- and suprathreshold receptive field properties of pyramidal neurons in layers 5A and 5B of rat somatosensory barrel cortex, J. Physiol. 556 (2) (2004) 601–622.

[19] B.W. Mel, Synaptic integration in an excitable dendritic tree, J. Neurophysiol. 70 (3) (1993) 1086–1101.

[20] M. Migliore, T.M. Morse, A.P. Davison, L. Marenco, G.M. Shepherd, M.L. Hines, Modeldb: making models publicly accessible to support computational neuroscience, Neuroinformatics 1 (1) (2003) 135–139.

[21] C.C.H. Petersen, B. Sakmann, The excitatory neuronal network of rat layer 4 barrel cortex, J. Neurosci. 20 (20) (2000) 7579–7586.

[22] C.C.H. Petersen, B. Sakmann, Functionally independent columns of rat somatosensory barrel cortex revealed with voltage-sensitive dye imaging, J. Neurosci. 21 (21) (2001) 8435–8446.

[23] P. Poirazi, T. Brannon, B.W. Mel, Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell, Neuron 37 (2003) 977–987.

[24] L. Rade, B. Westergren, Mathematics Handbook for Science and Engineering, Springer, Heidelberg, Berlin, New York, 1999.

[25] W. Rall, Branching dendritic trees and motoneuron membrane resistivity, Exp. Neurol. 2 (1959) 503–532.

[26] W. Rall, Core conductor theory and cable properties of neurons, in: E.R. Kandel (Ed.), Handbook of Physiology, vol. 1, American Physiology Society, Bethesda, 1977, pp. 39–97.

[27] B. Sakmann, Personal communication, 2005.
[28] A.T. Schaefer, M.E. Larkum, B. Sakmann, A. Roth, Coincidence detection in pyramidal neurons is tuned by their dendritic branching pattern, J. Neurophysiol. 89 (2003) 3143–3154.
[29] D. Schubert, R. Kötter, H.J. Luhmann, J.F. Staiger, Morphology, electrophysiology and functional input connectivity of pyramidal neurons characterizes a genuine layer Va in the primary somatosensory cortex, Cerebral Cortex 16 (2006) 223–236.
[30] S.L. Senft, G.A. Ascoli, Reconstruction of brain networks by algorithmic amplification of morphometry data, Lecture Notes in Computer Science 1606 (1999) 25–33.
[31] Senselab's model database (ModelDB), ⟨http://senselab.med.yale.edu/senselab/modeldb⟩.
[32] A. van Ooyen, J. van Pelt, Competition in neuronal morphogenesis and the development of nerve connections, in: G.A. Ascoli (Ed.), Computational Neuroanatomy: Principles and Methods, Humana Press, Totowa, NJ, 2002, pp. 219–244.

Jens P. Eberhard obtained a Diploma degree in Physics in 2001 and his Ph.D. in Mathematics in 2003 from the University of Heidelberg, Germany. He was a scholarship holder of the international graduate college "Complex Processes: Modelling, Simulation and Optimization". He works currently as a post-doc at the Interdisciplinary Center for Scientific Computing of the University of Heidelberg. His research interests include numerics of multiscale processes, upscaling, modelling and simulation of signal processing in neural networks, and software engineering.

Alexander Wanner is an undergraduate student of the faculty of mathematics at the University of Heidelberg, Germany. He currently works as a programmer for the NeuGen project, and he is working on his Diploma thesis about neuronal signal processing. His study interests are applied mathematics and software engineering.

Gabriel Wittum is professor of computational science at the University of Heidelberg. He holds a chair for Simulation in Technology. He received a doctoral degree in mathematics from Kiel University. His research focuses on modelling and simulation of problems from applied sciences and engineering. In particular he develops methods and software tools for the solution of problems from life sciences as well as engineering. He published about 100 scientific publications.