EvOL-Neuron: Neuronal Morphology Generation

Ben Torben-Nielsen *, Karl Tuyls, Eric Postma
MICC, University Maastricht
P.O. Box 616, 6200MD Maastricht, the Netherlands

Abstract

Virtual neurons are essential in computational neuroscience to study the relation between neuronal form and function. One way of obtaining virtual neurons is by algorithmic generation from scratch. However, a main disadvantage of current available generation methods is that they impose a priori limitations on the outcomes of the algorithms. We present a new tool, EvOL-Neuron, that overcomes this problem by putting a posteriori constraints on generated virtual neurons. We present a proof of principle and show that our method is particularly suited to investigate the neuronal form-function relation.

Key words: Virtual neuron, Neuronal morphology, Computational neuroanatomy

1. Introduction

Virtual neurons are digitized descriptions of biological neurons, with an emphasis on their morphology. In computational neuroscience their use is at least twofold. First, they compensate for the lack of vast amounts of morphometric data and are used for extensive modelling studies (1; 2; 3). Second, their synthesized nature enables the experimenter to have full control over morphometric parameters (4), which is required in the study of morphological effects on the neuronal function. In this article we present a new tool for the generation of virtual neurons: EvOL-Neuron.

An adequate description of neuronal morphology is required for studies investigating the influence of morphology on information processing capabilities of neurons. Up to date, three main methods exist to obtain realistic virtual neurons: tracing, algorithmic reconstruction, and generation from scratch (for a review see (1)). We argue that most of these methods suffer from the fact that they are biased by current biological knowledge

* Corresponding author.
Email address: b.torben-nielsen@micc.unimaas.nl (Ben Torben-Nielsen).

about neuronal morphology in the generation phase: the algorithms are devised in such a way that only 3D structures (i.e., virtual neurons) can be generated that reflect current insights and opinions. Intuitively, this seems to be a virtue but it seriously restricts the adaptivity of the generation algorithm to new biological detail (or evidence). Furthermore, current knowledge of neuronal morphology is too limited to claim that we know all data (i.e., properties or measurements) from which realistic virtual neurons can be generated. While several groups succeeded in reconstructing specific characteristics of neuronal morphology (e.g., dendritic branching patterns (5; 6)), it remains difficult to generate complete virtual neurons with both adequate topological (i.e., order and degree) and metrical (i.e., length and size) properties. More precisely, all variations on the morphometric properties make up a large *parameter space*. Existing generation methods specify the generation of virtual neurons in terms of these morphological properties. Consequently, the generated virtual neuron is always limited to combinations of known values of these properties, or to put it differently, limited to a small subset of the parameter space.

We propose a new methodology for generating virtual neurons that explores the immense parameter space for morphologies, that conform to current knowledge (rather than exploring specific parts of this space only). Our aim is to find an algorithmic description to generate virtual neurons that share the same morphological properties with a single (experimentally reconstructed) prototype neuron. L-Systems are used to generate candidate morphologies and Evolutionary Computation to guide the exploration in search for accurate morphologies. As mentioned before, existing methods for the generation of virtual neurons are limited to the generation of virtual neurons from a small part of the parameter space. We call this the *a priori limitation strategy* as the specification of virtual neurons is limited to combinations of statistically determined values of morphological properties. Thus, only a particular part of the parameter space is reached due to the limitation or bias inherent to the generation algorithm. Contrastingly, our method adopts a so-called *a posteriori constraining strategy* in which we consider all structures in the parameter space as a candidate virtual neuron, and explore this parameter space until a structure is found that complies with specific morphological properties.

Two main advantages of our method stand out. First, the exploratory capabilities of our method allow us to search the whole parameter space for structures that conform to current biological knowledge. This means that all possible outcomes are considered, and that the potential outcome is not limited in advance. It can be argued that a large potential parameter space is not advantageous as most of this space consists of biologically unrealistic morphologies. However, generally the exploration algorithm moves quickly to subsets of the parameter space where neuron-like structures can be found. This subset is not necessarily equal to the solution space of existing methods as EvOL-Neuron can find all subspaces of the parameter space where structures obey specific criteria. Second, our method is highly adaptive to new biological insights and evidence since we only need to update the exploration criteria. We do not need to redesign and implement a new algorithm to be in accordance to biological data [1].

[1] A drastic example would be the existence of trifurcations in neuronal morphologies. Despite the fact that in the current opinion about neuroanatomy trifurcations do not occur, several studies do report there existence (e.g., (7; 8)). As most generation algorithms are based on branching probability (9) the revelation of new biological detail might require a complete new design of most algorithms.

The remainder of this paper is outlined as follows. The next Section presents a brief overview of relevant techniques that are related to our method. Section 3 provides a detailed description of our method, and Section 4 is dedicated to how we select and validate generated virtual neurons. Results are presented in Section 5 and we conclude with a discussion in Section 6.

2. Related methods

The generation of morphologically accurate virtual (or synthetic) neurons has been studied for the past three decades. Advances in computational power in the last decade have boosted the power of these generation techniques.

A first step toward the generation of virtual neurons was taken by Hillman in 1979 who experimentally described *fundamental parameters* of neuronal morphology (10). Hillman concluded that seven morphometric properties were sufficient to describe (and classify) all types of neuronal morphologies, and, that realistic neuronal morphologies could be generated algorithmically by obeying these seven properties. More recently, a range of tools or methods based on Lindemayer systems (L-Systems) were introduced. The first applications of L-Systems for generation of neuronal morphologies can be traced back to Hamilton (11) and McCormick and Mulchandani (12). Both used an extended version of L-Systems; Hamilton used a dedicated grammar to support neuronal structures while McCormick and Mulchandani introduced stochastic L-Systems to generate neuronal structures. Despite the promising initial results with L-Systems it was not until the release of L-Neuron that the idea of L-Systems modelling to generate virtual neurons became popular. L-Neuron, a highly successful tool, was created by Ascoli and Krichmar (13). It relies on taking samples from density distributions (the so-called parameter-sampling) describing neuronal morphology as arguments for L-Systems to generate highly accurate and realistic virtual neurons. Several tools followed the same principle as L-Neuron and combined L-Systems with parameter sampling, i.e., Neuron PRM (14) and NeuGen (15). Both programs are designed to generate networks of morphologically realistic virtual neurons. All these methods rely on the a priori limitation strategy in which the specification of virtual neurons is limited to combinations of statistically determined values of morphological properties. As a consequence they are biased by the sparse data on neuronal morphology.

Our review of morphology generation tools is mainly restricted to those based on L-Systems. However, it is important to note that this is by no means an exhaustive review of generation techniques. Contrary to the use of L-Systems to describe (and generate from scratch) neuronal morphology, techniques exist that use a stochastic description generated by Monte Carlo models (16) or hidden Markov models (17). And contrary to generating morphologies from scratch, some techniques perform an algorithmic tracing of microscopic images (18) or raw anatomical preparations (19) to reconstruct neuronal morphology in a digitized way. A final type of generation method builds only particular branching patterns and are non-descriptive but include underlying mechanism in their modelling studies, e.g., (20; 5).

3. Methodology: EvOL-Neuron

We propose a new methodology to generate virtual neurons from scratch without putting a priori limitations on the candidate virtual neurons. The methodology is implemented in a computer program called EvOL-Neuron. The name of our method, EvOL-Neuron refers to our two-step methodology to generate virtual neurons: Evolutionary Optimization of L-Systems (EvOL). L-Systems are used to generate candidate 3D tree structures while Evolutionary Computation is used to optimize the accuracy of candidate virtual neurons by exploring the parameter space. The remainder of this Section explains the methodology and its implementation [2].

3.1. *Generation*

The generation part of EvOL-Neuron relies on a Lindenmayer system (L-System). L-System is a mathematical formalism of rule rewriting named after its inventor Aristid Lindenmayer (21; 22). Originally designed for describing the branched structures of plant morphology, it is highly suitable for describing virtual neuron morphology. The idea of L-Systems is simple yet powerful: an axiom defines the initial starting point, and the production rules define how to rewrite the axiom. In a cyclic fashion, all the symbols currently stored in the L-System are recursively rewritten according to the production rules. An illustration of the mechanism and its operation is given below.

axiom:	F[X]
rules:	F → YF
	X → BX
1^{st} cycle	YF[BX]
2^{nd} cycle	YYF[BBX]

In this example, the L-System consists of the alphabet $\{B, F, X, Y\}$; a single axiom containing the symbols $F[X]$, and two production rules $F \rightarrow YF$ and $X \rightarrow BX$, respectively. During the initialization phase, the L-System contains any string but the axiom, i.e., $F[X]$. The first rewrite cycle then substitutes all the symbols in the string with the corresponding production rule, i.e., F is substituted by YF and X by BX. As a result of the substitution, the L-System contains the string $YF[BX]$. The rewrite process is repeated a fixed number of times, resulting in a long string built iteratively.

A L-System in itself is nothing more than a way of generating a long string from a parsimonious description. In essence, L-Systems have no semantics. A meaning is given by the interpretation scheme. The interpretation scheme defines an alphabet of symbols and how they are interpreted to built a structure. Table 1 lists the alphabet used by EvOL-Neuron. This alphabet is a subset of the standard turtle geometry (23) interpretation scheme, and adapted to use with the Rotation-Elevation scheme; no special symbols are introduced for usage in neuroscience. In this scheme only two parameters are required to define every direction in 3D: the rotation angle on a plane, and the elevation angle on the

[2] A prototype of EvOL-Neuron can be downloaded at
`http://www.cs.unimaas.nl/b.torben-nielsen/evol-neuron/main.php?page=evol`.

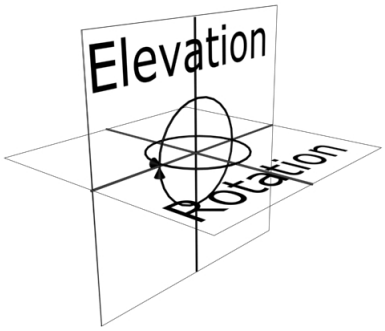

Fig. 1. Rotation-Elevation interpretation scheme. A point in 3D space is defined by the rotation, and the elevation on the plane orthogonally intersecting with the rotation plane.

Table 1
Alphabet used in the L-System used for generating neuronal morphologies

Symbol	**Purpose 3D**
$F(x)$	Move forward for x times the unit length, $x \in \Re^+$
$R(x)$	Adjust the rotation-angle with x degrees, $x \in [0, 360]$.
$E(x)$	Adjust the elevation-angle with x degrees, $x \in [0, 360]$.
[	Start a new branch. Put current position on the stack
]	End current branch. Pop position from stack

plane orthogonally intersecting the rotation plane (see Figure 1). Figure 2 illustrates the use of L-Systems for the generation of tree structures and how a structures develops with rewriting. It shows the structure generated by the L-System description presented in the inlay after 1 to 4 cycles. The illustrated L-system description has one axiom, `axiom_0`, and one production rule, `A`.

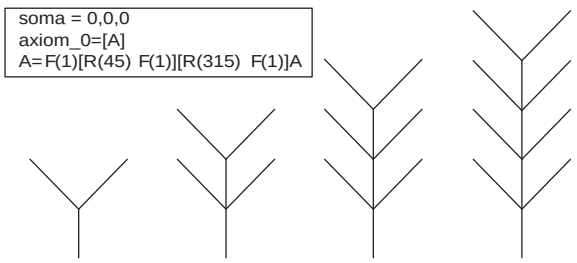

Fig. 2. Illustration of the geometric interpretation. The figures illustrate the development of an structure after rewriting cycle 1 to 4. The inlay lists the L-System description of the structures. In the description, `soma` defines the position of he soma; `axiom_0` is the first (and only) axiom; `A` is the first (and only) production rule. All symbols are explained in Table 1.

In theory, all rooted tree structures can be built in 3D using L-Systems equipped with the alphabet as listed in Table 1, if only arbitrary axioms and rules were allowed. Luckily, neuronal structures are bounded by a finite number of branching points and

thus have a limited order. In EvOL-Neuron we allow all 3D tree structures with an almost unlimited order as we do not put constraints on the length of axioms and rules. The output of EvOL-Neuron is an L-System description which is then interpreted to a SWC morphology file. SWC is a standard format to describe neuronal morphology (24). The interoperability with dedicated simulation software like NEURON (25) and Genesis (26) is assured by the publicly available tool CVAPP which can read SWC morphology files and convert them into the specific formats used by NEURON and Genesis.

3.2. *Optimization*

The optimization process is performed by Evolutionary Computation (EC). EC is a pragmatic programming method inspired by biological evolution (27; 28; 29) to explore large solution spaces. Different variations of the basic EC method exist; we employ genetic programming which is specifically tailored for tree-like structures as genomes (30) (cf. later).

EC exploits the principle of *survival of the fittest*. Candidate solutions to a certain problem (the so-called *individuals*) are encoded in individual genomes. A *population* is initialized with random individuals and every individual is then tested. The performance (*fitness*) of the individual for the task is assessed by means of a *fitness function*. Once the performances are assessed, the best individuals of a population are used for *reproduction*. Consequently, the new population will consist of descendants of the best performing individuals of the previous *generation*. This methodology proved to be powerful in problems with a considerable solution space (31; 32).

In this study, the subject of optimization is the matched accuracy of a generated 3D tree structure compared to a prototype (experimentally reconstructed) neuron. EC is defined by four essential characteristics: (i) genome encoding, (ii) evolutionary operators, (iii) fitness function, and (iv) the evolutionary parameter configuration. These four characteristics are addressed below.

3.2.1. *Genome encoding*

The genome encoding is an essential part of EC: it encodes the candidate solution for a specific problem that you want to optimize. In our case, L-Systems are used to describe neuronal morphology so the L-System description needs to be encoded. L-System descriptions can be highly complex (i.e., contain lots of symbols), and are hierarchical as a rule can be composed of branches, which are composed of symbols and symbols have arguments. These two properties make tree-encoding an efficient structure to use with the evolutionary operators (see below). Figure 3 illustrates the encoding, and the hierarchical relation in L-System descriptions.

3.2.2. *Evolutionary operators*

The evolutionary operators define how the *breeding process* creates a new population from the previous one. Three steps are associated with the creation of a new population: (i) selection, (ii) crossover, and, (iii) mutation. In biological terminology, selection ensures that the fittest individual is allowed to reproduce. Reproduction is the sexual reproduction where genes from both parents are transferred to the descendant. Mutation is the failure to make a perfect copy of a genome and ensures diversity.

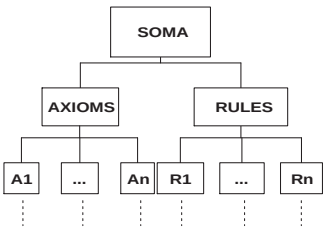

Fig. 3. Encoding of a L-System description by means of a tree.

A selection mechanism is used to pick individuals for inclusion in the breeding process. We use (μ, λ)-selection (33). λ represents the number of individuals in a population and μ is the number of individuals selected for the breeding process. Every generation, μ individuals are selected and λ descendants are generated from the selected individuals. The new λ descendants replace the old individuals, and a constant population size is maintained.

Crossover creates a descendant from two parents. We employ single-point crossover in which the descendant recombines exactly one part from both parents.

Mutation is the pseudo-random modification of an individual's genome before it is included in the new population. We employ four different mutation strategies. First, "argument mutation" alters the numerical argument of a symbol from the alphabet. Second, "symbol mutation" replaces a specific symbol in the genome by a different symbol while the argument remains the same. Third and fourth, "deletion" and "introduction" modify the genome by deleting or introducing a pseudo-random sub tree from/in the genome, respectively. when introduction is invoked, a sub tree of variable size is composed of symbols of the alphabet.

After invocation of the evolutionary operators the genome is checked for syntactic and semantic inconsistencies. For example, duplication of an axiom requires that the newly produced axiom receives a unique name.

3.2.3. *Fitness function*

The fitness function evaluates the comparison between the prototype and the generated neurons (see Figure 4) by assigning a so-called fitness value to the particular virtual neurons. The fitness value reflects the biological accuracy of the generated virtual neuron with respect to the prototype neuron. We define the accuracy by the accumulated distance in parameter space between real and virtual neurons. A good fitness value assignment is essentially the core of EvOL-Neuron and is described in detail in Section 4.

3.2.4. *Evolutionary parameter configuration*

The evolutionary configuration defines the exact implementation of an evolutionary optimization process. Values presented here are empirically chosen and proved to be advantageous in dummy experiments in which, for instance, only the size of the virtual neuron is optimized. We used a population size of 200 individuals. As an upper bound, the evolution was allowed to run for 1000 generations. In the selection procedure, μ represented the top 65% individuals. Crossover was invoked with a probability of 0.5, new individuals resulting from cross-over were not subject to any further mutation. The remainder of the new population was filled by individuals from the previous generation

and were subject to mutation: duplication probability of 0.12; deletion probability of 0.10; symbol-mutation probability of 0.05; and argument-mutation probability of 0.1.

A schematic overview of the methodology underlying EvOL-Neuron is illustrated in Figure 4. In summary, a real neuron is used as prototype. Virtual neurons are encoded as L-Systems, compared to the prototype and assigned a fitness. On the basis of the fitness value a new population of virtual neurons is made until they match with the biological neuron with respect to predefined properties.

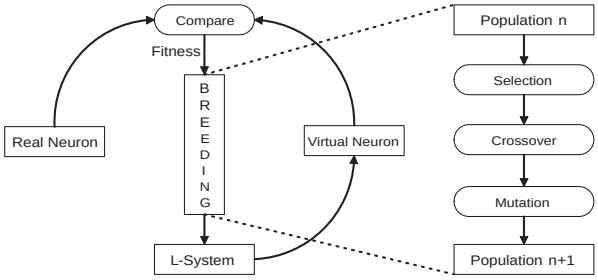

Fig. 4. Evolutionary Computation applied to the generation of virtual neurons. A population of L-Systems representing virtual neurons is generated and compared with (traced) real neurons. The comparison leads to a specific fitness assignment for each virtual neuron. Then, driven by a *survival of the fittest* mechanism a new population of L-Systems is created. This evolutionary cycle is repeated until the virtual-neuron data is indistinguishable from the real-neuron data with respect to predefined properties.

4. Validation of morphologies

The optimization process is guided by a correct assessment of the generated neuron. If a generated neuron is assessed to be a more accurate approximation of the prototype neuron, it will reproduce and gradually increase the overall accuracy of the whole population. The assessment is based on a quantitative comparison between morphological properties measured from the generated virtual neuron and the prototype. In this study we try to find good approximations of cat spinal cord α-motor neurons which we retrieved from the online archive NeuroMorpho (3). The original data originates from (34) and (35).

In a set of preliminary experiment we used morphological data from the literature to perform an assessment. Hillman's *fundamental parameters* (10) were used for this purpose. In these preliminary experiments we were able to optimize 3D tree structure so that their properties were in accordance to the fundamental parameters. Unfortunately, the fundamental properties proved to be insufficient to generate accurate virtual neurons (36): non biological structures that nevertheless agreed with Hillman's data were found. Therefore, we devised a custom fitness function. In this study we use 11 deliberately chosen properties to perform the assessment. With EC, there is a trade-off between the number of properties to optimize and the convergence of the fitness to a desired value. In other words, the more properties in the fitness function, the more difficult to find a solution. For this reason we tried to minimize the number of optimization criteria: number of stems (No_stem), percentage of bifurcations, stem rotation (T_rot) and elevation angle (T_elev), fractal dimension (FD, implemented as box-counting algorithm

on XY projection), number of bifurcations (No_bif), order, total length (L_tot), branch rotation (B_rot) and elevation (B_elev) angle and finally tropism ($TropsimF$). These properties are similar (but not equal) to the basic parameters used in L-Neuron (13). We need to include a measurement of the relative amount of bifurcations ("percentage of bifurcations") as we can also generate structures with n-furcations. However, currently n-furcations do not comply with current knowledge so we have to explore neurons that only have bifurcations. Next to these 11 properties we use the *heuristic* that structures must consist of at least 300 segments before they are considered candidates. This heuristic ensures a minimal level of complexity of the resulting virtual neuron. Structures without the required number of segment are assigned a negative fitness equal to $(-R+S)$, where R is the required and S the actual number of segments, respectively. Once a structure meets this first requirement it is assigned a real fitness value. The fitness function gradually increases the desired complexity of the generated virtual neuron by adding the next constraint to the fitness function only when the previous constraints are met. According to this scheme we optimize No_stem, percentage of bifurcations, $T\_rot, elev$ and FD. The remaining properties are optimized in parallel once a structure complies with all previously set criteria.

Values for the listed properties were computed and collected by dedicated computer programs written for this purpose. In total we implemented 43 morphological properties [3] . The properties can be roughly divided in two classes: single-variate and multivariate properties. Single-variate properties have a single value for a single neuron (e.g., total length), while multi-variate properties have multiple values for a single neuron (e.g., terminal lengths). We record all values for multi-variate properties which can then serve to compute distributional information. The properties which are not used in the optimization process can be seen as emergent properties of neuronal morphology and can be used to perform an adequate validation of the generated virtual neuron. As prototype we took the alphaMN3 cell, which had the following values: $No\_stem = 16$, $T\_rot = 27 \pm 104$, $T\_elev = -9 \pm 118$, $FD = 1.36$, $No\_bif = 35$, $order = 1.7 \pm 1.4$ (measured at every branching point and minimum and maximum 0 and 5, respectively), $L\_tot = 15765$, $B\_rot = -3 \pm 20$, $B\_elev = -4 \pm 31$ and $TropismF = 0.75 \pm 0.2$. In the optimization process we consider a property of the generated neuron in accordance to similar property of the prototype neuron when it falls in a *small* interval around the measured value of the prototype. The actual size of the interval is defined by the variance of the data measured from all prototype α-motor neurons.

5. Results

A series of ten evolutionary experiments is performed with a population size of 200 individuals. Eight runs obtained a best fitness of more than 80%, only two runs did not find anything more than the desired number of stems. A fitness of +80% means that most of the properties are optimized, *but not all* which implies that some features of thse generated neurons are not accurate. Three runs obtained a fitness of +99% which implies that these generated virtual neurons are in accordance with all the predefined properties. The three neurons that were in accordance with all 11 predefined properties

[3] A complete list can be found on:
http://www.cs.unimaas.nl/b.torben-nielsen/evol-neuron/main.php?page=evol

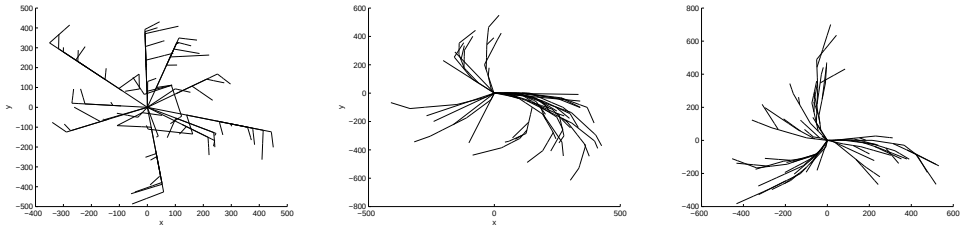

Fig. 5. α-motor neuron. Three generated neurons with +300 segments. All generated neurons obtained a fitness of $> 99\%$.

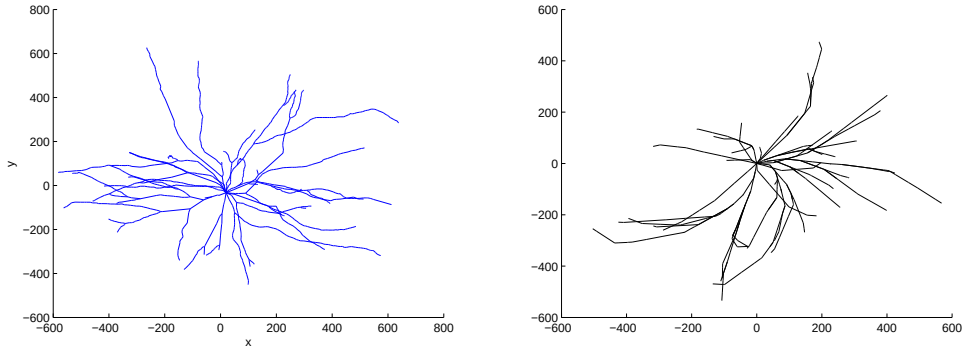

Fig. 6. α-motor neuron. Left: original experimentally reconstructed neuron. Right: generated neuron consisting of +600 segments.

are illustrated in Figure 5. The exact values of the optimized properties of the left-most neuron are: $No\_stem = 15$, $T\_rot = 19 \pm 106$, $T\_elev = 16 \pm 127$, $FD = 1.37$, $No\_bif = 40$, $order = 2.3 \pm 1.1$ (minimum and maximum order 0 and 4, respectively), $L\_tot = 15138$, $B\_rot = -1.4 \pm 16$, $B\_elev = 1 \pm 13$ and $TropismF = 0.81 \pm 0.2$. Especially the left-most neuron looks unnatural due to its "algorithmic appearance": structures repeating itself and a tendency to turn in one direction is clear. This is caused by the relative simplicity of the underlying L-System description. Therefore, we also did three evolutionary experiments in which the minimum number of segments was increased from 300 to 600. Out of these three runs, one run produced a $> 99\%$ fitness value. This neuron is illustrated in Figure 6 (right) together with the prototype neuron (left). The generated neuron has a strong similarity with the prototype neuron and has all quantitative and qualitive (i.e., visual) properties of a biological neuron.

The generated neurons are the product of an evolutionary optimization process. Figure 7 shows the development of the fitness during a complete evolutionary run of three. Both the best and average fitness of a generation are depicted. Note that in this figure the values below 0 are absolute values, and above 100 the values are scaled to a percentage. Both fitness values start at -300 due to the heuristic and grow rapidly to 0 which means they have reached the minimal required number of segments. The optimization process then tries to find 3D structures that match best with the predefined morphological properties. In the three illustrated cases, EC finds structures that are in accordance to the set criteria.

To illustrate that there is no specification about how virtual neurons should look like

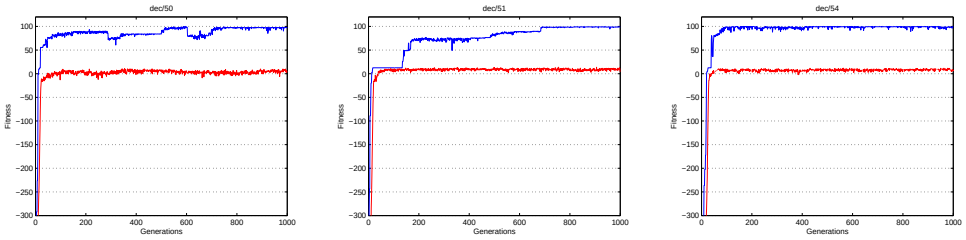

Fig. 7. Development of the fitness value of two evolutionary runs producing the neurons illustrated in Figure 5. The best fitness (top line) and the average fitness (bottom line) are displayed for each generation.

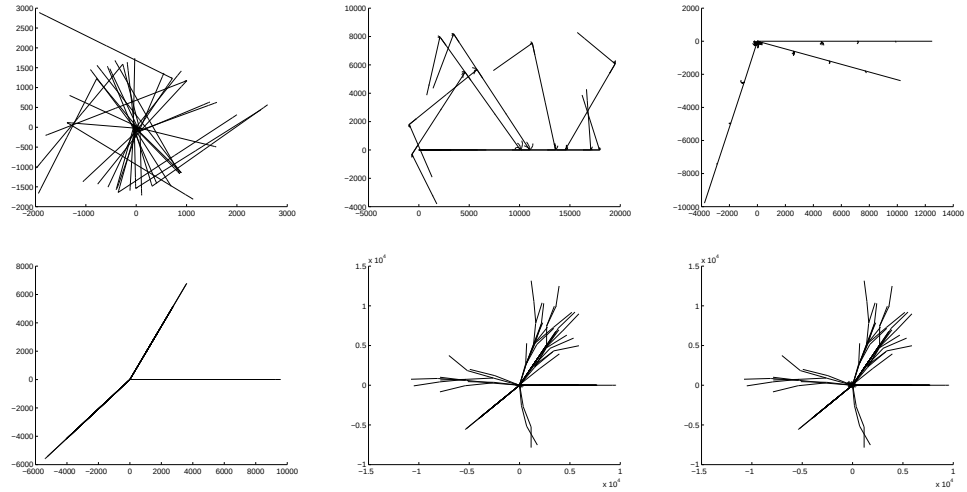

Fig. 8. Illustration of the mechanism underlying EvOL-Neuron. The top row illustrates the best structures after the first generation with fitness value > 0 (of the three successful runs). Clearly, these structures do not resemble neurons and hence reflect the unbiased nature of our method. Bottom row: best individual after 100, 400 and 600 generations taken from the second run (middle illustration in Figure 5 and this figure (top row)). Further explanation in the text.

in our program, we illustrate the best individuals in the first generation with a positive fitness taken from the three successful evolutionary runs. These structures are illustrated in Figure 8 (top line). It can be observed that these figures are all very different, and by no means resemble neurons. Nevertheless, starting from these structures EC can explore the parameter space and find successful structures. Figure 8 (bottom line) illustrates how the middle structure of the top line evolves to the successful structure illustrated in Figure 5 (middle). These structures are taken at intermediate steps in the evolutionary run, namely after generation 100, 400 and 600. It is clear that slowly biological features emerge in the evolving structures. This figure shows the process underlying EvOL-Neuron and how 3D structures gradually evolve to a desired morphology.

6. Discussion and conclusion

As mentioned in the Introduction, our method has two main advantages over other virtual generation methods and techniques. First, the implementation of the *a posteriori constraining strategy*. This strategy allows us the search the complete parameter space for virtual neurons that conform to current biological knowledge. In other words, we do not specify in advance what virtual neuron must look like; we only specify which properties it should obey. Existing methods implement the *a priori limitation strategy* in which the specification of virtual neurons is limited to a combination of specific values for the morphological properties. Second, the generation part of our method does not directly implement current biological knowledge into an algorithm which makes EvOL-Neuron highly adaptive with respect to new biological detail or evidence. Upon revealing new biological detail we need to include this data into the fitness function which guides the search process. On the contrary, existing methods might require a complete new algorithm to generate virtual neurons after the discovery of new biological details.

We identified two limitations of our method: realistic variability and the lack of guarantee to find "good" structures. At this moment we only approximate a single prototype neuron. Realistic variability of neurons within a specific class is not included yet. The other limitation is that the fitness function does not ensure a good result. A good match with the predefined set of properties of the prototype neuron is ensured, but a good match does not imply a realistic virtual neuron (see Figure 5(left)). In the future we want to investigate which properties are required to create a fitness that always correctly assesses generated structures.

One of the reasons to develop EvOL-Neuron was to investigate the neuronal form-

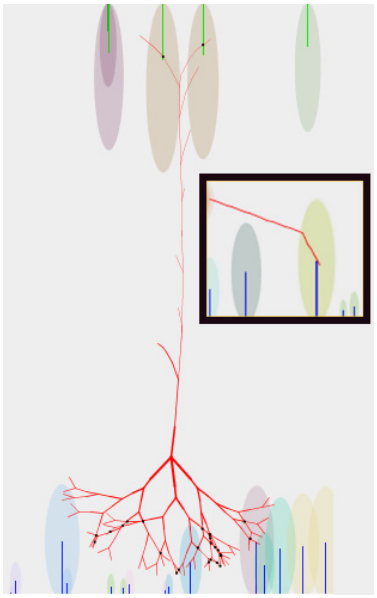

Fig. 9. Implemented experimental setup to perform an assessment based on functionality instead of morphological accuracy. This example illustrates a random setup and generated neuron.

function relation: what is the influence of neuronal morphology on the information processing capabilities of a neuron? Currently, we have no built-in functionality, i.e., to perform electro physiological experiments an external simulator like Genesis (26) or Neuron (25) needs to be used. However, we argue that EvOL-Neuron will prove very useful for the investigation of the form-function relationship. As outlined in Section 3 we generate virtual neurons and validate them with respect to a prespecified goal. At this moment this goal is morphological accuracy. In the future, we can also generate virtual neurons and validate them with respect to a specific phenomenological *functionality* of neurons. An example of such functionality can be a coincidence detector, and the required goal would be a specific activation pattern. Figure 9 illustrates a possible setup for this type of modelling. In this figure straight lines at the bottom and top indicate potential areas for synaptic connections: when a neuron has a certain proximity to such area a synaptic connection can be established. Dedicated simulators can then perform a detailed simulation and results from this simulation can be used to assess the fitness of a particular generated neuron. With this type of experiment we could investigate the neuronal form-function relationship in both directions. We can optimize a known morphology, and investigate the achieved functionality with this morphology. Alternatively, we can optimize a known functionality and analyse the morphology that supports this functionality.

Future enhancement of EvOL-Neuron can be achieved by constructing different fitness functions to increase the probability of successful evolution and decrease the chance of evolved but not natural structures. Also, the diameter of dendritic branches should be included in our model. This can be done either as a post processing step (i.e., algorithmically assign diameters to the evolved skeletons), or, as part of the evolutionary optimization.

To conclude, we can state that we succeeded in devising a new method for the generation of virtual neurons. We showed that EvOL-Neuron with its exploratory capabilities has several advantages over similar methods. Moreover, there is a strong indication that EvOL-Neuron is particularly suited for investigating the form-function relation in neurons.

Acknowledgements

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The authors want to thank Steven de Jong for help with some programming issues, and Ida G. Sprinkhuizen-Kuyper and the two anonymous referees for helpful comments that improved the manuscript.

References

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

[2] J. van Pelt, A. van Ooyen, H. Uylings, The need for integrating neuronal morphology databases and computational environments in exploring neuronal structure and function, Anat. Embryol 204 (2001) 255–265.

[3] G. A. Ascoli, Mobilizing the base of neuroscience data: the case of neuronal morphologies, Nature Neuroscience Reviews 7 (2006) 318–324.
[4] W. L. Kath, Computational Modelling of Dendrites, J. Neurobiol 64 (2005) 91–99.
[5] J. van Pelt, A. Schierwagen, Morphological analysis and modeling of neuronal dendrites, Math. Biosc. 188 (2004) 147–155.
[6] K. A. Lindsay, D. J. Maxwell, J. R. Rosenberg, G. Tucker, A new approach to reconstruction models of dendritic branching patterns, Mathematical Biosciences 205(2) (2007) 271–296.
[7] M. Kelly, U. Kuhnt, W. Wuttke, Morphological features of physiologically identified hypothalamic neurons as reveald by intracellular marking, Experimental Brain Research 34 (1979) 107–116.
[8] P. L. Edds-Walton, A. N. Popper, Dendritic arbors on the saccule and lagena in the ear of goldfish, *Carassius auratus*, Hearing Research 141 (2000) 229–242.
[9] D. E. Donohue, G. A. Ascoli, Local diameter fully constraints dendritic size in basal but not apical trees of ca1 pyramidal neurons, J. comp. Neurosci. 19 (2005) 223–238.
[10] D. Hillman, Neuronal shape parameters and substructures as a basis of neuronal form, Vol. The neurosciences, Fourth Study Program, The MIT Press, Cambridge, MA, 1979.
[11] P. Hamilton, A language to describe the growth of neurites, Biol. Cybern. 68 (1993) 559–565.
[12] B. H. McCormick, K. Mulchandani, L-System modeling of neurons, in: Proceedings of SPIE – Volume 2359, 1994, pp. 693–705.
[13] 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.
[14] J.-M. Lien, M. Morales, N. M. Amato, Neuron PRM: A Framework for Constructing Cortical Networks, Neurocomputing 52–54 (2003) 191–197.
[15] J. Eberhard, A. Wanner, G. Wittum, NeuGen: a toold for the generation of realistic morphology of cortical neurons and neural networks in 3d, Neurocomputing XX (2006) in press.
[16] R. Burke, W. Marks, B. Ulfhake, A parsimonious description of motorneuron dendritic morphology using computer simulation, Journal of Neuroscience 12(6) (1992) 2403–2416.
[17] A. V. Samsonovich, G. A. Ascoli, Statistical determinants of dendritic morphology in hippocampal pyramidal neurons: a hidden markov model, Hippocampus 15 (2005) 166–183.
[18] J. Evers, S. Schmitt, M. Sibila, C. Duch, Progress in functional neuroanatomy: Precise automatic geometric reconstruction of neuronal morphology from confocal image stacks, J. Neurophysiol. 93 (2005) 2331–2342.
[19] R. Scorcioni, G. A. Ascoli, Algortihmic reconstruction of complete axonal arborizations in rat hippocampal neurons, Neurocomputing 65-66 (2005) 15–22.
[20] A. van Ooyen, J. Diujnhouwer, M. W. H. Remme, J. van Pelt, The effect of dendritic topology on firing patterns in model neurons, Network: Computation in Neural Systems 13 (2002) 311–325.
[21] A. Lindenmayer, Mathematical models for cellular interactions in development i & ii, Journal of Theoretical Biology 18 (1968) 280–315.
[22] P. Prusinkiewicz, A. Lindenmayer, The algorithmic beauty of plants, Springer-

Verlag, 1990.
[23] H. Abelson, A. A. di Sessa, M.I.T. Press, Cambridge, 1982.
[24] R. Cannon, D. Turner, G. Pyapali, H. Wheal, An on-line archive of reconstructed hippocampal neurons., J Neurosci Methods. 84(1-2) (1998) 49–54.
[25] N. Carnevale, M. Hines, The NEURON book, Cambridge University Press, 2006.
[26] J. Bower, D. Beeman, The book of GENESIS: exploring realistic neural models with the general neural simulation system, TELOS, Santa Clara CA, 1998., 1998.
[27] J. Holland, Adaptation and Natural and Artificial Systems, University of Michigan Press, 1975.
[28] J. Koza, Genetic programming: On the programming of computers by means of Natural Selection, MIT Press, Cambridge, 1992.
[29] E. Boers, I. Sprinkhuizen-Kuyper, Combined Biological Metaphors, 2001, Ch. 6, pp. 153–183.
[30] C. Janikow, A methodology for processing problem constraints in genetic programming, Computers & Mathematics with Applications 32(8) (1996) 97–113.
[31] M. Mitchell, An Introduction to Genetic Algorithms, Cambridge, MA: MIT Press, 1996.
[32] C. A. Coello Coello, D. A. Van Veldhuizen, G. B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.
[33] T. Bäck, Evolutionary Algortihms in Theory and Practice, Oxford University Press, 1996.
[34] S. Cullheim, J. Fleshman, L. Glenn, R. Burke, Membrane area and dendritic structure in type-identified triceps surae alpha motoneurons, J Comp Neurol. 255(1) (1987) 68–81.
[35] F. Alvarez, J. Pearson, D. Harrington, D. Dewey, R. Fyffe, Distribution of 5-hydroxytryptamine-immunoreactive boutons on alpha-motoneurons in the lumbar spinal cord of adult cats, J Comp Neurol. 393(1) (1998) 69–83.
[36] B. Torben-Nielsen, K. Tuyls, E. O. Postma, Shaping realistic neuronal morphologies: An evolutionary computation method, in: International Joint Conference on Neural Networks (IJCNN2006), Vancouver, Canada, 2006.