Non-parametric algorithmic generation of neuronal morphologies

Benjamin Torben-Nielsen, Stijn Vanderlooy, Eric O. Postma

August 18, 2008

Abstract

Generation algorithms allow for the generation of Virtual Neurons (VNs) from a small set of morphological properties. The set describes the morphological properties of real neurons in terms of statistical descriptors such as the number of branches and segment lengths (among others). The majority of reconstruction algorithms use the observed properties to estimate the parameters of a priori fixed probability distributions in order to construct statistical descriptors that fit well with the observed data. In this article, we present a non-parametric generation algorithm based on kernel density estimators (KDEs). The new algorithm is called KDE-Neuron and has three advantages over parametric reconstruction algorithms: (1) no *a priori* specifications about the distributions underlying the real data, (2) peculiarities in the biological data will be reflected in the VNs, and (3) ability to reconstruct different cell types. We experimentally generated motor neurons and granule cells, and statistically validated the obtained results. Moreover, we assessed the quality of the prototype data set and observed that our generated neurons are as good as the prototype data in terms of the used statistical descriptors. The opportunities and limitations of data-driven algorithmic reconstruction of neurons are discussed.

1 Introduction

Accurate anatomical digital descriptions of neuronal morphologies, or virtual neurons (VNs), are rapidly gaining importance in the neurosciences. With the increasing availability of computational power it has become possible to simulate realistic neuronal morphologies (De Schutter and Bower, 1994) and even networks consisting of a large number of realistically shaped neurons (Markram, 2006). However, collections of non-identical VNs, which are required for network simulation, are hard to obtain and therefore scarce. The standard procedure for acquiring VNs is tracing or *experimental* reconstruction of neurons. Tracing can be done either with a lucid camera or with a semi-automatic reconstruction software tool such as NeuroLucida (Glaser and Glaser, 1990) or Neuromantic (Myatt et al., 2007). The tracing procedure is tedious and time consuming (Ascoli, 2007). For this reason, there is a considerable scientific interest in the *algorithmic* reconstruction[1] of neurons aiming at generating non-identical VNs

[1]For the sake of readability, 'generation' and 'reconstruction' are used as synonyms in the remainder of the text.

that are statistically indistinguishable from traced real neurons with respect to statistical descriptors (or measurements).

Currently, there are two main classes of algorithms to generate VNs from scratch: growth algorithms and reconstruction algorithms. Growth algorithms are used to test hypotheses about the mechanism underlying neuronal development. For this reason, growth algorithms are mechanistic models and 'grow' models in an abstract but plausible manner. There applicability is mostly limited to reproducing the topology of dendrites in so-called dendrograms (see, for instance, van Pelt and Schierwagen, 2004). However, these dendrograms are not sufficient to perform morphologically accurate network simulations. Reconstruction algorithms are descriptive algorithms and often do generate complete morphological descriptions. Our focus in this article is on reconstruction algorithms, in particular on the subclass of so-called *sampling algorithms* that sample from distributions of morphological measurements (Ascoli and Krichmar, 2000; Lien et al., 2003; Eberhard et al., 2007) to generate a VN. Note that the term 'growth' is used to denote a biologically plausible way of development. Sampling algorithms add long stretches of dendritic segments in a depth-first order. Therefore, sampling algorithms are considered to be descriptive reconstruction algorithms and not growth-algorithms.

A single parametric model is often too simple to capture the distributions of morphological measurements of real neurons (Burke et al., 1992; van Pelt and Schierwagen, 2004; Lindsay et al., 2007). The problems that arise from using a parametric model can be alleviated in two ways. The first way is by assuming that the morphological data are generated by a mixture of probability distributions of known form. However, fitting a finite mixture model to the data requires a priori knowledge of the number and types of distributions. Such knowledge is available for specific cell types only and may not even be valid. Consequently, mixture-model based algorithms can be applied to a limited number of cell types at best (i.e., pyramidal cells (Samsonovich and Ascoli, 2005b; Donohue and Ascoli, 2005; Eberhard et al., 2007), granule cell (Samsonovich and Ascoli, 2005a), motor neuron (Burke et al., 1992)). The second way to alleviate the simplicity is to use a cell-to-cell reconstruction strategy in which a single neuronal prototype is used instead of a set of neuronal prototypes (as in Samsonovich and Ascoli, 2003). A recently developed solution proposed by Lindsay et al. (2007) is to use a non-parametric reconstruction algorithm. Non-parametric models are statistical models that are constructed without making a priori assumptions about the distributions generating the real data. Their results are promising but the algorithm is still limited to reconstructing dendrograms of a single cell type. Our goal is to develop a successful approach in which several neuronal types can be generated *automatically* from a set of prototypical real neurons.

In this article we propose KDE-Neuron, a generic non-parametric reconstruction algorithm that suits the aforementioned purpose. Our approach is based on kernel density estimators (KDEs) to model the distribution of morphological properties. The algorithm is generic in the sense that it is not specifically tailored to fit a single cell type only. This feature originates from the non-parametric modelling of prototype data. KDE-Neuron offers the possibility to generate automatically representative cell types provided that sufficient morphological data are available. Additionally, the user can fine-tune the algorithm to provide a better fit with prototype data by setting several intuitive options. Setting the algorithm options (which coincide with algorithm parameters) does

not involve any adjustments of the algorithm itself in the sense that the modelling and sampling phases are left unchanged. Thus, KDE-Neuron is the first reconstruction algorithm capable of reconstructing different cell types without requiring modification of the algorithm itself. The KDE-Neuron algorithm has three advantages over other reconstruction algorithms: (1) no presumptions introduced in estimating the type and number of the distributions, (2) peculiarities in the biological data will be reflected in the VNs, and (3) the approach can reconstruct different cell types automatically.

We perform two sets of experiments in which we reconstruct motor neurons and granule cells, respectively. We assess the quality of the prototype data and show that the quality of certain properties is insufficient to construct reliable statistical models. Nevertheless, by application of a post-selection filter we successfully generate VNs that are statistically indistinguishable from the set of prototype neurons with respect to the assessed properties. Moreover, we will show that the goodness-of-fit between generated neurons and prototype neurons is highly dependent on the quality of the prototype data.

The remainder of this article is organised as follows. We start with a general outline of KDE-Neuron in Section 2. Subsequently, we present a detailed account of the different components of our algorithm in Section 3. The experimental setup containing the experimental procedures and the validation of results is explained in Section 4. In Section 5, the experimental results are presented and discussed in Section 6. Lastly, a conclusion is given in Section 7.

2 General outline of KDE-Neuron

Our proposed algorithm, KDE-Neuron, uses raw morphological descriptions in the SWC format (Cannon et al., 1998) as input. Subsequently, non-parametric statistical models of particular morphological properties contained in the raw data are obtained in the form of kernel density estimates (KDEs). By recursively sampling values from these KDEs and by combining the sampled values, we can reconstruct a virtual neuronal morphology. More specifically, such a combination of sampled values are used to construct a single dendritic segment. So, in each recursive step of KDE-Neuron a dendritic segment is added and the algorithm has to decide whether to terminate, bifurcate, or prolongate the current segment. The decision is made by inference from the KDE models. KDE-Neuron is an adaptation of the algorithm developed by Burke et al. (1992) and accommodates for non-parametric models and flexible inference from the models.

Figure 1 illustrates a schematic overview of the KDE-Neuron algorithm. Each numbered box indicates a main step in the algorithm. Here, we briefly go through all the main steps while in the next section we explain all components required to perform these steps. Before the actual reconstruction algorithm starts, univariate and multivariate KDE models of eight morphological properties and combinations of these properties are constructed, respectively (Step 0). The eight properties are listed in Table 1 (Type B). In Step 1, a sample is drawn from the appropriate KDE model in order to determine the number of stems in the reconstructed neuron. Step 2 uses three KDE models to sample the initial diameter (Dp) and two branching angles (rotation (R) and elevation (E), see Torben-Nielsen et al., 2008). In Step 3, a particular dendritic segment

is constructed and added to the virtual neuron. For this purpose, a length (L) and a new diameter (Dn) are sampled. Step 4 infers on the basis of the current segment and appropriate KDE models whether to bifurcate, terminate or prolongate the segment. If not terminating, one or two triplets containing the starting diameter of the next segment (Dp) and branching angles (R,E) are sampled in either Step 5 or 6. Step 5 requires two triplets for two daughter segments. Step 6 requires only a single triplet because the current segment is prolongated by addition of a new segment. If in Step 4 it is decided to terminate, it is checked whether there are still unfinished segments to complete. If all branches are completed, a single VN is finalized in the SWC format.

After reconstruction of a set of VNs, in Step 7, particular instances of the set of generated VNs that have biologically plausible properties are selected. The filtered set of VNs is then the final outcome of KDE-Neuron.

3 Detailed description of KDE-Neuron

The main steps outlined above are accommodated by six components: (1) the morphological properties of real neurons, (2) the KDE models of these properties, (3) the sampling algorithm for extracting VN properties from the KDE models, (4) the dendritic-segment generation algorithm, (5) the inference algorithm that decides to branch, prolongate, or terminate a branch, and (6) the post-selection filter. Note that these components are not equal to the main steps in the algorithm. Rather, each main step uses several of these components. The next six subsections describe the six components in detail. Additionally, a seventh subsection elaborates on the user-defined options by which KDE-Neuron can be configured.

3.1 Morphological properties of real neurons

A quantitative description of neuronal morphologies can be given by a wide variety of morphological properties that can be measured directly from a neuron description in the SWC format. Examples of such measured properties are the sets of basic and emergent properties (Hillman, 1979; Ascoli et al., 2001), excluded volume (da F. Costa et al., 2005), fractal dimensions (Fernández and Jelinek, 2001) and so forth. In this article we use a set of basic properties and emergent properties.

The basic properties are usually considered as a minimal set of morphological properties that is required to quantify, classify, and reconstruct neuronal morphologies (Hillman, 1979; Ascoli et al., 2001). We consider eight basic properties as listed in the top part of Table 1 (Type B). Despite the rationale behind basic properties, more measurements are required to quantitatively describe a full neuronal morphology. Therefore, we use an additional set of properties that emerge from the interaction between basic properties. An example of these so-called emergent properties is the overall length of a dendritic branch (L_tot) that emerges from several basic properties such as the number of stems (No_stems), the stem length (St_l_so), and individual segment lengths ($L\_Segments$). A list of used emergent properties is given in the bottom part of Table 1 (Type E).

We use a set of prototype neurons to collect measurements of basic and emergent properties for the KDE-Neuron algorithm. The number of measure-

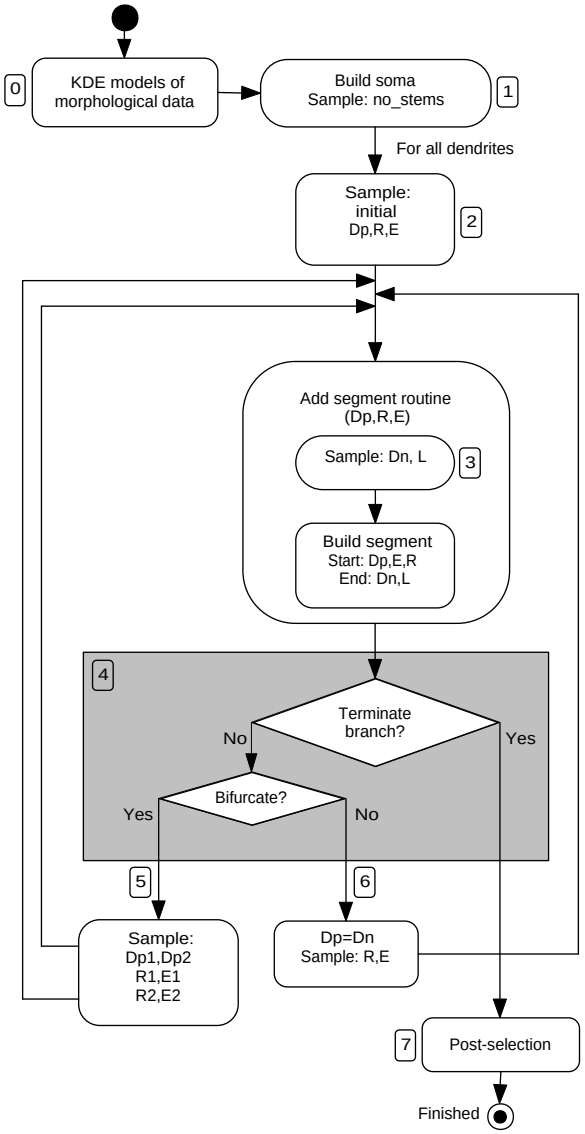

Figure 1: Schematic illustration of KDE-Neuron. The numbered boxes correspond to the main steps in the algorithm. In Step 0, the KDE models of morphological properties are constructed. A VN is then reconstructed by a combination of samples drawn from the KDE models. Such samples are taken in Steps 1, 2, 3, 5, and 6. In Step 4, it is decided whether to terminate, bifurcate or prolongate a segment. Step 7 is only applied after a set of neurons is reconstructed and automatically selects the biologically plausible reconstructions.

ments differs per morphological property because some properties occur more frequently within a neuron than others. For instance, there is only a single number of stems for each neuron, but there are multiple segment lengths in a neuron. The measured values for each property form a distribution over an

Name	Description	Type
no_stems	Number of stems	B
T_elev (initial E)	Stem elevation angle	B
T_rot (initial R)	Stem rotation angle	B
Stem_diam (initial Dp)	Stem diameter	B
B_elev (E)	Elevation angle at bifurcations	B
B_rot (R)	Rotation angle at bifurcations	B
St_l_so (L)	Stem length	B
L_biff (L)	Inter-bifurcation length	B
L_term	Terminal segment length	E
L_tot	Total dendritic length	E
L_Segments	Average all segment length	E
L_bif	Average inter-bifurcation segment length	E
No_bif	Number of bifurcations	E
Part_assym	Partition asymmetry	E
L_dist_avg_term	Euclidean distance from the soma to the terminal tips	E
L_path_avg_term	Path length from the soma to the terminal tips	E
Height_x	Size in X-dimension	E
Width_y	Size in Y-dimension	E
Depth_z	Size in Z-dimension	E
Order	Order of every dendritic compartment	E
Fractal_dim_xy	Fractal dimension of the X,Y projected morphology	E
Tropism_F	Ratio between segment length and distance travelled away from the soma.	E

Table 1: List of the basic morphological properties (Type B) and the emergent morphological properties (Type E). In the text we use the names in the first column. The names between brackets of the basic parameters are the abbreviations we use in Figure 1.

interval of admissible values. The nature of the distribution depends on the property and, more importantly, the neuronal type under consideration. Hence, as explained in the introduction, the use of a fixed parametrized distribution (or mixture of distributions) to model the unknown distribution of measurement values severely limits the applicability of the resulting reconstruction algorithms.

To overcome this problem, we model the measurements using KDEs, a non-parametric technique that does not make any assumptions about distributions generating the prototype data.

3.2 Modelling data with KDEs

At the core of KDE-Neuron are the non-parametric models that estimate the distributions of the morphological values used in the algorithm. Here, we explain how a distribution of morphological measurements is modelled by means of a KDE[2].

[2]Mathematical details related to this section can be found in the Appendix.

We assume that we have n morphological measurements from one of the morphological properties X, all measurements are independently drawn from an unknown underlying distribution (the iid assumption Bishop, 2006). In order to obtain the density of a specific measurement value x, we sum all other measurements x_i with weights that are decreasing with the distance between x and x_i ($i = 1, \ldots, n$). A so-called kernel function K establishes this weighting (smoothing) effect. The *kernel density estimate* $\hat{f}(x)$ is then defined as (Parzen, 1962):

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \quad , \tag{1}$$

where h is the bandwidth of the kernel. The value of the bandwidth determines the degree of smoothing. Clearly, a new measurement x that is close to many other points in the prototype data will receive many high contributions (kernel values) from these points and, as an end result, the new measurement receives a large value for the density $\hat{f}(x)$. A new measurement that is far away from any previously seen measurement will receive contributions modelled by the tail of the associated kernels, and consequently, it is assigned a low density. Although the principle is easy, the KDE has been shown to be very powerful; for more details we refer to (Silverman, 1986). A popular choice of kernel is the Gaussian. The estimator then models a distribution by centering a Gaussian at each of the n measurements and taking the average of the resulting densities at each point. More formally, the kernel density estimate becomes:

$$\hat{f}(x) = \frac{1}{nh\sqrt{2\pi}} \sum_{i=1}^{n} \exp\left[-\frac{(x - x_i)^2}{2h^2}\right] \quad , \tag{2}$$

where the bandwidth h corresponds to the standard deviation of the Gaussian. If h is set too small, then the estimate is too spiky and consequently contains a large amount of structure that is not present in the real distribution (undersmoothing). If h is set too large, then the estimate is too smooth and again fails to capture the shape of the distribution underlying the measurements (oversmoothing). An illustration of this effect is given in Figure 2. The proper setting of the bandwidth depends on the morphological property and the type of neuron. We compute the bandwidth automatically, as described in the appendix.

The KDE $\hat{f}(x)$ can only cope with univariate data. KDE-Neuron also features KDEs for multivariate model estimation, i.e., estimating the joint distribution of two or more morphological properties. For instance, in the bivariate case, such a distribution enables for determining the probability of a branch point given the current 'point' in a dendritic tree characterized by a path length and an order from the soma. Alternatively, one can use the marginal distribution that computes a distribution for one property independent of the value of the other property in the joint distribution. For example, what are the possible diameters for a segment independent of the path length from the soma. The mathematical treatment of multivariate KDEs is provided in the appendix.

3.3 Sampling from kernel density estimates

Once the KDE models are created, random samples taken from these models define the instantiations of VNs. Drawing random samples from a distribu-

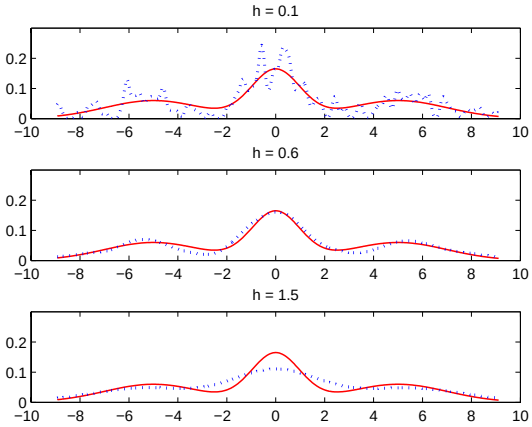

Figure 2: Illustration of the kernel density estimator in which a synthetical data set of 200 measurements is sampled from the distribution shown by the solid curve. Kernel density estimates are shown for various values of h by the dotted curve. If h is set too small (top panel) the result is too spiky, and if it is set too large (bottom panel) the result is too smooth. Only with the correct value of h the underlying distribution is estimated properly (centre panel).

tion is straightforward and exact when the distribution is parametrized, e.g., a widely spread method is the inverse sampling from a corresponding cumulative probability distribution (CDF) (Devroye, 1986). However, since KDEs are a non-parametric estimation technique, the question rises how to efficiently sample from it because the parametrized CDF is unknown.

Many sampling procedures have been proposed when parametric knowledge of the distribution is lacking (Bishop, 2006). KDE-Neuron employs a universally applicable procedure called *rejection sampling*[3]. We briefly explain the procedure. Assume a univariate KDE $\hat{f}(x)$ for the sake of simplicity. Also, assume without restriction a parametrized distribution $q(x)$ such that, when multiplied by a positive constant k, it encloses the KDE model. Formally stated, $kq(x) \geq \hat{f}(x)$ for all values of x. An illustration of such a distribution is given in Figure 3. The area under the solid curve defines the parametrized distribution. Within this area, the area under the dotted curve represents the KDE model.

Rejection sampling involves two steps. In the first step a number x_0 is drawn from $q(x)$. In the second step a random number u_0 is drawn from the uniform distribution over the interval $[0, kq(x_0)]$. If the condition $u_0 \leq \hat{f}(x_0)$ holds, then x_0 is accepted as a sample; otherwise it is rejected. Evidently, the procedure becomes computationally more efficient with increasing similarity between the distributions $kq(x)$ and $\hat{f}(x)$. For simplicity, in KDE-Neuron, $q(x)$ is defined as the uniform distribution. The value of k is set to the maximum density as evaluated by the KDE from the set of prototype properties.

The multivariate extension of rejection sampling is straightforward and requires a multivariate distribution q that encloses the multivariate KDE model $\hat{f}$. Conditional sampling corresponds to sampling from a 'slice' through the higher

[3]More advanced methods (in efficiency and effectivity) are possible but this is not the focus in the current work. We refer the interested reader to Ch. 11 of Bishop (2006)

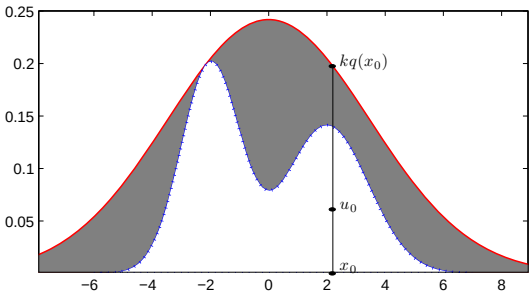

Figure 3: Illustration of the rejection sampling procedure. The KDE model $\hat{f}(x)$ is represented by the dotted curve. For an appropriate choice of k, the less complex distribution $kq(x)$ encloses the KDE model and is here represented by the solid curve. A pair of numbers (x_0, u_0) is generated and x_0 is said to be a sample from $\hat{f}(x)$ if the pair is not in the shaded area between the two distributions.

dimensional models. An example is sampling a proper segment diameter *given* the path length from the soma. In this case, the slice is taken at a particular value for the path length and results in a univariate distribution of the diameter conditioned on the path length.

3.4 The dendritic-segment generation algorithm

The segment-generation algorithm is motivated by the dendritic meandering observed in real neurons. Experimentally reconstructed neurons often display strong *contraction*, a measure of dendritic meandering (Scorcioni et al., 2004). Having a strong contraction means that individual segments are jagged. As a result, the ratio of path length over the Euclidean distance from the dendritic terminal to the soma is significantly larger than unity. Clearly, the overall morphology of a neuron is affected by the contraction. It might be that contraction as observed in traced neurons is an artefact of the tracing-process due to tissue shrinkage. However, the traced data is the only data we have access to and is considered to be the 'real' data in this study, i.e., if the data exhibits contraction than we have to mimic the contraction. Moreover, if the prototype data would not be considered to be the true data, every data-driven method and the following statistical analysis of the results would be in vein. Indeed, once the prototype data is not considered to be the true data, it is of no importance to show statistical similarity.

Therefore, contraction is a prominent feature of our segment-generation algorithm, and it is built in KDE-NEURON through *morphological compartmentalisation*: the division of a segment into a pre-specified number of compartments (conditioned on the segment length). The position and orientation of each segment is defined by its start-point and end-point (both in three dimensions). To simulate contraction, we sample the start-points of all but the first compartments from a Gaussian distribution centred on the end-point of the previous compartments. So, the start-point and the end-point of the segment itself remain fixed. The other points are altered where the degree of contraction depends on (i) the variances of the Gaussian for displacing the three-dimensional coordi-

nates, and (ii) the number of compartments in the segment. Figure 4 illustrates the simulated contraction of a segment.

It should be noted that our morphological compartmentalisation differs from the compartmentalisation used in physiological simulations (e.g., lambda_d compartmentalisation in NEURON (Carnevale and Hines, 2006)) which has an electrophysiological rather than a morphological significance.

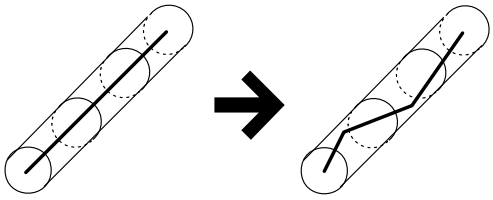

Figure 4: Illustration of morphological compartmentalisation of a segment in the segment-generation algorithm. Contraction arises by adding Gaussian noise to all intermediate segment points.

3.5 The inference algorithm

The inference algorithm decides to bifurcate, prolongate, or terminate a particular branch of a VN. We consider three decision procedures: Bayes, noisy Bayes, and typicalness. We begin by describing the two Bayesian procedures. Subsequently, we outline the typicalness decision procedure. The choice of the decision procedure depends on the type of neuron and the location within the dendritic branch. All procedures take three KDE models into account, one model for each of the distributions for branching, prolongating, and terminating compartments, respectively, that are obtained from measurements in the prototype set[4].

Bayesian decision procedures

The prior probabilities of bifurcation, prolongation, and termination are denoted by $f(S_{\text{bifur}})$, $f(S_{\text{prolong}})$, and $f(S_{\text{term}})$, respectively. The KDE models provide the class likelihoods $\hat{f}(x|S_{\text{bifur}})$, $\hat{f}(x|S_{\text{prolong}})$, and $\hat{f}(x|S_{\text{term}})$ where x is a particular compartment. Given these probabilities our interest is to decide for x to which type of compartment it belongs. We compute the posterior probabilities using Bayes' rule:

$$f(S_i|x) = \frac{f(S_i)\hat{f}(x|S_i)}{\hat{f}(x)} \quad , \tag{3}$$

with $i \in \{\text{bifur, prolong, term}\}$ and $\hat{f}(x)$ the probability of observing x.

The Bayes inference procedure is to make the decision associated with the largest posterior. It is known to be optimal in the sense that the number of incorrect decisions are minimized, assuming valid priors and correct class likelihoods (Bishop, 2006; Robert, 2007). It should be noted that for our purposes

[4]To avoid a pre-processing step, we directly work with compartments that are specified in the SWC format, rather than with segments.

the calculation of $\hat{f}(x)$ can be omitted since it is a constant term in Eq. 3. Nevertheless, the marginal probability is useful since it is the normalization constant ensuring that all $f(S_i|x)$ sum to one and are therefore valid probabilities. The disadvantage of this procedure is that in case where –given a particular point in a dendritic tree– two events are almost equally likely, the decision will always favour the most likely event. In other words, the reconstruction algorithm introduces an artefact in the reconstructed data (i.e., two events have different probability in the generated data while in the original data the two events have very similar probability).

The noisy Bayes procedure proposed by Lindsay et al. (2007) may overcome the disadvantage of the Bayes decision rule. In this procedure the posterior probabilities are used to subdivide the unit interval into three parts: $[0, f(S_{\text{bifur}}|x)]$, $[f(S_{\text{bifur}}|x), f(S_{\text{bifur}}|x) + f(S_{\text{prolong}}|x)]$, and $[f(S_{\text{bifur}}|x) + f(S_{\text{prolong}}|x), 1]$. Subsequently, a random number is drawn uniformly from the unit interval and the part associated with this number determines whether the segment bifurcates, prolongates, or terminates. The noisy Bayes inference procedure effectively prevents the occurrence of the aforementioned artefacts.

Typicalness decision procedure

KDE-NEURON mainly employs the typicalness decision procedure. This procedure treats the bifurcating, prolongating, and terminating compartments separately and assumes that branching and termination are the most important decisions (because of their effects on the resulting morphology). Moreover, due to the fact that we use prototype compartments rather than segments, the bifurcating and terminating compartments are always *outnumbered* by the prolongating compartments. Therefore sufficiently large samples for both compartments are difficult to obtain[5]. For a given compartment x, the typicalness procedure assesses the likelihood that it belongs to one of the three compartment types, i.e., how 'typical' the compartment is for the three types. The assessment proceeds as follows. Assuming that the type is 'bifurcation', we take a subset $T = \{t_1, \ldots, t_m\}$ from the set S_{bifur}. The typicalness of x for the bifurcation type is expressed as follows:

$$typ = \frac{\left| \left\{ i = 1, \ldots, m \;:\; \hat{f}(t_i \mid T) \leq \hat{f}(x \mid T) \right\} \right|}{|T|} \;, \tag{4}$$

where $\hat{f}(\cdot \mid T)$ is the outcome of the KDE-model of T. Similarly, we compute the typicalness of x being of the terminating type. The highest typicalness determines the decision to bifurcate or terminate, provided that the typicalness is equal or larger than a threshold value. Otherwise, the typicalness of x belonging to the prolongation type is computed. If the resulting typicalness is again falling below threshold, then the compartment becomes a terminating one; otherwise it becomes a prolonging one. At first this procedure might appear cumbersome but it follows the intuitive idea that the most typical decision is taken for a given point in the dendritic tree. Trying out each possible type in isolation from each other has been applied successfully in the field of machine learning (Bishop,

[5]Outnumbering occurs because every bifurcation compartment is surrounded by prolongating compartments, and terminating compartments are always preluded by prolongating compartments.

2006). The advantage of this procedure is that the decision to branch or terminate is not outnumbered anymore by the decision to prolongate. The typicalness procedure has no clear disadvantages for application in KDE-Neuron.

3.6 Post-selection of VNs

The previous five components are required to generate a VN. The post-selection filter is used only after a set of VNs is generated.

There are three reasons why a post-selection procedure is required. First, high variance in the prototype data in combination with a low number of prototypes leads to the fact that the sometimes non-representative samples are drawn because in order to construct a good statistical model there is large amount of prototype data required. And, once a non-representative sample is drawn, the next sample is likely to be non-representative again because the the condition of the next sample is non-representative. Or, in other words, once a bad sample is drawn, the next samples in that branch are likely to be equally bad and will result in a non-representative branch. Second, there is a big difference between the statistics associated with the set of generated VNs as a whole and the individual VNs in that set. It is possible (due to the reason outlined before) that individual VN are non-representative but that as a group they still lead to good result. For instance, different VNs can have a property that is incorrect, but the average of the individual incorrect properties can be perfectly consistent with the property as measured in the set. Thus, it is important to filter the individual VNs to assure that all individuals are at least plausible. Third, there is a conceptual difference between a local algorithm (that samples on local conditions) and a global algorithm (that samples on conditions set by the complete neuron). Indeed, a single generated branch can correspond statistically to a single branch from the prototype data, but this is not guarantee that a complete generated neuron which is composed of several branches fits with a complete prototype neuron.

To overcome these issues we take the measure of filtering the generated VNs so that only plausible individuals remain. The filter itself is straight-forward: properties of the generated neurons are checked whether they are within an interval of plausible values for that property as measured from the prototype data. Hence, filtering imposes relax constraints. A number of properties are tested in this way, and VNs that pass all tests are considered a valid outcome of KDE-Neuron. The exact properties used for the post-selection are listed in the Experimental setup section. Here, we have to remark that filtering of properties does not indicate that the algorithm in itself does not work.

3.7 User-defined options

KDE-Neuron has a number of user-defined options that offer the user to fine-tune the VNs generated by the algorithm. Table 2 lists the six options.

The first option determines the sampling of diameter values (i.e., Dn, $Dp1$, and $Dp2$) conditioned on the location within the dendritic tree. This option has five possible values, which are conditioning on the: (1) order, (2) degree, (3) path length from the soma, (4) Euclidean distance from the soma, or (5) diameter of the current point, i.e., the parent diameter. Combinations of the aforementioned values are also allowed. The second option defines the inference

procedure. The third option defines the conditioning of the KDE used in the chosen inference procedure; the values are identical to the values for the first option. The fourth option defines the threshold in the typicalness inference procedure. The fifth and sixth options define the number of compartments and the amount of Gaussian noise used to construct a segment.

It is important to note that these user-defined options do not modify the KDE-Neuron algorithm itself. The algorithm consist of three so-called 'neuroanatomical rules' (Ascoli, 1999) and these rules are unchangeable. The rules, however, do not prescribe which information they use and this is configured by the user-defined options. For instance, the rule describes that the probability of bifurcating is assessed, but does not define on the basis of which data this assessment is based. Thus, the user-defined options do not alter the algorithm itself and therefore the algorithm can be called 'general'. This strategy is in contrast to most existing approaches where new neuroanatomical rules were required to cope with different neuronal types (e.g., Ascoli et al., 2001).

#	**Setting**	**Description**
1	Diameter conditioning	Conditioning on either (1) order, (2) degree, (3) path length from soma, (4) Euclidean distance from soma, (5) parent diameter.
2	Inference procedure	Can be either (1) Bayes, (2) noisy Bayes, or (3) typicalness procedure.
3	Inference conditioning	Conditioning on either (1) order, (2) degree, (3) path length from soma, (4) Euclidean distance from soma, (5) parent diameter, or (6) all.
4	Threshold factor	The typicalness threshold value.
5	Contraction	Number of compartments within a segment.
6	Gaussian noise	The standard deviation of the Gaussian distribution for selecting end-point coordinates.

Table 2: Overview of user-defined options of KDE-Neuron that can be configured to obtain a good fit for specific cell types.

4 Experimental setup

The aim of the experiments performed with the KDE-Neuron algorithm is to evaluate its ability to create faithful VNs that are statistically indistinguishable from a set of prototype neurons. The experiments are performed with custom-built Java programs to extract morphological data, and with Matlab programs to construct the KDE models and generate VNs. In this section, we outline the experimental procedure and we explain the validation procedure.

4.1 Experimental procedure

We algorithmically reconstructed two types of neurons, namely motor neurons and granule cells. We decided for these two neuronal types because they are representative types. Moreover, they are often used in neuroanatomy and therefore using these particular types allows for straightforward comparison with other

techniques for generating neuronal morphologies. Below, for the two neuronal types we describe (i) the data on which the reconstruction is based and (ii) the values for the user-defined options in the experiments.

First, we outline the experimental setup for the motor neuron experiments. Data for the prototype set were taken from the NeuroMorpho archive (Ascoli, 2006, and `http://www.neuromorpho.org`) and are originally published by Cameron et al. (1991). The prototype set contains 22 cat spinal cord alpha motor neurons of which we selected 15. The remaining 7 neurons were discarded because they contained either axonal reconstructions or inconsistencies that conflicted with our data-acquisition program. Three typical motor neurons from the archive are illustrated in Figure 5 (top row). The user-defined options were as follows[6]. The KDE model for the diameter is conditioned on the path length from the soma. As inference procedure a hybrid of the noisy Bayes and typicalness procedure is used; only at orders less than two the typicalness procedure is applied. We opt for this hybrid procedure as the noisy Bayes appeared to be working the best but failed to make a correct decision at lower orders in preliminary experiments. The remaining options were configured as follows. The KDE models used for the inference procedure were conditioned on the order and path-length. The threshold factor (used in the typicalness procedure) was set to 0.2 for terminating and 0.1 for bifurcating. The segments were built with an odd number $((L/20) \times 2) + 1$ compartments[7] where L is the sampled value for the segment length, and the variances of the Gaussian used for the contraction were $\sigma_x^2 = 2$, $\sigma_y^2 = 0.5$ and $\sigma_z^2 = 0.5$. Twelve properties were used for the post-selection filter, namely: No_stems, St_l_so, T_elev, T_rot, $Fractal\_dim\_xy$, No_bif, $Order$, L_tot, $L\_Segments$, $L\_dist\_avg\_term$, $L\_path\_avg\_term$, and $Tropism\_F$. These values are selected heuristically as they proved to be appropriate in earlier work (Torben-Nielsen et al., 2008).

Second, the granule cells experiments were performed as follows. The data for the prototype set is also taken from the NeuroMorpho archive, and was originally published by Rihn and Claiborne (1990). This set contains 43 rat hippocampus granule cells which we all use as prototypes. Three typical neurons from this archive are illustrated in Figure 5 (bottom row). The user-defined options were identical to those of the motor neuron experiments with the following exceptions. The used inference procedure is the typicalness procedure, and the KDE models were based on the path length only. The post-selection filter was based on ten properties: except for T_elev and T_rot the filter was similar to the filter used for motor neurons. We removed the stem angles from the filter as they displayed too much variation in the prototype. The discussion addresses this issue.

4.2 Data quality and validation of experimental results

KDE-Neuron generates VNs from statistical models while making only a few assumptions about the data and about the desired neuronal morphology. Essen-

[6]We selected the options in a simple manner by trying out a few different settings in preliminary experiments. So, even though we will show good results with these options, there is still space for more fine-tuning. A grid search over the options can for example be used for an extensive parameter search.

[7]Electrophysiological simulations are more precise when an odd number of compartments is used (Carnevale and Hines, 2006).

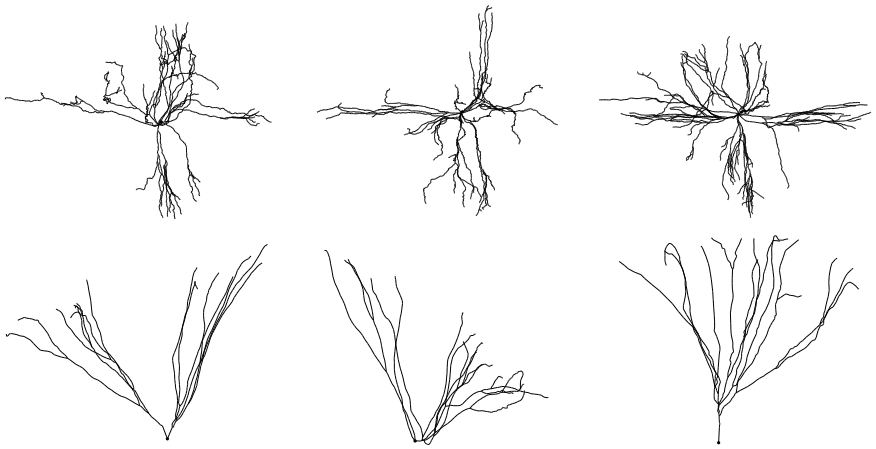

Figure 5: Illustration of prototype neurons taken from the NeuroMorpho archive. The neurons are displayed in an X,Y projection. Top: motor neurons. Bottom: granule cell. All illustrations are scaled to fit the page.

tially, we let the data speak for itself. It follows that the results generated with KDE-Neuron can at best be as good as the data itself. Two facts can decrease the quality of the prototype data. First, the large variation in real morphologies of the same neuronal type gives rise to a complex distribution of its properties. Second, the small number of prototype neurons. The combination of a complex distribution and small number of prototype neurons may hamper the quality of the statistical model because the observed samples can be a incomplete representation of the true distribution. Therefore, we need to assess the quality of the prototype data and validate the results obtained with KDE-Neuron with respect to the measured data quality.

Assessing the data quality and the validation of the results can be done both in the same manner, namely by measuring the similarity between *two* sets of samples for a single property. In case of the data quality assessment, the properties to be tested both originate from the prototype data. In case of the validation of the results, one set of samples is taken from the prototype data while the other set is taken from the generated virtual neurons. Independent of which data they are taken from, we note that each of the two sets represents a distribution of a specific property.

Similarity of the two distributions is tested with a hypothesis test, namely the Wilcoxon rank-sum test which is a non-parametric alternative to the standard two-sample t-test (Lehmann and Romano, 2006). The null-hypothesis is that the two samples are generated from an identical distribution. An outcome of 0 or 1 indicates that the test is accepted or rejected (with 5% significance level). Applying only a single test is often inconclusive, and therefore a scheme of repeatedly performing such a test is used by applying bootstrapping (Witten and Frank, 2005; Alpaydin, 2004). In this scheme, the two samples used for testing the null-hypothesis are repeatedly selected at random (with substitution) from the complete data set. Then, the significance of the number of acceptances and rejections is determined with a sign test.

The data quality can now be defined as a distribution that has a non-significant number of rejections in the bootstrap scheme. If the number of rejections is significant, then the data quality is defined to be poor. We perform 100 bootstrap tests with 5 and 20 prototype neurons in the sample for motor neurons and granule cells, respectively. With 100 tests and the 5% significance level, a number of at least 62 rejections implies that the two samples differ significantly and, that therefore the data is of poor quality. Less than 62 rejections implies that the data is of good quality. With respect to the quality assessment, we note that a good data quality indicates that the distribution is suitable to construct a model from. Inversely, from poor quality data it is impossible to construct a reliable model without an abundance of data points. With respect to the validation of results, a non-significant number of rejections indicates a high similarity between the prototype data set and the generated data set. Therefore, the VNs can be considered to be representative for the prototype neurons, based on a solid statistical foundation.

By using the bootstrapping scheme in combination with the Wilcoxon rank-sum test and a final conclusive sign test, we assess both the quality of the data and validate the experimental results. In addition to this type of quantitative validation, a visual inspection or qualitative validation of the generated results is performed (since both types of validation are necessary; one good result does not imply the other).

5 Results

Below we present the results obtained by generating two types of neurons. We remark that all presented results are biologically plausible with respect to the tested properties because of the post-selection filter. Here, we define plausible as within the range of observed values in the prototype data. The results and subsequent analysis of the result are mainly intended to show that the VNs are not only biologically plausible, but but are statistically *similar* to the set of prototype neurons. Thus, that properties of the sets of prototype neurons and generated neurons are distributed similarly.

5.1 Motor neurons

As a first step, we assessed the quality of the aforementioned archive of motor neuron data. The assessment is summarized and visualized in Figure 6 (left). Bootstrapping was performed with two pools containing five randomly selected neurons, and was repeated for 100 times. The X-axis lists the tested properties: (1) B_elev, (2) B_rot, (3) No_stems, (4) L_bif, (5) St_l_so, (6) L_tot, (7) No_bif, (8) $Part\_assym$, (9) $L\_dist\_avg\_term$, (10) $L\_path\_avg\_term$, (11) $Depth\_z$, (12) $Width\_y$, (13) $Height\_x$, (14) $L\_Segments$, (15) L_term, and (16) $Order$. These tested properties correspond to the 12 properties of the post-selection filter added with 4 properties which are commonly used in the validation of results (see for instance Ascoli et al., 2001; Samsonovich and Ascoli, 2005b). The bars in Figure 6 represent the total number of rejections of the Wilcoxon rank-sum test; the line at $y = 62$ indicates the significant number of rejections at the 5% significance level. It can be observed that three properties (viz. $L\_dist\_avg\_term$, $L\_path\_avg\_term$, and $Order$) have a signifi-

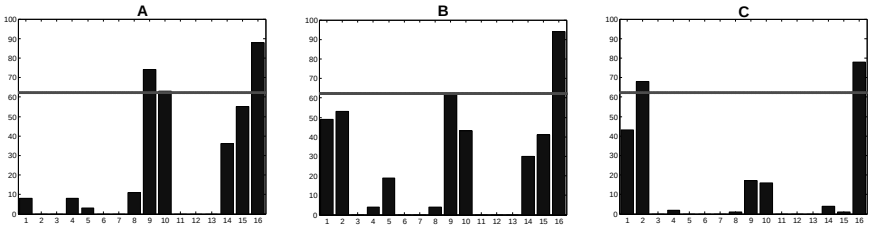

Figure 6: Bootstrapping applied to different samples of motor neuron data. A: quality assessment of the motor neuron prototype data. B: similarity test between prototype neurons and generated VNs. C: assessment of the generated data. The bars represent the total number of rejections of the Wilcoxon rank-sum test for each tested property. A value of 62 or more is significant (represented by the horizontal solid line). The X-axis represents the tested properties: (1) B_elev, (2) B_rot, (3) No_stems, (4) L_bif, (5) St_l_so, (6) L_tot, (7) No_bif, (8) $Part\_assym$, (9) $L\_dist\_avg\_term$, (10) $L\_path\_avg\_term$, (11) $Depth\_z$, (12) $Width\_y$, (13) $Height\_x$, (14) $L\_Segments$, (15) L_term, and (16) $Order$.

cant number of rejections indicating poor data quality of these properties. The remaining properties have less rejections and can be considered to be of good quality. Bearing the quality measurement in mind, we can already state that it is nearly impossible to find a good fit between the poor-quality prototype data and generated data for the three associated properties.

We generated 1000 motor neurons with KDE-Neuron, of which 271 were selected by the post-selection filter. The generated motor neurons rejected by the filter were mainly rejected due to an erroneous fractal dimension (87% of rejections), and to a lesser extent by the number of bifurcations (6%) and total dendritic length (4.5%). The few remaining rejection were caused by different tested parameters. Six examples are illustrated in Figure 7. These visualisations allow for a qualitative validation of the KDE-Neuron algorithm. All six examples show that the VNs exhibit three main visual characteristics of motor neurons: (1) several dendritic branches growing away from the soma and branching extensively in a sphere-like manner, (2) bifurcations close to the soma, and (3) dendrites without bifurcations.

A quantitative validation of the results is performed using the bootstrap procedure of which the outcomes are illustrated in Figure 6 (B and C). The bar chart on the right quantifies the data quality of the generated data set. It can be observed that the properties in the generated data seem to be of good quality because generally the bars are much smaller than the threshold value of 62. This can be explained as follows. We generate a large amount of neurons from statistical models and repeatedly sample from the same model (regardless of its quality). Consequentially, the model will be faithfully reproduced by the many samples drawn from it. Therefore, the variance is generally going down and the quality of the generated data, as expressed in the rank-sum test, improves. Two properties have a significant number of rejections and are thus based on poor quality data, viz. B_rot and $Order$. The $Order$ has a similar level of quality in the original data so this makes a good comparison with the prototype data.

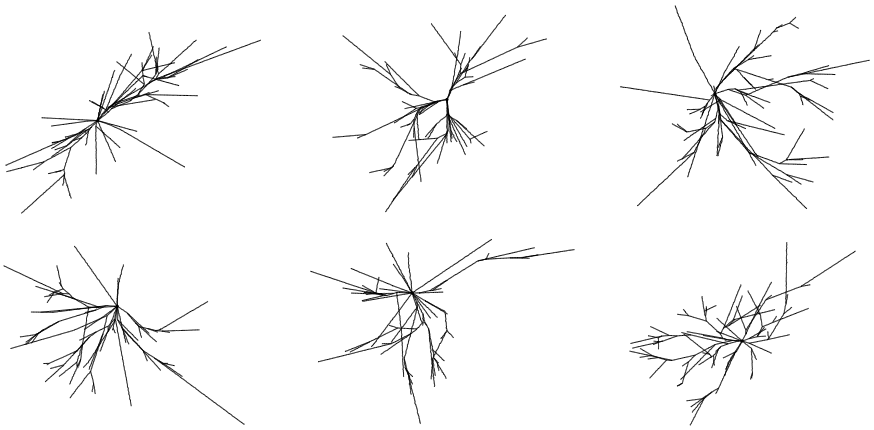

Figure 7: Illustration of six generated virtual motor neurons projected on the X,Y plane. These neurons are representative for the 271 successfully reconstructed motor neurons.

The *B_rot* however, appears to be of good quality in the prototype data and hence indicates a potential mismatch between prototype data and generated data. Potential mismatches can be investigated by checking the results of the bootstrap scheme with two samples from both prototype and generated neurons (Figure 6, B). If the number of rejections for a specific property is not significant, we may conclude that the data exhibits similarity between prototype and generated data. This is also the case for *B_rot* and thus there is no mismatch; only the quality of the generated data shows a higher degree of incoherence. Continuing, it can be seen in the Figure 6 (B) that there are a few cases where the hypothesis test is rejected, and thus indicating a good match between the prototype data and the generated data.

However, only two properties have a significant number of rejections, namely for *L_dist_avg_term* and *Order*. These two are part of the list of properties that were predicted to be impossible to model because of the poor quality of the prototype data. All the other tested properties have a significant number of acceptances, which indicates a strong similarity between the prototype neurons and generated VNs.

As a final visual verification, we also show a box-plot in Figure 8 with the tested properties of both the prototype neurons and generated VNs. The X-axis denotes the properties and the Y-axis the scaled value of each property. Properties are coupled with the observation from the prototype set ('P' in the label) and generated set ('G' in the label). The boxes have lines at the lower quartile, median and upper quartile. Whiskers extend to the most extreme value within 1.5 times the interquartile distance from the end of the box. It is clear that the range for each post-selection property of the generated data lies inside the range of the prototype data. In addition, it can be seen that although the exact number of values might be different, the distribution statistics depicted by the box plots are largely similar; even for properties based on poor-quality data. In some cases, the prototype data displays a large number of outliers while the generated data has larger interquartile distances and therefore less outliers, for instance, the branching angles (labelled '1_P' and '1_G'). This visual discrepancy

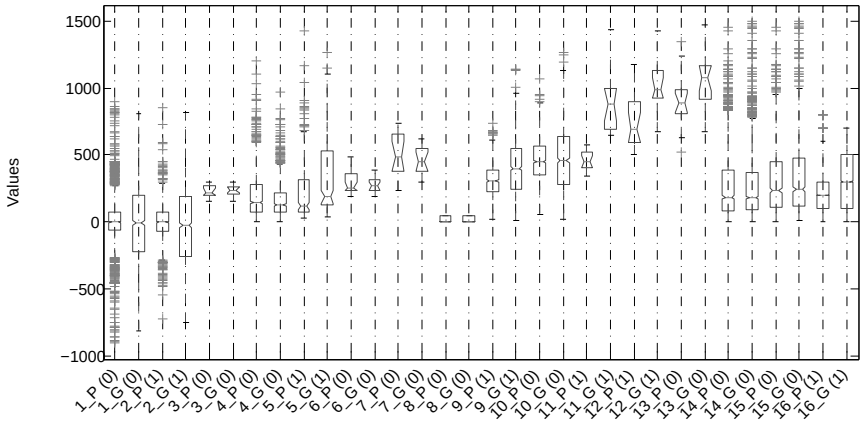

Figure 8: Box-plots illustrating the distribution of the tested properties for the prototype data ('P') and generated data ('G'); the numbers correspond with the property numbers in Figure 6. More details in the main text.

is caused by a different number of values in the prototype set and generated set.

The different validation strategies indicate that the motor neurons generated by KDE-NEURON fit well with the prototype neurons.

5.2 Granule cells

The granule cell experiments were conducted in a similar fashion as those of the motor neurons. We start again by assessing the data quality. Figure 10 (left) illustrates the outcome of the bootstrap scheme performed with two pools of 20 neurons (both from the prototype data) and 100 repetitions. The quality is highly similar to the quality of the motor neuron data. The same set of properties has a significant number of rejections: *L_dist_avg_term*, *L_path_avg_term*, and *Order*. In addition to the quality assessment, preliminary experiments showed that reconstructing the branching angles was problematic: the statistical models of the stem angles and branching angles show a substantial amount of variation. Figure 9 (C) and Figure 10 (property 1 and 2) illustrate this observation. However, upon inspection of the real neurons it was found that granule cell have a strong tendency to grow in a single direction. Consequentially, there must be a higher order relation between the branching angles to correct for pronounced stem angles. For instance, the environment could provide clues to the developing neuron which we cannot include into our algorithm due to limited prototypes. Therefore, we had to fix the models of the *B_rot* and *B_elev*. Without the fixation of the branching angles, none of the generated VNs passed the post-selection filter.

We generated 1000 virtual granule cells with KDE-NEURON, and after post-selection 108 VNs remained. The generated granule cells that were removed from the results were rejected due similar reasons as in the case of the rejection of generated motor neurons, i.e., mainly due to an erroneous fractal dimension (78% of rejections) and the number of bifurcations (15%). The remaing rejec-

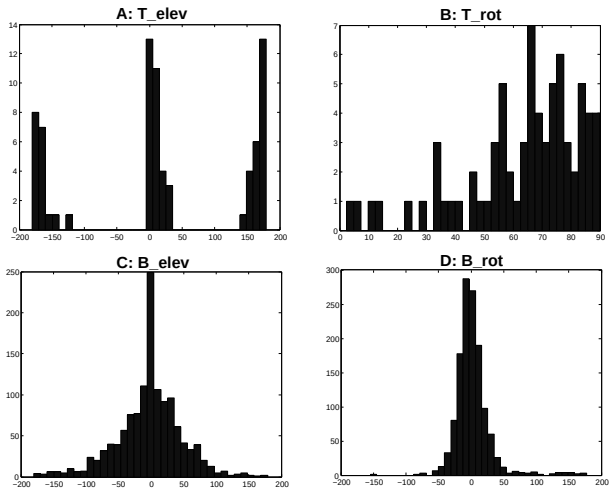

Figure 9: Histograms indicating the strong variation in the stem and branching angles. The top row represents the stem angles (A and B), while the bottom row represents the branching angles (C and D). In the left column the elevation angles are shown (A and C) and the right column shows the rotation angles (B and D). The x-axis represents the angle in degrees while the y-axis represents the number of occurrences. Real granule cells, however, do look very uniformly indicating higher order relations between the branching angles in single branches. Due to limited prototype data our algorithm did not 'discover' these relations and we had to fix the distributions of the angles.

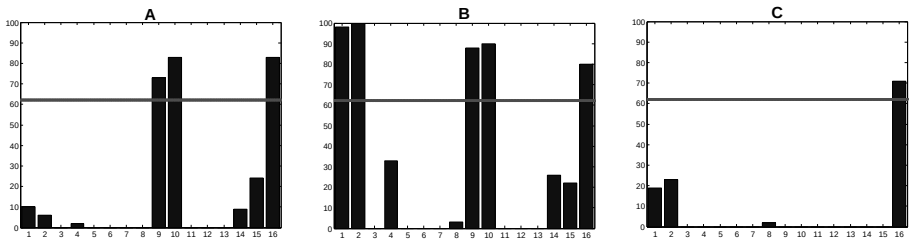

Figure 10: Bootstrapping applied to different samples of granule cell data. A: quality assessment of the granule cell prototype data. B: similarity test between prototype neurons and generated VNs. C: assessment of the generated data. Illustration is analogous to Figure 6.

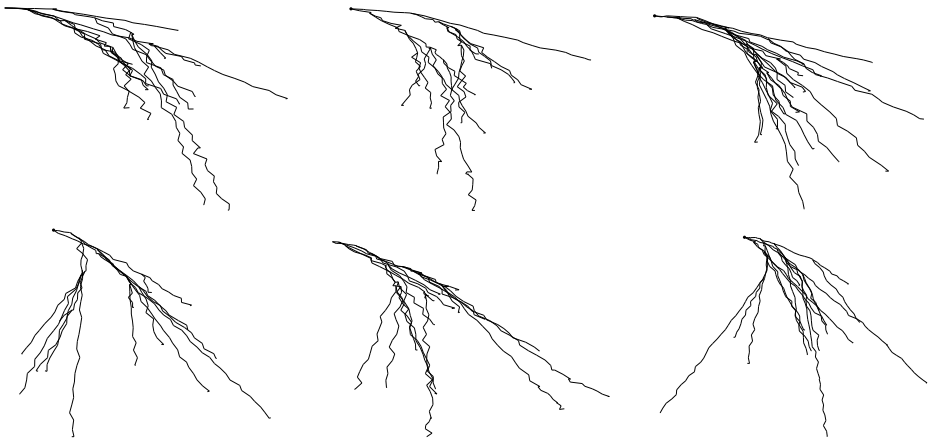

Figure 11: Illustration of six generated virtual granule cells projected on the X,Y plane. These neurons are representative for the 108 successfully reconstructed granule cells.

tions are due to an inappropriate total dednritic length and number of stems[8]. Six of these granule cells are displayed in Figure 11. The fact that we had to fix the branch-angle distributions is clearly visible from Figure 10: the quality of the generated data is good (Figure 10, C) but the coherence between the generated data and prototype data is low (Figure 10, B). Contrary to most real granule cells, there is no difference between the branching angles close to the soma and the branching angles that are located more distally. However, due to the post-selection filter, the Euclidean distance and the path length between the soma and the terminals remains plausible. With the exception of some branching angles, the generated VNs exhibit typical characteristics of granule cells, viz. one or two stems that quickly branch and prolongate without further branching. Lastly, the contraction of dendritic segments also appears highly realistic.

A quantitative validation of the results is again performed with the bootstrap scheme. When looking at the generated data set (Figure 10 (C) we observe that only data for the *Order* is of poor quality, which is a similar observation as in the prototype data. What concerns the similarity between the generated data and the prototype data (Figure 10, middle), we observe that the data of the same three properties as in the prototype data are of poor quality. Since the prototype data of these particular properties was of poor quality as well, this outcome is the best one that we can get. A real mismatch is indicated by the significant number of rejections for the branching angles (B_rot and B_elev). This mismatch is caused by fixing the branching angles rather than using a statistical model (for the aforementioned reason). The bootstrap scheme applied to the prototype data and the generated data provides similar results as the motor neuron results: properties that were of poor data quality in the prototype data provide the worst fit. However, this fit is the best possible fit

[8]The low number of filtered neurons can be attributed to the fixed distribution of branching angles because, in order to pass the filter, a particular sequence of branch angles and segments lengths is required. For instance, the filter tests the Euclidean distance between soma and terminal tips, which is entirely defined by the angles and lengths of a branch.

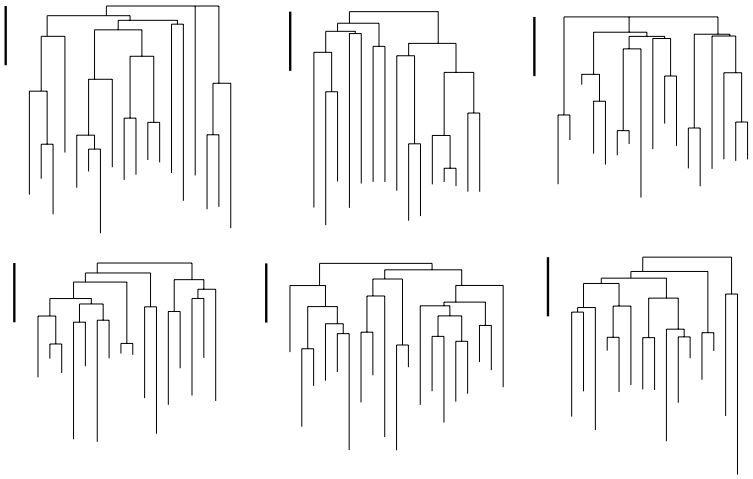

Figure 12: Dendrograms of prototype granule cells (top) and generated virtual granule cells (bottom). The dendrograms have a high resemblance.

given the data quality.

As a final verification, in Figure 12 we show the dendrograms of three prototype granule cells and three generated virtual granule cells, top and bottom row respectively. Clearly, there is a considerable resemblance between the prototype dendrograms and the dendrograms from generated VNs. This observation is verified by comparing the number of stems, bifurcations, and the fact that the bifurcations occur at different path lengths from the soma (and sometimes fairly close to the terminals).

From the qualitative and quantitative validation of the virtual granule cells generated with KDE-Neuron, we can state that the generated VNs fit well with the prototype data. A point of concern is the branching angle, which is further elaborated on in the discussion.

6 Discussion

First, we compare KDE-Neuron to other reconstruction algorithms. Then, we interpret the results and argue that KDE-Neuron gets the most out of the data, and that the performance is upper bounded by the availability of good quality data. Lastly, we discuss possible future lines of research.

6.1 Comparison with other approaches

The reconstruction results obtained with KDE-Neuron show its viability to generate VNs with realistic morphologies. In this subsection, we compare our algorithm to related sampling reconstruction algorithms. The comparison is guided by the three advantages of KDE-neuron, viz. (i) faithful non-parametric modelling of morphological parameters, (ii) independent of cell type, and (iii) generation of full 3D reconstruction.

First, non-parametric models allow for accurate modelling of an arbitrarily complex distribution provided that representative prototype data is available. These models have the advantage over parametric models that they do not require a priori assumptions about the distribution(s) generating the prototype data. Indeed, they can capture the full sample of observed data without requiring a priori knowledge of the number and types of distributions (Wand and Jones, 1995). Moreover, assumptions about the distribution underlying the samples can obscure rare data points, or even recurring peculiarities in the data. In contrast, non-parametric techniques let the data speak for themselves and therefore take each data point into account with a priori equal influence on the model to be learned. Because VNs are instantiations formed by combinations of samples from the statistical models, it is of great importance to have an appropriate model. Thus, we can state that non-parametric models are the best choice. Currently, the approach developed by Lindsay et al. (2007) is the only other approach using KDE models. However, this approach does not comply with the two other advantages of our algorithm (listed below).

Second, KDE-Neuron provides a generic algorithm that is able to reconstruct morphologies of different neuronal cell types. The results of both the reconstructed motor neurons and granule cells are obtained by the same algorithms. The non-parametric algorithm from Lindsay et al. (2007) requires particular preprocessing steps and this leads to a decrease in flexibility and hence specialisation for one cell type. In L-Neuron and related algorithms such as NeuroPRM (Lien et al., 2003) and NeuGen (Eberhard et al., 2007), new neuroanatomical rules or correction factors need to be introduced to generate VNs that fit well with a range of different cell types. Consequentially, L-Neuron is not generic because the algorithm needs to be updated to work with new different cell types. In KDE-Neuron all required neuroanatomical rules are used, and can be configured by specifying a few options. Some other proposed algorithms are also highly specific for particular cell types. For instance, an algorithm by Samsonovich and Ascoli (2005b) takes the soma and stems of a traced neuron and only afterwards the algorithm reconstruct the remainder of the granule cell.

Third, our method aims at the generation of full 3D morphologies. The VNs generated with KDE-Neuron are all unique and fit well with the morphological properties as observed in a prototype set. Several other algorithms including L-Neuron (and related methods) generate full morphologies. However, these algorithms do not have the advantage of using non-parametric models.

From these comparisons, we may conclude that we succeeded in our objective to generate sets of reconstructed neurons that are statistically similar to a prototype set with respect to the tested properties. No other approach is currently able to combine the three advantages of KDE-Neuron.

6.2 Limitations of data-driven reconstruction algorithms

KDE-Neuron is a purely data-driven descriptive algorithm. It extracts all the information for generating a VN from KDEs generated from real data. The algorithm is descriptive in the sense that it reconstructs the final morphologies and disregards growth processes or interactions with the environment. As such, we may use our findings to make statements about the opportunities and limitations of data-driven algorithms for neuron reconstruction. The opportunities

were already outlined in the previous subsection. Despite these important opportunities created by using a data-driven algorithm, there are three limitations of descriptive and data-driven algorithms, namely (1) the lack of environmental interactions, (2) the lack of explicit knowledge about neuronal morphologies and (3) the availability and quality of the morphological data corpus.

The first two limitations are interwoven and we discuss them together. Two observations from our results illustrate the first two limitations. The first observation is that KDE-NEURON failed to reproduce the branching angles in the granule cells. The second observation is that the post-selection procedure filters neurons with particular combinations of morphological properties, and, as a result, reshapes the distributions of the generated data for some properties. Figure 13 provides an illustration of such a reshaping. In Figure 13 (A) we plotted the histogram and KDE model constructed for the *B_elev* property of the prototype data. In Figure 13 (B) we show similar plots constructed with generated data. It is clear that the right KDE is a reshaped version of the original KDE and hence indicates a mismatch between prototype neurons and generated VNs. The reshaping is caused by the post-selection filter. Due to the filtering, the distribution of branching angles obtained from the prototype data does not match with the distribution obtained from the generated data. So, filtering affects the generated distribution of branching angles yielding a bad match with the prototype distribution, whereas filtering is necessary for obtaining plausible VNs.

This paradox can be explained in two ways. The first explanation is that the statistical model was unable to reproduce the underlying characteristics in the data. An example of such a characteristic is that granule cells tend to grow into a single direction regardless their starting angle. This characteristic leads to the observation that if the stem angle is far from the desired growing direction, the branching angle at the first bifurcation will 'correct' the direction of the dendrite. Thus, the first branching angle seems to complement or correct the stem angle. This type of characteristic is generally attributed to chemical gradients in the environment surrounding growing neurons (Feng et al., 2005; Samsonovich and Ascoli, 2005a; Luczak, 2006) and is neglected by a descriptive model. A related way of explaining the paradox is that all neurons filtered by the post-selection filter are plausible to the extent that the tested properties are within biologically observed ranges. However, combinations of these properties are not tested. Consequentially, a generated neuron can have 'impossible' combinations of properties.

Underlying both explanations is the fundamental limitation of descriptive models such as KDE-Neuron that the influence of the environment is not taken into account, and, potential intra-neuronal relations between morphological properties are discarded. There is an abundance of literature on the neuronal growth in an environment and how cell growth (Scott and Luo, 2001) is supported from within the cell i.e., the principles underlying intra-neuronal relations (Horch and Katz, 2002). However, there appears to be a lack of *quantitative descriptions* of both interactions with the environment, and the intra-neuronal relations. When these effects are modelled, the models rather try to indicate that there is indeed an influence of the environment on growth. For instance, in Samsonovich and Ascoli (2003, 2005b) it is shown that with a straightforward mechanism of somatocentric tropism pyramidal neurons can take their shape. Moreover, in Luczak (2006) it was shown that the neuronal morpholo-

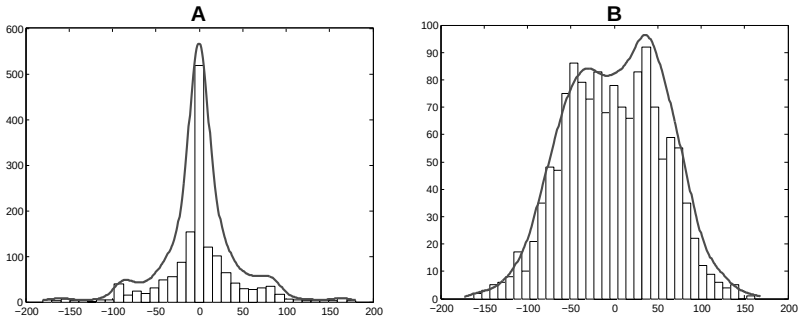

Figure 13: Distribution of the B_elev for the prototype data (A) and generated data (B). In both figures the KDE model of the data is illustrated by the solid curve and multiplied by 30000 and 15000 to match scale.

gies of different cells can be 'aggregated' (instead of reconstructed) solely by expressing environmental cues. With respect to the intra-neuronal relations, in Nowakowski et al. (1992) it was shown that branching angles can be determined by competition between different branches of the same neuron. Without any quantitative theory, no such information can be used by a data-driven method.

The last limitation concerns the size and the quality of the available morphological data corpus. We discuss the size and quality separately. The size of the data has a direct effect on the quality of the statistical models and therefore directly affects the generated VNs. Without a sufficient number of prototypes and derived data samples, it is hard to induce a statistical model that includes all the peculiarities present in the prototype data. The necessity of sufficiently large data sets is generally recognized in the field of machine learning (Alpaydin, 2004). The quality of the data sets also affects the quality of the induced models. Because of a lack of standards and conventions in tracing neurons (Kaspirzhny et al., 2002; Scorcioni et al., 2004; Szilágyi and De Schutter, 2004; Ambros-Ingerson and Holmes, 2005), traced neurons originating from different researchers and labs might be incompatible (Burns, 2001). Next to the incompatibility between different traced neurons, tracing individual neurons remains hard. For instance, the diameter of a dendritic segment is hard to determine and different researchers have different reasons to choose the diameter that they assign. As a result, prototype neurons that are 'digitized' by different researchers from different labs show wide variation (Kaspirzhny et al., 2002; Steuber et al., 2004). Non-parametric models are known to be highly efficient in modelling peculiarities in data, but it is still impossible to determine whether a particular peculiarity is just noise or signal. Because a data-driven model depends on the available data, this imposes a limitation on any data-driven algorithm.

6.3 Future enhancements

Future enhancements of KDE-Neuron address the aforementioned limitations. For instance, studies from Feng et al. (2005); Samsonovich and Ascoli (2005b); Luczak (2006) show that it is possible to include environmental influences into a reconstruction algorithm. The main issue that remains is the definition of

quantitative theories, and once available, their integration into algorithms like KDE-Neuron. For future research it would be interesting to start investigating how to integrate environmental influences while still maintaining full control over all morphological properties. At the same time, the quality and availability of morphology data should be improved.

There is one particular line of research that can enhance our algorithm in the future, namely establishing a truly conditional sampling procedure. In KDE-Neuron, the values for the basic properties are sampled independently and only the inference about termination, bifurcation, and prolongation is conditional. However, as outlined before, relations between morphological properties should be taken into account. Two enhancements are proposed. First, conditional sampling of all values for basic properties. Second and more substantial, sampling conditioned not only on morphological properties but also on previously taken samples. In case of the granule cell, this type of conditional sampling might solve the limitation caused by environmental cues: the algorithm knows the stem angle of a branch and can sample a good subsequent angle so that the dendrite grows in the correct direction. A good starting point to investigate the truly conditional samples might be by employing Bayesian networks (Neapolitan, 2003) or Gaussian filters or particle filters (Thrun et al., 2005).

7 Conclusions

We conclude that the KDE-Neuron algorithm is capable of generating virtual neurons that are are representative for the prototype neurons. Provided that a sufficiently large prototype set can be compiled, KDE-Neuron offers biologically realistic virtual neurons that can be used for the analysis and modelling of neurons and neural networks. Future work addresses the validation of KDE-Neuron on a wider variety of neuron types and further optimizing the efficiencies and performances of its constituent algorithms.

Information sharing statement

The prototype software can be downloaded from BTN's website, http://www.cs.unimaas.nl/btorbennielsen/evol-neuron/. The data acquisition software is written in Java (free download from http://java.sun.com). The generation algorithm itself (including the statistical model construction) is written in Matlab (MathWorks, Inc.), which requires a license for its use. A manual is supplied on the same website to outline how to perform the different steps, i.e., data acquisition, generation of VNs and post-filtering, and analysis of the results.

Acknowledgements

BTN was funded by the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The authors thank Drs. Jaap van Pelt and Klaus Stiefel, and Steven de Jong for fruitful discussion on this topic. We also wish to thank the three anonymous reviewers for their suggestions that improved the manuscript.

References

Alpaydin, E. (2004). *Introduction to machine learning*. MIT Press, Cambridge, MA.

Ambros-Ingerson, J. and Holmes, W. R. (2005). Analysis and comparison of morphological reconstructions of hippocampal field CA1 pyramidal cells. *Hippocampus*, 15:302–315.

Ascoli, G. A. (1999). Progress and perspectives in computational neuroanatomy. *Anatomical Record*, 257:195–207.

Ascoli, G. A. (2006). Mobilizing the base of neuroscience data: the case of neuronal morphologies. *Nature Neuroscience Reviews*, 318(7):318–324.

Ascoli, G. A. (2007). Success and rewards in sharing digital reconstructions of neuronal morphology. *Neuroinformatics*, 5:154–160.

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

Ascoli, G. A., Krichmar, J. L., Scorcioni, R., Nasuto, S. J., and Senft, S. L. (2001). Computer generation and quantitative morphometric analysis of virtual neurons. *Anatomy and Embryology*, 204:283–301.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer-Verlag, New York, NY.

Burke, R., Marks, W., and Ulfhake, B. (1992). A parsimonious description of motorneuron dendritic morphology using computer simulation. *Journal of Neuroscience*, 12(6):2403–2416.

Burns, G. (2001). Knowledge management of the neuroscientific literature: the data model of the neuroscholar system. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 356:1187–1208.

Cameron, W., He, F., Kalipatnapu, P., Jodkowski, J., and Guthrie, R. (1991). Morphometric analysis of phrenic motoneurons in the cat during postnatal development. *Journal of Comparative Neurology*, 314(4):763–776.

Cannon, R., Turner, D., Pyapali, G., and Wheal, H. (1998). An on-line archive of reconstructed hippocampal neurons. *Journal of Neuroscience Methods*, 84(1-2):49–54.

Carnevale, N. and Hines, M. (2006). *The* NEURON *book*. Cambridge University Press, Cambridge, UK.

da F. Costa, L., Barbosa, M., and Coupez, V. (2005). On the potential of the excluded volume and autocorrelation as neuromorphometric descriptors. *Physica A: Statistical Mechanics and its Applications*, 348:317–326.

De Schutter, E. and Bower, J. M. (1994). An active membrane model of the cerebellar purkinje cell I. simulation of current clamps in slice. *Journal of Neurophysiology*, 71(1):375–400.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, NY.

Donohue, D. E. and Ascoli, G. A. (2005). Local diameter fully constraints dendritic size in basal but not apical trees of ca1 pyramidal neurons. *Journal of Computational Neuroscience*, 19:223–238.

Eberhard, J., Wanner, A., and Wittum, G. (2007). NeuGen: a tool for the generation of realistic morphology of cortical neurons and neural networks in 3D. *Neurocomputing*, 70(1-3):327–342.

Feng, N., Ning, G., and Zheng, X. (2005). A framework for simulating axon guidance. *Neurocomputing*, 68:70–84.

Fernández, E. and Jelinek, H. F. (2001). Use of fractal theory in neuroscience: methods, advantages and potential problems. *Methods*, 24:309–321.

Glaser, J. and Glaser, E. (1990). Neuron imaging with Neurolucida - a PC-based system for image combining microscopy. *Computerized Medical Imaging and Graphics*, 14:307–317.

Hillman, D. (1979). *Neuronal shape parameters and substructures as a basis of neuronal form*, volume The neurosciences, Fourth Study Program, chapter 27. The MIT Press, Cambridge, MA.

Horch, H. W. and Katz, L. C. (2002). Bdnf release from single cells elicits local dendritic growth in nearby neurons. *Nat Neurosci*, 5(11):1177–1184.

Jones, C., Marron, J., and Sheather, S. (1996). A brief survey of bandwidth selection for density estimation. *Journal of American Statistical Association*, 91(433):401–407.

Kaspirzhny, A. V., Gogan, P., Horcholle-Bossavit, G., and Tyc-Dumont, S. (2002). Neuronal morphology data bases: morphological noise and assessment of data quality. *Network: Computation in Neural Systems*, 13:357–380.

Lehmann, E. and Romano, J. (2006). *Testing statistical hypotheses (3rd edition)*. Springer-Verlag, Berlin Heidelberg.

Lien, J.-M., Morales, M., and Amato, N. M. (2003). Neuron PRM: A Framework for Constructing Cortical Networks. *Neurocomputing*, 52–54:191–197.

Lindsay, K. A., Maxwell, D. J., Rosenberg, J. R., and Tucker, G. (2007). A new approach to reconstruction models of dendritic branching patterns. *Mathematical Biosciences*, 205(2):271–296.

Loader, C. (1999). Bandwidth selection: Classical or plug-in? *The Annals of Statistics*, 27(2):451–438.

Luczak, A. (2006). Spatial embedding of neuron trees modeled by diffusive growth. *Journal of neuroscience methods*, 157:132–141.

Markram, H. (2006). The Blue Brain Project. *Nature Reviews Neuroscience*, 7:153–160.

Myatt, D., Hadlington, T., Ascoli, G., and Nasuto, S. (2007). Inter-user variability of semi-manually reconstructed dendritic trees with the freeware tool neuromantic. *Journal of Microscopy*, submitted.

Neapolitan, R. E. (2003). *Learning bayesian networks*. Prentice Hall.

Nowakowski, R. S., Hayes, N. S., and Egger, M. D. (1992). Competitive interactions during dendritic growth: a simple stochastic growth algorithm. *Brain research*, 576:152–156.

Parzen, E. (1962). On estimation of probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076.

Raykar, V. and Duraiswami, R. (2006). Very fast optimal bandwidth selection for univariate kernel density estimation. Technical Report CS-TR-4774, University of Maryland, College Park.

Rihn, L. and Claiborne, B. (1990). Dendritic growth and regression in rat dentate granule cells during late postnatal development. *Brain Research. Developmental Brain Research*, 54(1):115–124.

Robert, C. (2007). *The Bayesian Choice: from decision-theoretic foundations to computational Implementation*. Springer-Verlag, New York, NY.

Samsonovich, A. V. and Ascoli, G. A. (2003). Statistical morphological analysis of hippocampal principal neurons indicates cell-specific repulsion of dendrites from their own cell. *Journal of Neuroscience Research*, 71:173–187.

Samsonovich, A. V. and Ascoli, G. A. (2005a). Algortihmic description of hippocampal granule cell dendritic morphology. *Neurocomputing*, 65-66:253–260.

Samsonovich, A. V. and Ascoli, G. A. (2005b). Statistical determinants of dendritic morphology in hippocampal pyramidal neurons: a hidden markov model. *Hippocampus*, 15:166–183.

Scorcioni, R., Lazarewicz, M. T., and Ascoli, G. A. (2004). Quantitative morphometry of hippocampal pyramidal cells: Differences between anatomical classes and reconstructing laboratories. *Journal Comparative Neurology*, 473:177–193.

Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley and Sons Inc., New York, NY.

Scott, E. and Luo, L. (2001). How do dendrites take their shape? *Nature (neuroscience)*, 4:(4)359–365.

Silverman, B. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London, UK.

Steuber, V., De Schutter, E., and Jaeger, D. (2004). Passive models of neurons in the deep cerebellar nuclei: the effect of reconstruction errors. *Neurocomputing*, 58–60:563–568.

Szilágyi, T. and De Schutter, E. (2004). Effects of variability in anatomical reconstruction techniques on models of synaptic integration by dendrites: a comparison of three internet archives. *European Journal of Neuroscience*, 19:1257–1266.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. Oxford University Press, Cambridge, Mass.

Torben-Nielsen, B., Tuyls, K., and Postma, E. O. (2008). Evol-neuron: virtual neuron generation. *Neurocomputing*, 71(4-6):963–972.

van Pelt, J. and Schierwagen, A. (2004). Morphological analysis and modeling of neuronal dendrites. *Mathematical Biosciences*, 188:147–155.

Wand, M. and Jones, C. (1995). *Kernel smoothing*. Chapman and Hall, London, UK.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques (2nd edition)*. Morgan Kaufmann, San Francisco, CA.

A Kernel density estimates

In the appendix we give more details on the implementation of kernel density estimation for univariate and bivariate prototype data. We also explain how the automatic selection of the bandwidth parameter has been performed which is an important issue that we did not address so far.

A.1 Univariate KDEs

An intuitive, non-parametric way to model observed data is by means of a histogram estimator, which is formally defined by:

$$\hat{f}(x) = \frac{|\{x_i \in N(x),\ i = 1, \ldots, n\}|}{nh} \ , \tag{5}$$

where $N(x)$ is the bin centred at x and of width h. A KDE is a smooth version of the histogram estimator were a kernel function K is used that counts observations close to x with weights that are decreasing with the distance from x (Parzen, 1962):

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \ . \tag{6}$$

A popular choice of kernel function is the Gaussian distribution with zero mean and unit variance. The KDE with Gaussian kernel function becomes:

$$\hat{f}(x) = \frac{1}{nh\sqrt{2\pi}} \sum_{i=1}^{n} \exp\left[-\frac{(x - x_i)^2}{2h^2}\right] \ , \tag{7}$$

with h the bandwidth (standard deviation) of the Gaussian components. The kernel density estimator is a legitimate probability density function when the kernel function is non-negative everywhere and integrates to one. It is easily verified that this holds for the Gaussian. The parameter h acts as the smoothing parameter.

A.2 Multivariate KDEs

Suppose we have a sample consisting of n bivariate observations of the form $(x_1, y_1), \ldots, (x_n, y_n)$ that are independently drawn from an unknown probability distribution f over the random variable (X, Y). The KDE for the joint distribution is then:

$$\hat{f}(x, y) = \frac{1}{nh_x h_y} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_x}\right) K\left(\frac{y - y_i}{h_y}\right) \quad , \tag{8}$$

with h_x and h_y the bandwidths for the random variables X and Y, respectively. Since we are using the Gaussian as kernel function the estimator for the joint distribution becomes:

$$\hat{f}(x, y) = \frac{1}{nh_x h_y \sqrt{(2\pi)^2}} \sum_{i=1}^{n} \exp\left[-\frac{(x - x_i)^2}{2h_x^2} - \frac{(y - y_i)^2}{2h_y^2}\right] \quad . \tag{9}$$

In the rest of this appendix we write equations explicitly for the Gaussian kernel although they can be written in terms of any other kernel function.

The kernel density estimate of a marginal distribution can be calculated from an application of the sum rule of probability. For example, the estimator for the marginal distribution $f_X(x)$ of X (assuming Y to be a real-valued random variable) is:

$$\hat{f}_X(x) = \int_{-\infty}^{+\infty} \hat{f}(x, y) \mathrm{d}y = \frac{1}{nh_x \sqrt{2\pi}} \sum_{i=1}^{n} \exp\left[-\frac{(x - x_i)^2}{2h_x^2}\right] \quad , \tag{10}$$

since the Gaussian integrates to one over the real line. The kernel density estimator for $f_Y(y)$ can be defined analogously. A kernel density estimator for a conditional density, e.g., the estimator of the density function of X given that Y is some value y, is computed by a straightforward application of the product rule of probability:

$$\hat{f}(x|y) = \frac{\hat{f}(x, y)}{\hat{f}_Y(y)} = \frac{\frac{1}{h_x \sqrt{2\pi}} \sum_{i=1}^{n} \exp\left[-\frac{(x - x_i)^2}{2h_x^2} - \frac{(y - y_i)^2}{2h_y^2}\right]}{\sum_{i=1}^{n} \exp\left[-\frac{(y - y_i)^2}{2h_y^2}\right]} \quad . \tag{11}$$

The above equations for density estimation with respect to bivariate observations can be generalized in a straightforward manner to observations with more than two dimensions.

A.3 Automatic bandwidth selection

The bandwidth is the single parameter for kernel density estimation using Gaussian kernel. It acts as a smoothing factor and consequently determines the quality of the estimator. The best bandwidth has to be learned from the data since the data, besides prior knowledge if available, are the only objective basis to decide when the estimator is oversmoothing or undersmoothing. Various methods have been proposed for the task of *automatic bandwidth selection*. A brief survey is found in (Wand and Jones, 1995; Jones et al., 1996).

It is natural to define the best bandwidth as the one resulting in the best KDE. The quality of a KDE $\hat{f}(x)$ with respect to the true density $f(x)$ is often measured by the integrated squared error:

$$\mathrm{ISE}(\hat{f}, f) = \int_{-\infty}^{+\infty} (\hat{f}(x) - f(x))^2 dx \ . \tag{12}$$

The ISE depends on a particular sample of n observations since the sample is used as information for the KDE. The mean integrated squared error averages the integrated squared error over all samples:

$$\mathrm{MISE}(\hat{f}, f) = \int_{-\infty}^{+\infty} \mathrm{E}\left[(\hat{f}(x) - f(x))^2\right] dx \ , \tag{13}$$

where we have used $\mathrm{E}[\cdot]$ to denote expected value. Clearly, the bandwidth with minimum MISE gives us an estimator that on average has lowest ISE for a particular sample of n observations. Given that n is sufficiently large, an approximation to MISE can be shown to be (Scott, 1992):

$$\mathrm{AMISE}(\hat{f}, f) = \frac{R(K)}{nh} + \frac{h^4 \mu_2(K)^2 R(f'')}{4} \ , \tag{14}$$

where K is the kernel and the following definitions are used:

$$R(g) = \int_{-\infty}^{+\infty} g(x)^2 dx, \text{ and } \mu_2(g) = \int_{-\infty}^{+\infty} x^2 g(x) dx \ . \tag{15}$$

Setting the derivative of Eq. 14 with respect to the bandwidth to zero gives the optimal solution for the bandwidth:

$$h = \left[\frac{R(K)}{\mu_2(K)^2 R(f'') n}\right]^{1/5} \ . \tag{16}$$

Unfortunately this solution depends on the second order derivative of the true density $f(x)$ which is unknown. Moreover, estimating the derivative is a more difficult problem than estimating the density itself (Raykar and Duraiswami, 2006).

There are several approaches to approximate the best bandwidth in terms of AMISE. The basic approach is called *rules of thumb* and assumes that the data is generated by a Gaussian distribution (Wand and Jones, 1995). Given data of dimensionality d, the optimal bandwidth for the i-th variable is then given by:

$$h_i = \left(\frac{4}{d+2}\right)^{1/(d+4)} n^{-1/(d+4)} \hat{\sigma}_i \ , \tag{17}$$

with $\hat{\sigma}_i$ the estimator for the standard deviation of the i-th variable. For univariate data ($d = 1$), a more specific estimate of the bandwidth is:

$$h_i = 0.9 \, S \, n^{-1/5} \ , \tag{18}$$

with S the minimum of the standard deviation and three-quarters of its interquartile range (Silverman, 1986). In our experiments we found that the rules of thumb bandwidths for multivariate data are often better than the bandwidths found by some more advanced approaches, even though the assumption of Gaussian distributions is not satisfied. This is in line with the findings reported by Loader (1999).