Hashing Forests for Morphological Search and Retrieval in Neuroscientific Image Databases

Sepideh Mesbah[1,2,*], Sailesh Conjeti[1,*], Ajayrama Kumaraswamy[3],
Philipp Rautenberg[2,3], Nassir Navab[1,4], and Amin Katouzian[1]

[1] Computer Aided Medical Procedures, Technische Universität München, Germany
[2] Max Plank Digital Library, München, Germany
[3] Computational Neuroscience, Ludwigs Maximillian Universität München, Germany
[4] Computer Aided Medical Procedures, Johns Hopkins University, USA

Abstract. In this paper, for the first time, we propose a data-driven search and retrieval (hashing) technique for large neuron image databases. The presented method is established upon hashing forests, where multiple unsupervised random trees are used to encode neurons by parsing the neuromorphological feature space into balanced subspaces. We introduce an inverse coding formulation for retrieval of relevant neurons to effectively mitigate the need for pairwise comparisons across the database. Experimental validations show the superiority of our proposed technique over the state-of-the art methods, in terms of precision-recall trade off for a particular code size. This demonstrates the potential of this approach for effective morphology preserving encoding and retrieval in large neuron databases.

1 Introduction

Neuroscientists often analyze the 3D neuromorphology of neurons to understand neuronal network connectivity and how neural information is processed for evaluating brain functionality [1, 2]. Of late, there is a deluge of publicly available neuroscientific databases (especially 3D digital reconstructions of neurons), which consist of heterogeneous multi-institutional neuron collection, acquired from different species, brain regions, and experimental settings [3, 4]. Finding relevant neurons within such databases is important for comparative morphological analyses which are used to study age related changes and the relationship between structure and function [1].

Recently, the concept of the neuromorphological space has been introduced where each neuron is represented by a set of morphological and topological measures [1]. Authors in [1] used this feature space for multidimensional analysis of neuronal shape and showed that the cells of the same brain regions, types, or species tend to cluster together in such a space. This motivated us to leverage this space for evaluating inter-neuron similarity and propose a data-driven retrieval (hashing) system for large neuron databases. Recently, a neuron search

[*] S. Mesbah and S. Conjeti contributed equally towards the work.

N. Navab et al. (Eds.): MICCAI 2015, Part II, LNCS 9350, pp. 135–143, 2015.
DOI: 10.1007/978-3-319-24571-3_17

algorithm was proposed by [2], where pairwise $3D$ structural alignment was employed to find similar neurons. In another approach, authors in [5] focused on the evaluation of morphological similarities and dissimilarities between groups of neurons deploying clustering technique using expert-labeled metadata (like species, brain region, cell type, and archive). These existing neuronal search and retrieval systems are either too broad (*lacking specificity in search*) or too restrictive (*dependent on exact expert-defined values/meta information*) and thus not suitable for reference-based hashing in large scale neuron databases.

Several efficient encoding and searching approaches have been proposed for large scale retrieval in machine learning and medical image computing, including **(1)** generalized hashing methods like data independent methods (*e.g.* Locality Sensitive Hashing (LSH) [6]), data-driven methods (*e.g.* Spectral Hashing (SH), and Self Taught Hashing (STH) [7, 8]); and **(2)** application-specific methods including composite hashing for histopathological image search [9] and on finding similar coronary angiograms in $2D$ X-ray database [10].

The LSH generates binary encoding by partitioning feature space with randomly generated hyperplanes [6]. Unlike LSH, instead of working on the original feature space, the SH generates hash codes using low dimensional representation obtained using Laplacian Eigenmaps (LEM) [7]. Later, Zhang *et al.* [8] introduced self-learnt hashing functions by median-thresholding the low dimensional embedding from LEM and training a support vector machine (SVM) classifier as the hash function for encoding new input data. Data independent methods like LSH require large code words to efficiently parse the feature space as they rely on their asymptotic nature for convergence [11]. For similar code lengths, SH and STH can provide better retrieval as the hashing function is based on the data distribution. However, these data driven methods are mainly challenged by their restricted scalability in code size and lack of independence of the hashing functions. It has also been observed that increasing code length in such data driven methods may not necessarily improve performance, depending on the data characteristics. Redressing these issues are crucial for fast growing heterogeneous database like neuroscientific databases, where both scalability and sensitivity are important.

In this paper, we propose a data-driven hashing scheme for large neuron databases called hashing forests (HF). These are established upon unsupervised random forests trained to parse the neuromorphological space in a hierarchical fashion. We demonstrate that HF can generate more sensitive code words than LSH, SH or STH by effectively utilizing its tree-structure and ensemble nature. Trees in HF are trained independently and they are more easily scalable to larger code lengths than SH and STH. In comparison to random forest hashing method proposed in [11], we introduce an inverse coding scheme which effectively mitigates database-wide pairwise comparisons, which is better suited for fast large-scale hashing. We formulate tree-traversal path based coding scheme for more efficient hierarchical parsing of the neuromorphological space. Further, to the best of our knowledge, this is the first work that focuses in entirety on hashing in large neuroscientific databases.

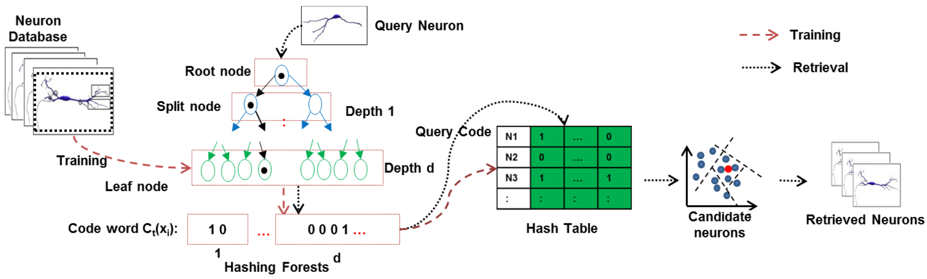


Fig. 1. Schematic of proposed search and retrieval scheme using Hashing Forests.

2 Methodology

Ideally, the hashing method should generate compact, similarity-preserving and easy to compute representations (called *code words*), which can be used for accurate search and fast retrieval [7]. In the context of neuroscientific databases, the desired similarity preserving aspect of the hashing function implies that *morphologically similar neurons are encoded with similar binary codes.* In the neuromorphological space, the normalized Euclidean distance defined between neurons correlates well with the inter-neuron morphological similarity. However, it is not desirable for hashing in large neuron databases owing to high memory expenditure in addition to increased computational time complexity. Consider the neuromorphological feature vector of a neuron n_i, say $\mathbf{x}_i := \{\mathbf{x}_1, .., \mathbf{x}_k, .., \mathbf{x}_n\} \in \mathbb{R}^N$, were N is the space dimensionality. A detailed description of the feature extraction and biological significance is presented in [12]. We model hashing functions through *unsupervised* randomized forest (called hashing forests $\mathcal{H}$) which parse and encode the neuromorphological subspaces in similarity-preserving binary codes. Fig. 1 illustrates the schematic of the overall hashing process, which is discussed in following section.

Training Phase: Hashing forest ($\mathcal{H}$) is an ensemble of binary decision trees which partitions the feature space hierarchically based on learnt binary split functions. A forest of T binary trees with maximum depth of d, requires $(T \times d)$ bits to encode each neuron. Each tree is grown in an *unsupervised* fashion by recursively partitioning the selected feature sub-space (*say into left and right with s_l and s_r samples respectively*) at each node. This is based on a univariate split function (*say* ϕ) which optimizes tree balance by minimizing $\mathcal{E}(\phi)$ $(\mathcal{E}(\phi) = max\{(s_l/s_r) - 1, (s_r/s_l) - 1\})$ [13] . The splitting continues until the maximum defined tree-depth d is reached. The time complexities to grow a tree and ensemble a forest are tabulated in Table 1 (S1-S2).

Hash Table Generation: Given a trained tree (t) of the hashing forest $\mathcal{H}$, each neuron n_i (*characterized by* $\mathbf{x}_i$) in the database is passed through it till it reaches the leaf node. We propose to use the neuron's traversal path to generate the tree-specific code word $C_t(\mathbf{x}_i)$, instead of just the leaf indices. For a tree t of depth d, the split and the leaf nodes are assigned breadth-first order indices

Table 1. Time Complexity Analysis

S1	**Building tree**	$O(\sqrt{N} * M * d)$ [13]	Training
S2	**Building forests**	$O(T * \sqrt{N} * M * d)$	
S3	**Generating hash table**	$O(T * d) + O(M * S)$	
S4	**Generating Single Query Code**	$O(T * d)$	Retrieval
S5	**Retrieving with Forward Code**	$O(M * S)$	
S6	**Retrieving with Inverse Code**	$O(T * d) + O(T * (d - 1))$	
S7	**Quick sort**	$O(M \log M)$	

Symbols: Code word Size $S = T(2^{d+1} - 2)$); Number of Trees T; Number of Features N; Tree Depth d; Retrieval Database Size M.

(say k_t) which are associated with binary bit b_{k_t} in the code word $C_t(\mathbf{x}_i)$. For a particular neuron, if node k_t is part of its path, then b_{k_t} is set to 1, otherwise its set to 0. This leads to a $(2^{d+1} - 2)$ bit sparse code word $C_t(\mathbf{x}_i)$ with (d) ones. It must however be noted that only d bits are required to generate $C_t(\mathbf{x}_i)$ as there are only 2^d possible traversal paths, each leading to a unique leaf node. We repeat the same process for every other tree in the forest to generate the sparse code block $C_{\mathcal{H}}(\mathbf{x}_i)$ of size $(T * (2^{d+1} - 2))$ for each neuron. The time complexity of this step is shown in Table 1 (S3). For faster retrieval, we pre-compute the code blocks for all M neurons in retrieval/training database $\mathcal{D}$ and generate a hash table of size $M \times (T * (2^{d+1} - 2))$. This is stored using $(M * T * d)$ bits along with traversal paths saved in a $(2^d * (2^{d+1} - 2))$ binary look-up table.

Inverse Coding: Each bit b_{k_t} in $C_{\mathcal{H}}$ encodes a unique neuromorphological subspace, which is constrained by the split functions of t leading to node k_t. In order to avoid pair-wise comparisons between the neurons during retrieval in large databases, we formulate an inverse coding scheme. We transpose the hash table to generate the inverted hash table $\mathcal{I}$ which is a sparse $((T*(2^{d+1}-2)) \times M)$ dimensional matrix. This implies that for feature vector $\mathbf{x}_i$, if bit b_{k_t} in $C_t(\mathbf{x}_i)$ is 1, then $\mathcal{I}(k_t, i) = 1$, and it belongs to the feature subspace encoded by b_{k_t}.

Testing Phase: For a given *query* neuron n_q (with feature vector $\mathbf{x}_q$), the corresponding code block $C_{\mathcal{H}}(\mathbf{x}_q)$ is generated in a similar fashion to the Hash Table Generation phase. In the direct retrieval formulation, pairwise comparisons (*through hamming distance*) between $C_{\mathcal{H}}(\mathbf{x}_q)$ and code blocks of neurons (say $C_{\mathcal{H}}(\mathbf{x}_i)$ for neuron n_i) in the retrieval database $\mathcal{D}$ are made to evaluate inter-neuron similarity $\mathcal{S}(n_q, n_i)$ *i.e.*

$$\mathcal{S}(n_q, n_i) = \frac{1}{(T * (2^{d+1} - 2))} \sum_{\forall bits} (C_{\mathcal{H}}(\mathbf{x}_q) == C_{\mathcal{H}}(\mathbf{x}_i)) \qquad (1)$$

If n_q generates the same code block as a neuron in $\mathcal{D}$ (*i.e. both belong to the same neuromorphological subspace*), we assign perfect similarity to them ($S = 1$). However, the pairwise comparison for large scale databases is computationally expensive as seen from its time complexity in Table 1 (S5). To mitigate this, in the inverse coding, we formulate the similarity function as $T * (d-1)$ dimensional similarity accumulator cell $\mathcal{A}_{n_q}$. Given the code block $C_{\mathcal{H}}(\mathbf{x}_q)$ for the query neuron n_q, $\mathcal{A}_{n_q}$ is calculated as:

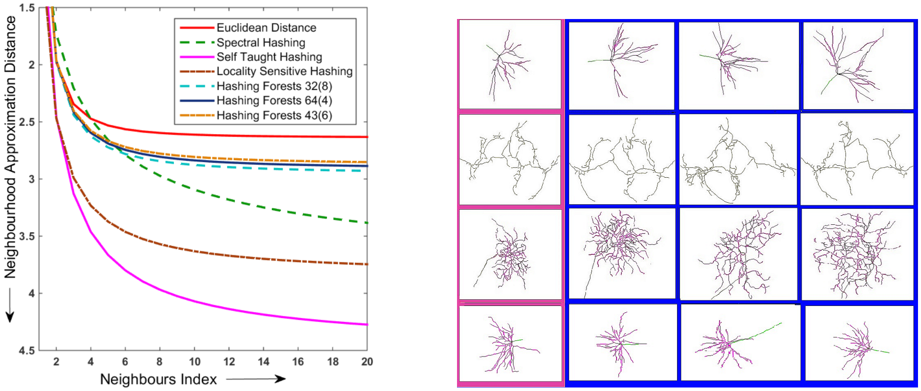
$$\mathcal{A}_{n_q}(i) = \frac{1}{T * (d-1)} \sum_{\forall k_t} \mathcal{I}(k_t, i) \text{ if bit } b_{k_t} \text{ in } C_{\mathcal{H}}(\mathbf{x}_q) = 1 \tag{2}$$

The inter-neuron similarity $\mathcal{S}(n_q, n_i)$ is related to $\mathcal{A}_{n_q}$ as $\mathcal{S}(n_q, n_i) = \mathcal{A}_{n_q}(i)$. Such an inverse formulation is computationally more efficient for large databases(as seen from the order complexity in Table 1 (S6)), than the forward scheme (Table 1 (S5)). In the task of retrieving an ordered set of K most morphologically similar neurons from $\mathcal{D}$, we dynamically generate a set of candidate neurons (say size of $2K$) from $\mathcal{D}$ by using quick sort on the hashing forest similarity (Table 1 (S7)). We further sort them by using normalized Euclidean distance between the neurons in this set and the *query* neuron for more morphologically consistent ranking. The additional computational expense due to this refinement step is low as $K << M$. (For validation purposes, this refinement using normalized Euclidean distance is performed on all the comparative methods considered in Section 3)

3 Experiments and Observations

Database: We used 18106 publicly available 3D reconstructions of neurons (extracted from [3, 15–23]). We employed Lmeasure toolbox [12] to extract 3D neuromorphological features, which characterize different aspects of neuron structure including whole neuron morphology, bifurcation features, and neuron compartment level features [1]. Successful morphology-preserving hashing in neuroscientific databases depends on the efficacy of the code word to compactly represent this neuromorphological space as well as efficiently compute interneuron similarity using the code word.

Experiment 1: *Effective Neighbourhood Approximation with fixed codesize* We introduce the *Neighbourhood Approximation* (NA) graph which models how close the neighbourhood estimated using code words (*from the hashing method*), approximates the true neighbourhood around a neuron in the neuromorphological space. For a particular hashing method, NA for the j^{th} neighbour is defined as the average of the normalized Euclidean distance between the neurons and retrieved j^{th} neighbour for all neurons in $\mathcal{D}$. Let, for neuron n_i (with feature vector $\mathbf{x}_i^0$), the j^{th} neighbour have a feature vector $\mathbf{x}_i^j$, then $\text{NA}(j) = \frac{1}{M}\left(\sum_{i=1}^{M} \varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j)\right)$ where $\varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j) = \left(\sqrt{\frac{1}{N}\sum_{a=1}^{N} \frac{\left(x_{ia}^0 - x_{ia}^j\right)^2}{s_a}}\right)$. $\varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j)$ is the normalized Euclidean distance between $\mathbf{x}_i^0$ and its j^{th} neighbour $\mathbf{x}_i^j$ with a^{th} feature standard deviation s_a estimated over the whole database which is chosen for invariance to scales of different features. We use the NA graph calculated using neighbourhood based on normalized Euclidean distance (NA_{EUC}) as the benchmark and the hashing method which performs closest to it is considered as



(a) NA Graph

(b) Four examples of retrieved neurons.

Fig. 2. Evaluation of the proposed retrieval system. **Fig. 2a:** Neighborhood Approximation (NA) graph for the comparative methods and configurations of hashing forests for fixed codeword size (256 bits). **Fig. 2b:** For each query neuron on the left (pink), the three best matches are shown on the right (blue). (*High resolution images are provided in the supplementary material.*)

to have the best neighbourhood approximation in comparison. NA graph is evaluated over the target database and the results for all the comparative methods are reported in Fig. 2a. For fair validation, we keep the size of the code-block fixed at 256 bits for this experiment. Further, to examine the sensitivity of HF parameters (number of trees (T) vs. depth (d)) towards neighbourhood approximation, we tested three different configurations varying tree depths: $T = 32$ trees with $d = 8$ ($32 * 8 = 256$ bits); $T = 64$ trees with $d = 4$ ($64 * 4 = 256$ bits); and $T = 43$ trees with $d = 6$ ($43 * 6 = 258$ bits(~ 256 bits))

Observations on Experiment 1: Fig. 2a demonstrates that for a fixed code size, HF approximates neighbourhood better than the comparative LSH, SH and STH methods. Further, analysis of the sensitivity of HF parameters (T *vs.* d) towards NA, demonstrated that HF with 43 trees and depth 6 performs better than other shallower and deeper tree configurations. Thus, we infer that there exists an optimal depth for a given codesize, where HF optimally parses the neuromorphological space. Fig. 2b demonstrates the performance for 4 distinct neurons with the closest neighbours retrieved using HF and shows close morphological similarity amongst the neurons and the neighbours.

Experiment 2: *Hashing retrieval precision vs. Code block size.* We evaluate the retrieval performance by computing the $F1_{score}$ as follows:

$$F1_{score} = \sqrt{Precision \times Recall} = \sqrt{\frac{|\mathcal{N}_{\varepsilon}(n_i) \bigcap \mathcal{N}_{\mathcal{H}}(n_i)|^2}{|\mathcal{N}_{\mathcal{H}}(n_i)| \times |\mathcal{N}_{\varepsilon}(n_i)|}}.$$

This measure is often used in information retrieval algorithms to better understand the tradeoff between precision and recall. The $\mathcal{N}_{\varepsilon}(n_i)$ represents the set

Table 2. Experiment 2: $F1_{score}$ (in %) vs. Code Size (in *bytes*)

Code Size	**Case 10**				**Case 25**			
(*in bytes*)	LSH	SH	STH	HF	LSH	SH	STH	HF
8	6.0	14.0	**16.2**	7.2	5.06	13.03	**14.55**	8.98
16	13.4	23.6	18.6	**37.0**	11.26	22.01	17.46	**39.21**
32	17.8	34.4	19.6	**51.2**	16.32	33.39	17.46	**51.86**
64	27.8	40.4	19.6	**69.2**	26.31	35.92	17.58	**60.59**
128	37.4	26.2	12.4	$\boxed{\textbf{79.0}}$	35.80	22.77	10.12	$\boxed{\textbf{63.12}}$

Note: The best performance for a fixed code size is shown in **boldface**. and the best result amongst all the comparative methods is $\boxed{\text{frameboxed}}$.

of neurons 'relevant' to the query neuron, which is the top K nearest neighbours depined upon the normalized Euclidean distance in the neuromorphological space and $\mathcal{N}_{\mathcal{H}}(n_i)$ represents the retrieved neurons through hashing as a set of k morphologically similar neurons. We measeur the $F1_{score}$ for all comparative methods by varying the codeblock size from 8 bytes to 128 bytes in geometric order of 2. We cmopare the performance for two cases of top 10 and 25 retrieved neurons, as tabulated in Table 2.

Observations on Experiment 2: Table 2 demonstrates the superiority of proposed retrieval technique over comparative methods for both cases (10 NN and 25NN), when sufficient code size was chosen ($>$ 8 bytes). It must be noted that we chose larger code sizes over conventional code sizes ($>$ 16 bytes), as it was observed that precision-recall performances for HF and comparative methods for smaller code sizes were not sufficient enough for the application at hand. HF is able to encode distinct subspaces of the neuromorphological space better due to its hierarchical tree structure. It must also be noted that for larger code sizes, the retrieval performance of both SH and STH significantly decreases (from 64 to 128 bytes), as noisy eigenvectors corresponding to higher eigenvalues are increasingly used in encoding. For comparable performance, LSH needs a large code size to achieve high performance which is computationally expensive (corroborates observations reported by [11]). We also observe that there is no significant improvement in the performance of STH with increasing code size, though it outperforms all methods for small code size (8 bytes). We further infer that increasing code size rather augments HF performance, thus HF is more scalable towards large code sizes than the comparative methods.

4 Conclusions

In this paper, we present hashing forests as an effective approach for accurate retrieval of morphologically similar neurons in large neuroscientific image databases. HF uses unsupervised random forests to extract compact representation of neuron morphological features that enables efficient query, retrieve and analysis of neurons. The use of ensemble of trees and hierarchical tree-structure makes hashing forests more robust to noisy neuromorphological features (observed due to inconsistent 3D digital reconstruction of neuron). Due to independent training, HF is easily scalable to large code sizes as the database size and heterogeneity increases. To the best of our knowledge, this is the first paper to present hashing in neuroscientific databases and demonstrates higher flexibility for reference-based retrieval over existing alternative methods [14]. HF is established to preserve morphological similarity while encoding extracts has better precision and recall *per* bit than the comparative methods. The formulation for HF is generic and it can be easily leveraged for other large-scale reference based retrieval systems. The proposed formulation utilizes inverse coding in HF which helps avoid pairwise comparisons across the database while retrieving, without compromising on retrieval performance.

References

1. Costa, L.D.F., et al.: Unveiling the neuromorphological space. Front. Comput. Neurosci. 4, 150 (2010)
2. Costa, M., et al.: NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. bioRxiv, 006346 (2014)
3. Ascoli, G.A., et al.: NeuroMorpho. Org: a central resource for neuronal morphologies. J. Neurosci. 27(35), 9247–9251 (2007)
4. Rautenberg, P.L., et al.: NeuronDepot: keeping your colleagues in sync by combining modern cloud storage services, the local file system, and simple web applications. Front. Neuroinform. 8, 55 (2014)
5. Polavaram, S., et al.: Statistical analysis and data mining of digital reconstructions of dendritic morphologies. Front. Neuroanat. 8, 138 (2014)
6. Slaney, M., et al.: Locality-sensitive hashing for finding nearest neighbors. IEEE Signal Process. Mag. 25(2), 128–131 (2008)
7. Weiss, Y., et al.: Spectral hashing. In: NIPS, pp. 1753–1760 (2009)
8. Zhang, D., et al.: Self-taught hashing for fast similarity search. In: ACM SIGIR 2010, vol. 33, pp. 18–25 (2010)
9. Zhang, X., et al.: Towards Large-Scale Histopathological Image Analysis: Hashing-Based Image Retrieval. IEEE Trans. Med. Imaging 34(2), 496–506 (2015)
10. Syeda-Mahmood, T., Wang, F., Kumar, R., Beymer, D., Zhang, Y., Lundstrom, R., McNulty, E.: Finding similar 2D X-ray coronary angiograms. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds.) MICCAI 2012, Part III. LNCS, vol. 7512, pp. 501–508. Springer, Heidelberg (2012)
11. Yu, G., et al.: Scalable forest hashing for fast similarity search. In: IEEE ICME 2014, pp. 1–6 (2014)
12. Scorcioni, R., et al.: L-Measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. Nat. Protoc. 3(5), 866–876 (2008), `http://cng.gmu.edu:8080/Lm/`
13. Moosmann, F., et al.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS, pp. 985–992 (2007)
14. Neuromorpho, `http://neuromorpho.org/neuroMorpho/MorphometrySearch.jsp`
15. Neuromorpho Search, `http://neuromorpho.org/neuroMorpho/index.jsp`

16. Jacobs, B., et al.: Regional dendritic and spine variation in human cerebral cortex: a quantitative golgi study. Cereb. Cortex (2001)
17. Anderson, K., et al.: The morphology of supragranular pyramidal neurons in the human insular cortex: a quantitative Golgi study. Cereb. Cortex 19, 2131–2144 (2009)
18. Kong, J.H., et al.: Diversity of ganglion cells in the mouse retina: unsupervised morphological classification and its limits. J. Comp. Neurol. 489(3), 293–310 (2005)
19. Coombs, J., et al.: Morphological properties of mouse retinal ganglion cells. Neuroscience 140(1), 123–136 (2012)
20. Rodger, J., et al.: Long-term gene therapy causes transgene-specific changes in the morphology of regenerating retinal ganglion cells. PLoS One 7(2), e31061 (2012)
21. Chiang, A.S., et al.: Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution. Curr. Biol., 1–11 (2011)
22. Jacobs, B., et al.: Neuronal morphology in the African elephant (Loxodonta africana) neocortex. Brain Struct. Funct. 215, 273–298 (2010)
23. Jacobs, B., et al.: Life-span dendritic and spine changes in areas 10 and 18 of human cortex: a quantitative Golgi study. J. Comp. Neurol. 386(4), 661–680 (1997)