frontiers in Neuroinformatics

METHODS
published: 02 March 2022
doi: 10.3389/fninf.2022.828169

RealNeuralNetworks.jl: An Integrated Julia Package for Skeletonization, Morphological Analysis, and Synaptic Connectivity Analysis of Terabyte-Scale 3D Neural Segmentations

Jingpeng Wu[1]†, Nicholas Turner[1,2], J. Alexander Bae[1,3], Ashwin Vishwanathan[1] and H. Sebastian Seung[1,2]*

[1] Princeton Neuroscience Institute, Princeton University, Princeton, NJ, United States, [2] Department of Computer Science, Princeton University, Princeton, NJ, United States, [3] Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ, United States

OPEN ACCESS

Edited by:
Ting Zhao,
Janelia Research Campus,
United States

Reviewed by:
Dawen Cai,
University of Michigan, United States
Daniel Raimund Berger,
Harvard University, United States

***Correspondence:**
Jingpeng Wu
jwu@flatironinstitute.org

†Present address:
Jingpeng Wu,
Center for Computational Neuroscience, Flatiron Institute, New York, NY, United States

Received: *03 December 2021*
Accepted: *10 February 2022*
Published: *02 March 2022*

Citation:
Wu J, Turner N, Bae JA, Vishwanathan A and Seung HS (2022) RealNeuralNetworks.jl: An Integrated Julia Package for Skeletonization, Morphological Analysis, and Synaptic Connectivity Analysis of Terabyte-Scale 3D Neural Segmentations. Front. Neuroinform. 16:828169. doi: 10.3389/fninf.2022.828169

Benefiting from the rapid development of electron microscopy imaging and deep learning technologies, an increasing number of brain image datasets with segmentation and synapse detection are published. Most of the automated segmentation methods label voxels rather than producing neuron skeletons directly. A further skeletonization step is necessary for quantitative morphological analysis. Currently, several tools are published for skeletonization as well as morphological and synaptic connectivity analysis using different computer languages and environments. Recently the Julia programming language, notable for elegant syntax and high performance, has gained rapid adoption in the scientific computing community. Here, we present a Julia package, called RealNeuralNetworks.jl, for efficient sparse skeletonization, morphological analysis, and synaptic connectivity analysis. Based on a large-scale Zebrafish segmentation dataset, we illustrate the software features by performing distributed skeletonization in Google Cloud, clustering the neurons using the NBLAST algorithm, combining morphological similarity and synaptic connectivity to study their relationship. We demonstrate that RealNeuralNetworks.jl is suitable for use in terabyte-scale electron microscopy image segmentation datasets.

Keywords: skeletonization, morphological analysis, clustering, connectomics, Julia language, neuron morphology, neuron connectivity

INTRODUCTION

Neural morphology and synaptic connectivity are closely related to brain function. With both nanometer resolution and a large field of view, advanced Electron Microscopes can produce large-scale image stacks (Kornfeld and Denk, 2018; Yin et al., 2020). Image voxels, pixels in a 3D image volume, can be clustered as individual neurons manually (Kasthuri et al., 2015) or automatically

using computer vision technologies (Lee et al., 2017, 2019, 2021; Januszewski et al., 2018; Macrina et al., 2021). Benefiting from the rapid development of deep learning (LeCun et al., 2015), the performance of automated segmentation approaches has greatly improved (Beier et al., 2017; Lee et al., 2017, 2019). With additional help from proofreading (Kim et al., 2014; Zhao et al., 2018; Dorkenwald et al., 2020; Hubbard et al., 2020), reconstructed neurons with synaptic connectivity can be used for scientific discovery (Deutsch et al., 2020; Januszewski et al., 2020; Vishwanathan et al., 2020).

Neurons are like trees and their skeletons can be used for morphological analysis. Skeleton or centerline representation is widely used in the morphological analysis (Stepanyants and Chklovskii, 2005; Halavi et al., 2008; Cuntz et al., 2011; Parekh and Ascoli, 2013; Armañanzas and Ascoli, 2015). In contrast to manual tracing and getting a neuron skeleton directly, most existing automated segmentation methods produce voxel labeling and are skeletonized in another step.

Synapses can also be detected automatically (Huang et al., 2018; Turner et al., 2020; Buhmann et al., 2021; Liu and Ji, 2021). Synaptic connectivity analysis can be used to detect motifs or communities. Although several software tools exist for each processing or analysis step, they were normally implemented using different computer languages. There is a lack of a consistent computational environment for the whole analysis pipeline, and users have to switch back and forth between different programming languages and environments.

Traditionally, developers normally use an interpreted language for prototyping, such as Python or MATLAB (MathWorks, Inc., Natick, MA, United States), and then translate the code to a compiled language, such as C or C++, to speed up the computation for large scale deployment. This was called a "two-language problem." Although some packages, such as Cython and pypy, can be used to help generate lower-level code, there still exist a lot of restrictions. Recently, a programming language with both intuitive syntax and high performance, called Julia (Bezanson et al., 2017), was designed to tackle this problem and has gotten more and more popular in the scientific computing community (Perkel, 2019). Benefiting from this design, prototype code can be compiled just in time and transformed into efficient binary code. As a result, we do not need to rewrite the prototype code using another low-level language, such as C or C++. Motivated by this elegant design, we use Julia to implement some essential analysis steps, including skeletonization, morphological analysis, and connectivity analysis, in two software packages called RealNeuralNetworks.jl and BigArrays.jl.

MATERIALS

We demonstrate the usage of RealNeuralNetworks.jl by analyzing a dataset with some proofread neurons. The details of this dataset, including sample preparation, imaging, automated segmentation, proofreading, was previously reported (Vishwanathan et al., 2017, 2021). Briefly, a sample (about 250 μm $\times$ 120 μm $\times$ 80 μm) from a zebrafish larvae brainstem was stained, sectioned, and imaged using a Zeiss Sigma field emitting scanning electron microscope. The image voxel size is 5 nm $\times$ 5 nm $\times$ 45 nm, and the final image volume size is over four terabytes with a voxel bit-depth of 8 (256 gray levels). Images are aligned and segmented automatically using a convolutional neural network (Lee et al., 2017; Wu et al., 2021). Based on the automated segmentation, about three thousand objects, including neurons or orphan neurites, were proofread using a modified Eyewire system (Kim et al., 2014; Greene et al., 2016; Bae et al., 2018; Vishwanathan et al., 2021). The final plain segmentation was exported to Google Cloud and visualized using Neuroglancer (Maitin-Shepard, 2021; **Figure 1**).

METHODS AND RESULTS

Data Storage

Segmentation and skeleton data are stored in Google Cloud Storage. The cutout and saving of segmentation chunks are implemented in a standalone Julia package, called BigArrays.jl (see section "Code Availability"). This is similar in functionality to the Python package CloudVolume (Charles et al., 2020; Silversmith et al., 2021b), and the data format is compatible with both packages. The cutout and saving of chunks were implemented on the client, so no intermediate server was needed. Benefiting from the distributed storage system in the cloud, the cutout and saving performance scales linearly with the number of operations. Besides skeletonization, BigArrays.jl was designed for general usage and could be used to handle arrays that are too large to fit in RAM. For example, a potential application is solving the out-of-memory issue in the simulation of quantum computing using tensor networks (Fishman et al., 2020) (Personal Communication).

For skeletonization, we can store the results in several formats. Currently, we support SWC with plain text encoding (Ascoli et al., 2001) which is widely used in most other analysis tools. Additionally, We also created a customized binary representation of SWC and all the numbers are encoded as binary scalar values directly and the loading and saving speed is greatly accelerated. For the synapses, it was detected externally and the result was saved using a language agnostic format "CSV."

Additionally, the data, including segmentation volume and skeletons, are formatted following Neuroglancer Precomputed. As a result, the data could be visualized directly using Neuroglancer (Maitin-Shepard, 2021) once they are uploaded to the cloud storage without any additional work.

Distributed Skeletonization of Neurons

To speed up skeletonization, we implemented the hybrid cloud distributed computation architecture in python-based chunkflow (Wu et al., 2021). The object IDs were used to define tasks and all the IDs were ingested to a queue in Amazon Simple Queue Service (SQS) using a Julia package called AWSSDK.jl (2021). The skeletonization of each neuron is independent of each other, so performance scales linearly with the number of nodes allocated.

Because task management (in SQS) and storage management are both distributed, we can launch workers on any computer with an internet connection and cloud authentication. Each

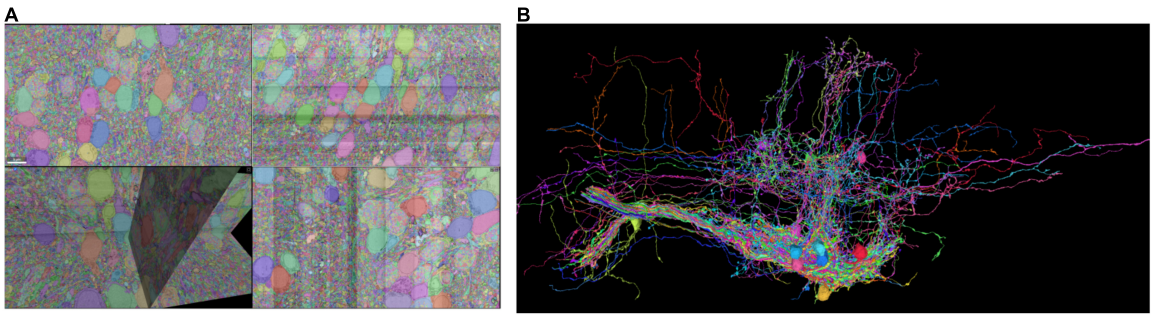

FIGURE 1 | Sparse segmentation after proofreading. **(A)** Some of the neurons are proofread and the fragments are agglomerated as individual neurons. **(B)** Some of the proofread neurons are visualized.

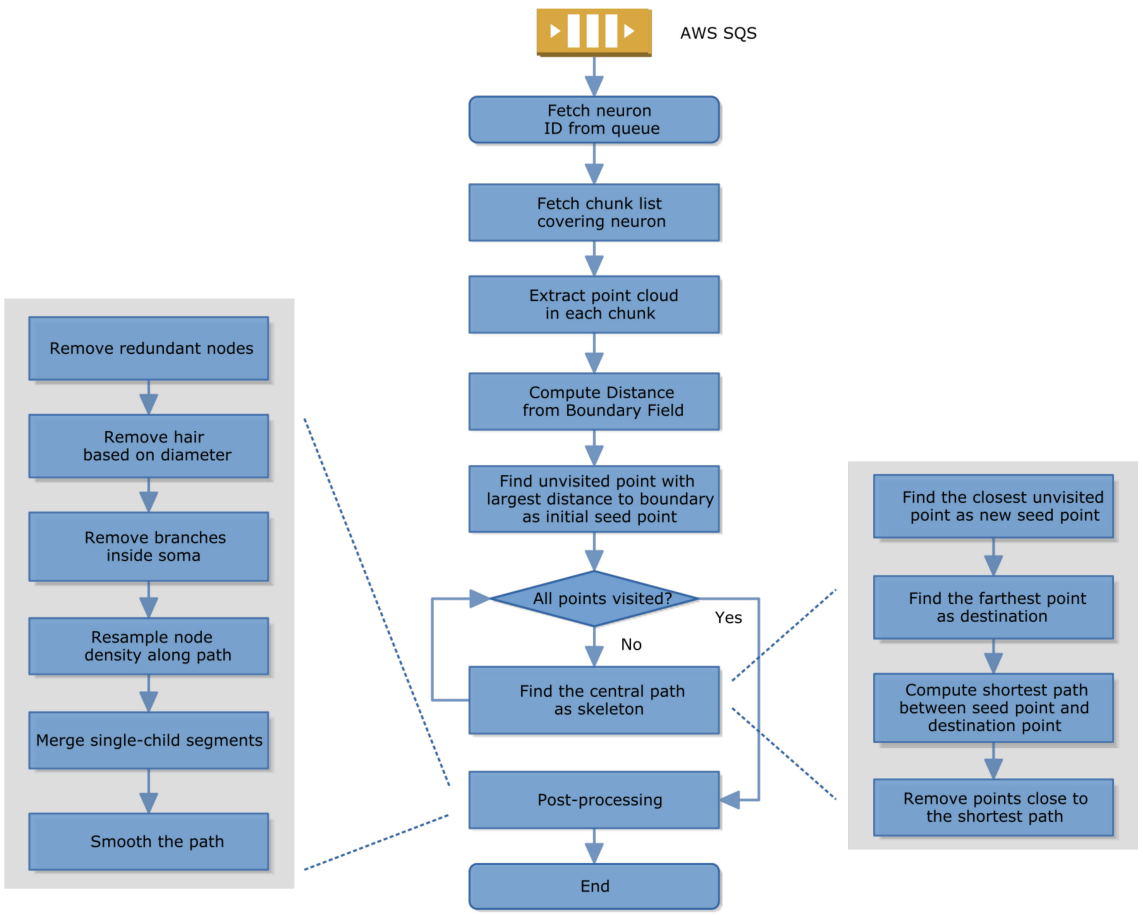

FIGURE 2 | Skeletonization computation in a worker.

task performs skeletonization for one object, called sparse skeletonization. The computation pipeline on the worker uses a modified TEASAR algorithm (Sato et al., 2000; Bae et al., 2018; Silversmith et al., 2021a). Briefly, the steps are as follows (**Figure 2**).

1. A worker fetches a task from SQS;
2. It then fetches the segmentation chunk list covering that object or neuron;
3. It extracts the point cloud of that object; It computes the distance from the boundary of the binary mask of that object;
4. It finds a point with the largest distance to the boundary as a seed;
5. If not all the points are visited, find a new central path by computing the shortest path from seed to the furthest unvisited point and then mark all the nearby points as unvisited;

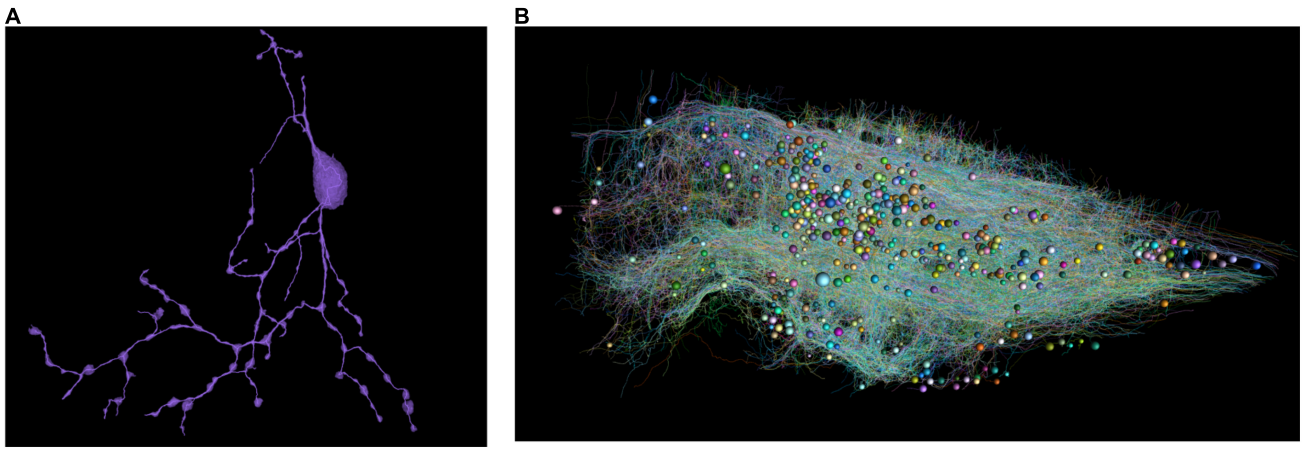

FIGURE 3 | (A) Skeletonize of a single neuron. Note that broken parts were reconnected. **(B)** All the skeletons with a random color. The spheres represent cell bodies with varying diameters.

6. If all the points have already been visited, the skeletonization is done and it switches to postprocessing, including removing redundant nodes, removing hair by comparing the diameter and path length, removing branches inside the cell body, resampling the node density to make it more evenly distributed along the path, removing empty branches, smoothing.

Given a sparsely or densely segmented volume, we extract the centerline or skeleton of its neurites one by one using a modified TEASAR algorithm (Sato et al., 2000; Bae et al., 2018; Silversmith et al., 2021a). Given a bit-packed binary volume representing a neuron, the foreground voxels are extracted as a point cloud. The distance from each point to the nearest boundary was computed as a Distance from Boundary Field (DBF). Find the point with the largest DBF as a seed point. Construct an undirected graph with points as nodes and neighboring points are connected with edges. Find the farthest unvisited point as the destination and compute the shortest path as the skeleton using Dijkstra's algorithm (Dijkstra, 1959). Points around the skeleton are marked as visited and not used in the following computation. Find the unvisited point closest to visited points as the new seed and iterate until all the points are visited. If the segmentation voxel is not continuous, we can look for the nearest terminal node (**Supplementary Figure 1**) to reconnect within a distance threshold. Note that the binary representation was bit-packed and the memory usage was reduced by 8 fold.

As a result, all proofread neurons are skeletonized (**Figure 3**). The distributed computation was performed in Google Cloud.

TABLE 1 | Features for single neuron morphology analysis.

Features	Description
Segment order	The order increases from the root node while branching
Segment length	The path length of a single segment
Branching angle	The angle of two segments in a branching point
Tortuosity	The curvature of a segment
Distance to root path length	The minimum path distance from the segment to root node
Average radius	The mean of all the nodes radius in the segment
Radius from soma	For each node, the Euclidean distance from the soma
Terminal segment path length	The path length of each terminal segment
The ratio of neck diameter to head	Could be used to identify spines

Morphological Features for Single Neuron Analysis

We decompose each neuron into segments or single nodes and compute their features. Definitions of node, branching node, root node, terminal node, segment, and the terminal segment are in **Supplementary Figure 1**. Additionally, an irreducible node corresponds to a soma, branching node, or terminal node. Based on existing literature (Uylings and van Pelt, 2002; Schierwagen, 2008; Schierwagen et al., 2010; Cuntz et al., 2011), we implemented some widely used morphological features for the skeletons and demonstrated the results using our zebrafish dataset (**Table 1** and **Figure 4**). In the spines of mammalian brains, the diameter of the neck is normally much smaller than the head, thus we added a feature to measure the ratio of neck diameter to head (**Figure 4H**).

Morphological Features of Many Neurons

For a number of neurons, we would like to encode each neuron using a feature vector, which could be used in neuron type clustering. Based on the literature, we have also implemented several widely used features (**Table 2**) and applied them to our zebrafish dataset (**Supplementary Figure 2**; Uylings and van Pelt, 2002; Schierwagen, 2008; Schierwagen et al., 2010; Cuntz et al., 2011; Wanner et al., 2016).

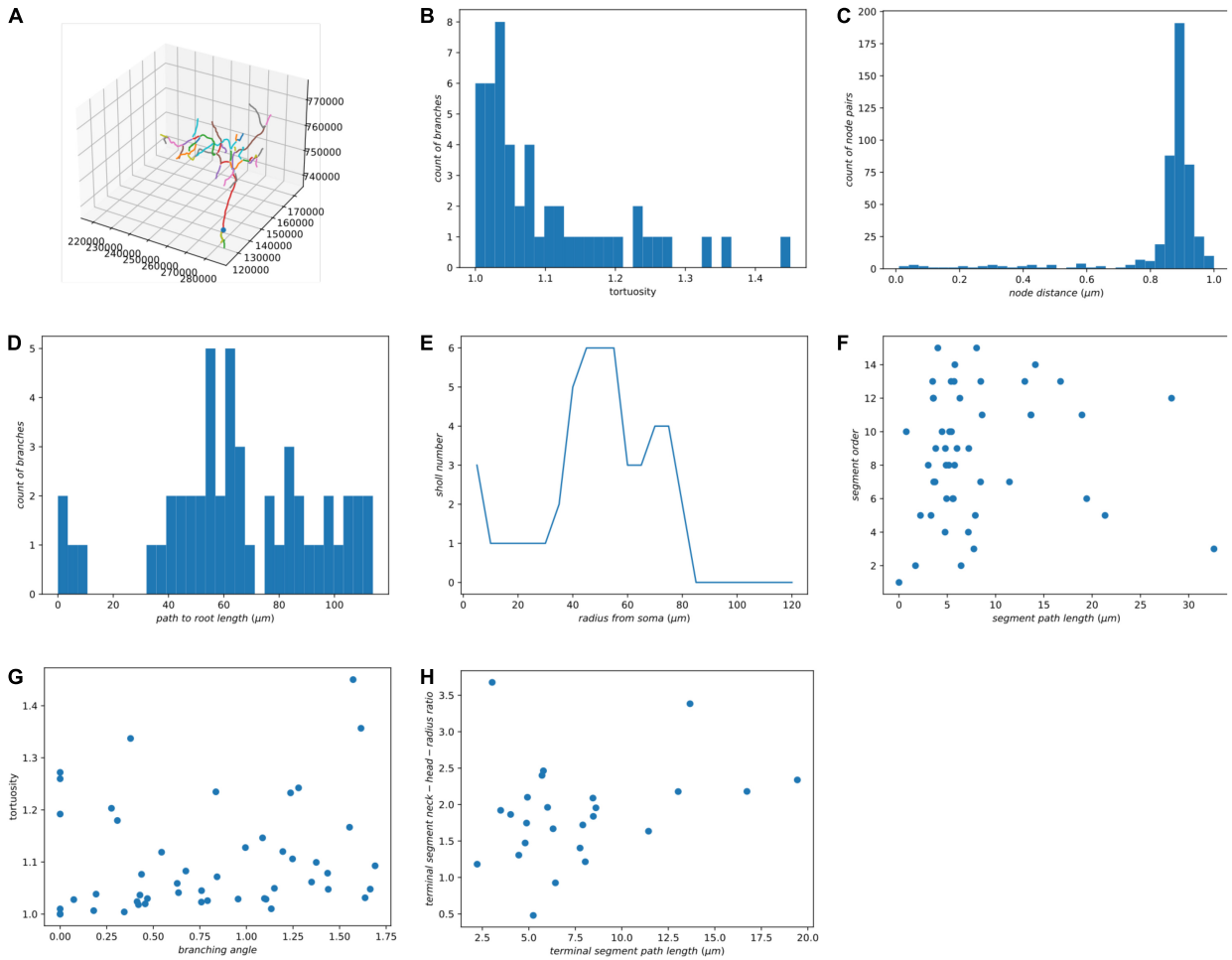

FIGURE 4 | Some morphological features of a single neuron. **(A)** The morphology of a neuron is visualized in Jupyter Notebook. **(B)** Histogram of tortuosity of neuron segments. **(C)** Histogram of neighboring node distance. **(D)** Histogram of path length to the root node. **(E)** Sholl analysis. **(F)** Segment path length versus segment order. **(G)** Branching angle in radians versus tortuosity of segments. **(H)** Terminal segment path length versus terminal segment neck-head radius ratio.

Morphological Clustering Using NBLAST

Most of traditional morphological features do not measure the spatial distribution of neurons. An automatic neuron type classification method, called NBLAST (Costa et al., 2016), measures the spatial distribution and is getting popular. The original method was implemented in R and C++. In order to incorporate this method in our analysis ecosystem, we implemented this algorithm from scratch using Julia. We performed hierarchical clustering (**Supplementary Figure 3**) using Clustering.jl (Stukalov and Lin, 2021) and classified the neurons into 23 types based on the NBLAST similarity scores (**Figure 5**). The visualization was created using Neuroglancer.

Synaptic Connectivity Combined With Morphology

After neuron segmentation and synapse detection were done externally, we can construct a graph of the neural network. Within the graph, the neurons are nodes and the synapses are edges. We use the synapse number as a connectedness metric for neurons. The more synapses connecting two neurons, the closer they are. Based on the distance matrix, we can perform hierarchical clustering, reorder the connectivity matrix, and identify some communities (**Figure 6A**).

Once we have the skeleton morphological features and synaptic connectivity, we can combine them. We can order the neurons in the connectivity matrix using NBLAST hierarchical clustering. As a result (**Figure 6B**), there are some morphologically similar neurons highly connected with each other. Morphologically similar neurons tend to have stronger synaptic connections as well (**Figure 6C**), which is consistent with previous findings in the mouse visual cortex (Lee et al., 2016).

DISCUSSION

RealNeuralNetworks.jl was built to process voxel segmentation datasets from Serial Section Electron Microscopy images.

TABLE 2 | Features of a single neuron.

Features	Description
Distance from soma to the center of skeleton mass	A metric to measure symmetricity centered by soma
Total path length	The physical length of all the skeleton paths
The number of branching points	
Median segment length	The median segment length of all the segments starts and ends at irreducible nodes
3D Sholl Analysis (Sholl, 1953)	Count the intersections to spheres centered on the root node
Average branching radian	The mean of the branching angles
Average tortuosity	The average value of the ratio of the path length to the Euclidean distance between irreducible nodes
Asymmetry	The distance of the soma node to the arbor center of mass
Typical radius	The Euclidian distance of the dendritic arbor points to the center of mass
Fractal dimension	Measures similarity across scales
Root node radius	The radius of the root node which is normally the soma
Total dendrite path length	If the dendrite segments are classified
Longest segment path length	
Convex hull volume	
Surface area	
Post-synapse number	Number of postsynaptic sites
Pre-synapse number	Number of presynaptic sites

Some components, such as skeletonization and morphological analysis, can be reused for sparsely labeled neurons in Light Microscopy images.

Comparison With Related Tools

Most existing tools are specifically designed for one or two analysis steps, rather than providing a one-stop solution and a consistent computational environment. Compared with some related software, RealNeuralNetworks.jl has a more complete toolset for the analysis (**Table 3**).

NeuTu (Zhao et al., 2018) was built mainly for proofreading neuron reconstruction from Electron Microscopy images. Besides that, it can also measure neuron shape similarity and perform clustering of neuron types (Zhao and Plaza, 2014). The measurement is built upon arbor density maps which is much more computationally heavy than skeleton-based NBLAST (Costa et al., 2016). Although the sparse skeletonization of NeuTu was also built upon the TEASAR algorithm (Sato et al., 2000), the geodesic distance between neighboring voxels is measured using the image intensity rather than distance map in our implementation. Thus, the skeleton accuracy is correlated with image quality.

Currently, RealNeuralNetworks.jl only has some widely used morphological features and is not as complete as L-Measure (Scorcioni et al., 2008) and TREES toolbox (Cuntz et al., 2010). Vaa3D (Peng et al., 2010, 2014) was built for light microscopy image processing, especially neuron tracing, and has a much richer set of tracing algorithms.

Kimimaro (Silversmith et al., 2021a) was built for dense skeletonization rather than sparse skeletonization. Currently, it does not have a bit-packed binary representation of segmentation volume and requires much more memory for sparse usage.

Why Julia

Julia is a modern language with nice features for both scientific computing and general programming (Bezanson et al., 2017; Perkel, 2019). It performs just-in-time compilation for the code, so performance can be comparable with C/C++. In addition, it has an intuitive syntax and an interactive programming interface like MATLAB (MathWorks, Inc., Natick, MA, United States), which is useful for prototyping and experiments. It is open-source with a permissive license, so it is much easier to deploy in the cloud compared with commercial languages requiring a license, such as MATLAB (MathWorks, Inc., Natick, MA, United States). Julia can be used interactively in Jupyter Notebooks (The "Ju" is from the name of Julia) (Perkel, 2019). Julia is increasingly popular in the scientific computing community. It has been downloaded over 25 million times and over 5000 packages are registered.

For most of the interpretable languages, such as Python and MATLAB, manipulating single elements in one or nesting loop is normally tens or hundreds of times slower than low-level languages, such as C and C++. For good performance, programmers are limited to using "vectorized" operations which were actually implemented in lower-level languages. In our applications, we perform a lot of voxel manipulations that are hard to express in "vectorized" operations. Benefiting from the just-in-time compilation, all of such operations can be implemented directly in Julia with good performance.

For the computation in local cluster or supercomputer, Julia was designed for distributed computing at the beginning and has gained a dramatic rise in the high-performance computing community. Our packages are expected to be adaptable in a local cluster.

Limitations

The skeletonization module was designed for sparse skeletonization rather than dense skeletonization. For sparse skeletonization, we can skeletonize some neurons of interest while the proofreading is ongoing. It would be too computationally expensive to iterate over the neurons individually in a terabyte-scale or petabyte-scale image volume. For dense skeletonization, Kimimaro is a better alternative (Silversmith et al., 2021a).

Currently, RealNeuralNetworks.jl only has limited support for visualization, such as functions for skeleton visualization. For more complicated plots, users must build their own scripts or Jupyter Notebooks based on other Julia visualization packages. Compared with the TREES toolbox (Cuntz et al., 2010, 2011), RealNeuralNetworks.jl does not have an interactive skeleton editing interface. Compared with

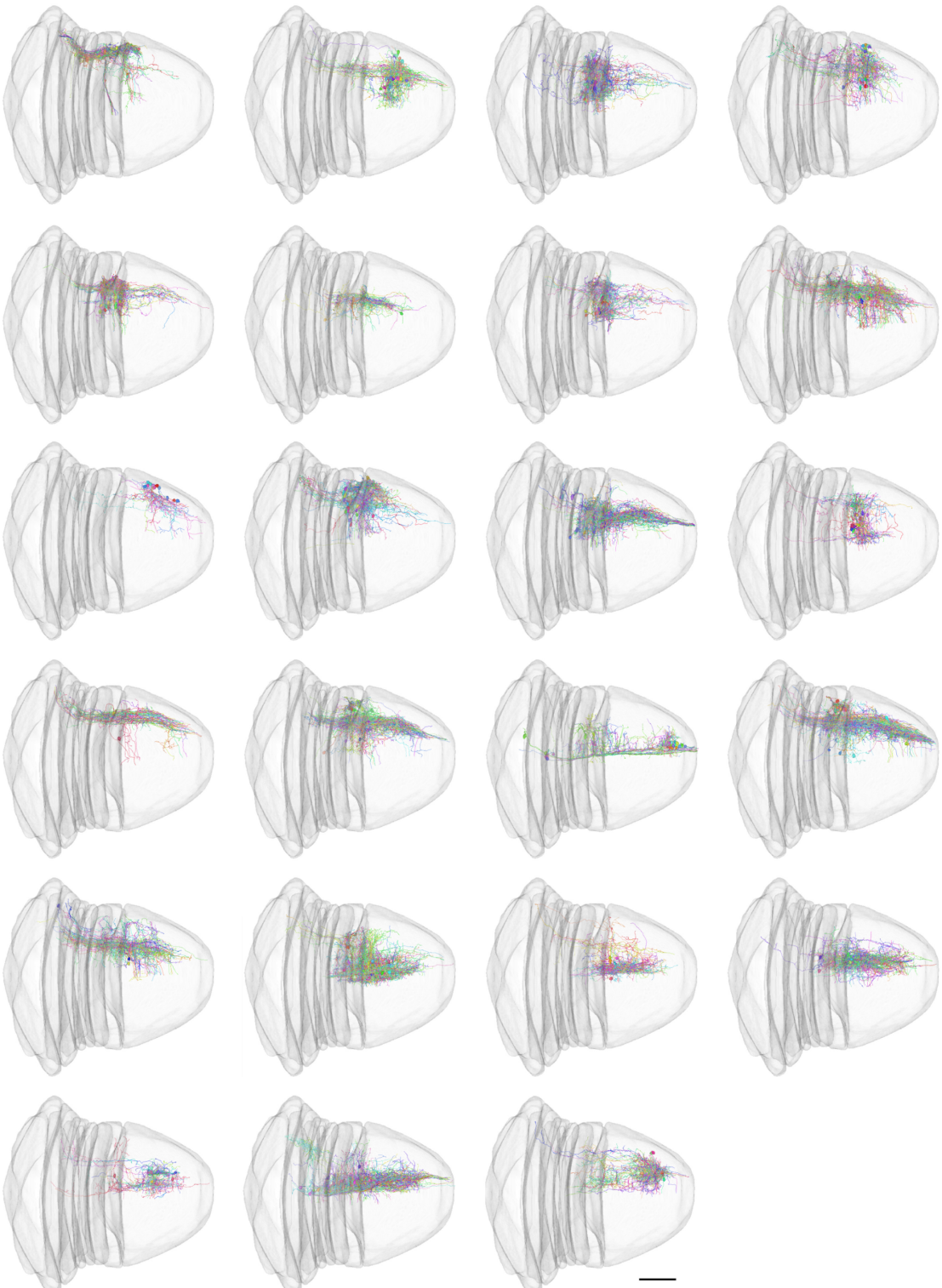

FIGURE 5 | NBLAST classification of neurons. The scale bar in the last image is 100 μm.

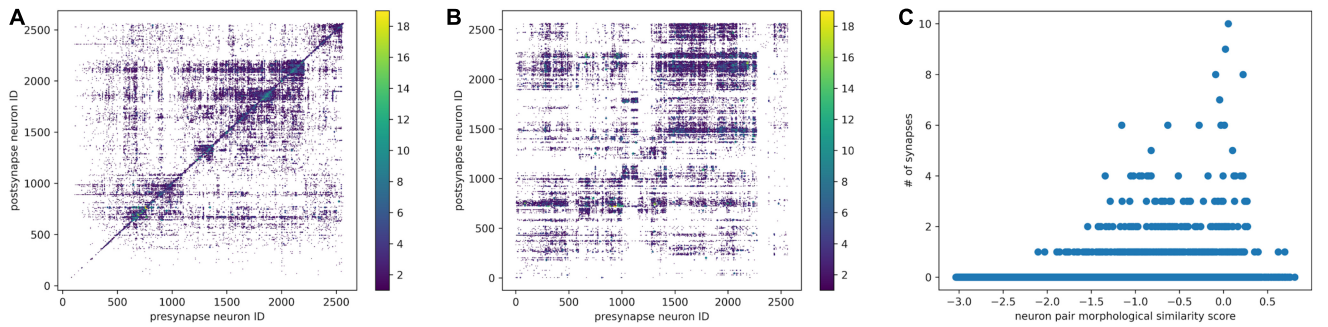

FIGURE 6 | Combine morphological NBLAST clustering and synaptic connectivity. **(A)** The synaptic connectivity matrix was reordered by hierarchical clustering based on connectivity distance. **(B)** The synaptic connectivity matrix was reordered according to hierarchical clustering based on the NBLAST score. The synapse number is encoded in the point diameter and color. **(C)** For each neuron pair, the relationship between NBLAST morphological similarity and number of synapses.

TABLE 3 | Comparison of software tools.

Tool/Feature	References	Language	Skeletonization	Morphological features	NBLAST similarity	Synaptic connectivity
L-Measure	Scorcioni et al., 2008	Java		✓		
NBLAST	Costa et al., 2016	R, C++			✓	
NeuroM	Palacios et al., 2021	Python		✓		
NeuTu	Zhao and Plaza, 2014	C++	✓			
TREES toolbox	Cuntz et al., 2010	MATLAB		✓		
Vaa3D	Peng et al., 2014	C++	✓	✓		
CBLAST	Januszewski et al., 2020	Python, R, C++			✓*	✓
3D BrainCV	Wu et al., 2014	MATLAB	✓	✓		
Kimimaro	Silversmith et al., 2021a	Python, C++	✓			
RealNeuralNetworks.jl		Julia	✓	✓	✓	✓

**CBLAST uses NBLAST for similarity measure.*

L-Measure, there are some missing morphological features in RealNeuralNetworks.jl.

Julia is a young language with rapid development and adoption in the scientific computing community. However, many of the packages are still evolving and are not yet stable.

CONCLUSION

In summary, we present a Julia-based tool, called RealNeuralNetworks.jl, for sparse skeletonization, morphological analysis, and synaptic connectivity analysis. We provide an integrated computational environment for the analysis pipeline. We have demonstrated the utility of this package by processing a Zebrafish segmentation dataset. We hope that it could be useful for other connectomics projects in the future.

CODE AVAILABILITY

The code is open-sourced in GitHub: https://github.com/seung-lab/RealNeuralNetworks.jl. The BigArrays.jl is available in GitHub as well: https://github.com/seung-lab/BigArrays.jl. The Jupyter Notebooks are available in GitHub: https://github.com/jingpengw/realneuralnetworks-notebook.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

JW implemented the software, performed the experiments, and wrote the manuscript. NT translated the MATLAB skeletonization code to Julia. JB improved the TEASAR algorithm and implemented it in MATLAB. AV contributed to sample preparation, imaging, and management of proofreading. HS designed and conceptualized the study. All authors contributed to the article and approved the submitted version.

FUNDING

HS acknowledges support from the NIH/NEI R01EY027036, NIH R01 NS104926, and R01 EY027036. HS acknowledges support from NIH/NCI UH2CA203710, ARO W911NF-14-1-0407, and Mathers Foundation, as well as assistance from Google, Amazon, and Intel. HS is grateful for support from the Intelligence Advanced Research Projects Activity (IARPA) via Department

of Interior/Interior Business Center (DoI/IBC) contract number D16PC0005. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation thereon.

ACKNOWLEDGMENTS

We would like to thank Merlin Moore, Kyle Wille, Ryan Willie, Selden Koolman, Sarah Morejohn, Ben Silverman, Doug Bland, Celia David, Sujata Reddy, Anthony Pelegrino, Sarah Williams, and Dan Visser for manual annotation and validation, Amy Robinson for EyeWire management, Kisuk Lee for convolutional neural network training, Will Wong and William M. Silversmith for image data transformation for Eyewire, and William M. Silversmith and Pat Gunn for manuscript editing.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fninf.2022.828169/full#supplementary-material

Supplementary Figure 1 | Nomenclature of neuron skeleton parts.

Supplementary Figure 2 | Distribution of morphological features.

Supplementary Figure 3 | Hierarchical clustering using the NBLAST score.

REFERENCES

Armañanzas, R., and Ascoli, G. A. (2015). Towards the automatic classification of neurons. *Trends Neurosci.* 38, 307–318. doi: 10.1016/j.tins.2015.02.004

Ascoli, G. A., Krichmar, J. L., Nasuto, S. J., and Senft, S. L. (2001). Generation, description and storage of dendritic morphology data. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.* 356, 1131–1145. doi: 10.1098/rstb.2001.0905

AWSSDK.jl (2021). *JuliaCloud*. Available online at: https://github.com/JuliaCloud/AWSSDK.jl (accessed February 1, 2022).

Bae, J. A., Mu, S., Kim, J. S., Turner, N. L., Tartavull, I., Kemnitz, N., et al. (2018). Digital museum of retinal ganglion cells with dense anatomy and physiology. *Cell* 173, 1293.e19–1306.e19. doi: 10.1016/j.cell.2018.04.040

Beier, T., Pape, C., Rahaman, N., Prange, T., Berg, S., Bock, D. D., et al. (2017). Multicut brings automated neurite segmentation closer to human performance. *Nat. Methods* 14, 101–102. doi: 10.1038/nmeth.4151

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. (2017). Julia: a fresh approach to numerical computing. *SIAM Rev.* 59, 65–98. doi: 10.1137/141000671

Buhmann, J., Sheridan, A., Malin-Mayor, C., Schlegel, P., Gerhard, S., Kazimiers, T., et al. (2021). Automatic detection of synaptic partners in a whole-brain Drosophila electron microscopy data set. *Nat. Methods* 18, 771–774. doi: 10.1038/s41592-021-01183-7

Charles, A. S., Falk, B., Turner, N., Pereira, T. D., Tward, D., Pedigo, B. D., et al. (2020). Toward community-driven big open brain science: open big data and tools for structure, function, and genetics. *Annu. Rev. Neurosci.* 43, 441–464. doi: 10.1146/annurev-neuro-100119-110036

Costa, M., Manton, J. D., Ostrovsky, A. D., Prohaska, S., and Jefferis, G. S. X. E. (2016). NBLAST: rapid, sensitive comparison of neuronal structure and construction of neuron family databases. *Neuron* 91, 293–311. doi: 10.1016/j.neuron.2016.06.012

Cuntz, H., Forstner, F., Borst, A., and Häusser, M. (2010). One rule to grow them all: a general theory of neuronal branching and its practical application. *PLoS Comput. Biol.* 6:e1000877. doi: 10.1371/journal.pcbi.1000877

Cuntz, H., Forstner, F., Borst, A., and Häusser, M. (2011). The TREES Toolbox—Probing the basis of axonal and dendritic branching. *Neuroinformatics* 9, 91–96. doi: 10.1007/s12021-010-9093-7

Deutsch, D., Pacheco, D., Encarnacion-Rivera, L., Pereira, T., Fathy, R., Clemens, J., et al. (2020). The neural basis for a persistent internal state in Drosophila females. *eLife* 9:e59502. doi: 10.7554/eLife.59502

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–271. doi: 10.1007/bf01386390

Dorkenwald, S., McKellar, C., Macrina, T., Kemnitz, N., Lee, K., Lu, R., et al. (2020). FlyWire: online community for whole-brain connectomics. *bioRxiv [preprint]* doi: 10.1101/2020.08.30.274225

Fishman, M., White, S. R., and Stoudenmire, E. M. (2020). *The ITensor Software Library for Tensor Network Calculations. arXiv:2007.14822*. Available online at: http://arxiv.org/abs/2007.14822 (accessed November 2, 2021).

Greene, M. J., Kim, J. S., and Seung, H. S. (2016). Analogous convergence of sustained and transient inputs in parallel on and off pathways for retinal motion computation. *Cell Rep.* 14, 1892–1900. doi: 10.1016/j.celrep.2016.02.001

Halavi, M., Polavaram, S., Donohue, D. E., Hamilton, G., Hoyt, J., Smith, K. P., et al. (2008). NeuroMorpho.org implementation of digital neuroscience: dense coverage and integration with the NIF. *Neuroinformatics* 6, 241–252. doi: 10.1007/s12021-008-9030-1

Huang, G. B., Scheffer, L. K., and Plaza, S. M. (2018). Fully-automatic synapse prediction and validation on a large data set. *Front. Neural Circuits* 12:87. doi: 10.3389/fncir.2018.00087

Hubbard, P. M., Berg, S., Zhao, T., Olbris, D. J., Umayam, L., Maitin-Shepard, J., et al. (2020). Accelerated EM connectome reconstruction using 3D visualization and segmentation graphs. *bioRxiv [preprint]* 2020.01.17.909572. doi: 10.1101/2020.01.17.909572

Januszewski, M., Kornfeld, J., Li, P. H., Pope, A., Blakely, T., Lindsey, L., et al. (2018). High-precision automated reconstruction of neurons with flood-filling networks. *Nat. Methods* 15, 605–610. doi: 10.1038/s41592-018-0049-4

Januszewski, M., Maitin-Shepard, J., Blakely, T., Leavitt, L. J., Li, P. H., Lindsey, L., et al. (2020). A connectome and analysis of the adult Drosophila central brain. *eLife* 9:e57443. doi: 10.7554/eLife.57443

Kasthuri, N., Hayworth, K. J., Berger, D. R., Schalek, R. L., Conchello, J. A., Knowles-Barley, S., et al. (2015). Saturated reconstruction of a volume of neocortex. *Cell* 162, 648–661. doi: 10.1016/j.cell.2015.06.054

Kim, J. S., Greene, M. J., Zlateski, A., Lee, K., Richardson, M., Turaga, S. C., et al. (2014). Space-time wiring specificity supports direction selectivity in the retina. *Nature* 509, 331–336. doi: 10.1038/nature13240

Kornfeld, J., and Denk, W. (2018). Progress and remaining challenges in high-throughput volume electron microscopy. *Curr. Opin. Neurobiol.* 50, 261–267. doi: 10.1016/j.conb.2018.04.030

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521:436.

Lee, K., Lu, R., Luther, K., and Seung, H. S. (2021). Learning and segmenting dense voxel embeddings for 3D neuron reconstruction. *IEEE Trans. Med. Imaging* 40, 3801–3811. doi: 10.1109/TMI.2021.3097826

Lee, K., Turner, N., Macrina, T., Wu, J., Lu, R., and Seung, H. S. (2019). Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy. *Curr. Opin. Neurobiol.* 55, 188–198. doi: 10.1016/j.conb.2019.04.001

Lee, K., Zung, J., Li, P., Jain, V., and Seung, H. S. (2017). *Superhuman Accuracy on the SNEMI3D Connectomics Challenge. ArXiv170600120 Cs*. Available online at: http://arxiv.org/abs/1706.00120 (accessed February 18, 2022).

Lee, W.-C. A., Bonin, V., Reed, M., Graham, B. J., Hood, G., Glattfelder, K., et al. (2016). Anatomy and function of an excitatory network in the visual cortex. *Nature* 532, 370–374. doi: 10.1038/nature17192

Liu, Y., and Ji, S. (2021). *CleftNet: Augmented Deep Learning for Synaptic Cleft Detection from Brain Electron Microscopy. ArXiv210104266 Cs*. Available online at: http://arxiv.org/abs/2101.04266 (accessed March 7, 2021).

Macrina, T., Lee, K., Lu, R., Turner, N. L., Wu, J., Popovych, S., et al. (2021). Petascale neural circuit reconstruction: automated methods. *bioRxiv [preprint]* 2021.08.04.455162. doi: 10.1101/2021.08.04.455162

Maitin-Shepard, J. (2021). *WebGL-Based Viewer for Volumetric Data. Goolgle*. Available online at: https://github.com/google/neuroglancer (accessed April 1, 2019).

Palacios, J., lidakanari, Zisis, E., Coste, B., MikeG, Vanherpe, L., et al. (2021). BlueBrain/NeuroM: v3.0.1. *Zenodo* doi: 10.5281/zenodo.5355891

Parekh, R., and Ascoli, G. A. (2013). Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron* 77, 1017–1038. doi: 10.1016/j.neuron.2013.03.008

Peng, H., Bria, A., Zhou, Z., Iannello, G., and Long, F. (2014). Extensible visualization and analysis for multidimensional images using Vaa3D. *Nat. Protoc.* 9, 193–208. doi: 10.1038/nprot.2014.011

Peng, H., Ruan, Z., Long, F., Simpson, J. H., and Myers, E. W. (2010). V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nat. Biotechnol.* 28, 348–353. doi: 10.1038/nbt.1612

Perkel, J. M. (2019). Julia: come for the syntax, stay for the speed. *Nature* 572, 141–142. doi: 10.1038/d41586-019-02310-3

Sato, M., Bitter, I., Bender, M. A., Kaufman, A. E., and Nakajima, M. (2000). "TEASAR: tree-structure extraction algorithm for accurate and robust skeletons," in *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, (IEEE), 281–449. doi: 10.1107/S1600577516011498

Schierwagen, A. (2008). Neuronal morphology: shape characteristics and models. *Neurophysiology* 40, 310–315. doi: 10.1007/s11062-009-9054-7

Schierwagen, A., Villmann, T., Alpár, A., and Gärtner, U. (2010). "Cluster analysis of cortical pyramidal neurons using SOM," in *Proceeding of the Artificial Neural Networks in Pattern Recognition, 4th IAPR TC3 Workshop, ANNPR 2010*, (Cairo), 120–130.

Scorcioni, R., Polavaram, S., and Ascoli, G. A. (2008). L-Measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nat. Protoc.* 3, 866–876. doi: 10.1038/nprot.2008.51

Sholl, D. A. (1953). Dendritic organization in the neurons of the visual and motor cortices of the cat. *J. Anat.* 87:387.

Silversmith, W., Bae, J. A., Li, P. H., and Wilson, A. M. (2021a). Seung-lab/kimimaro: zenodo Release v1. *Zenodo* doi: 10.5281/zenodo.5539913

Silversmith, W., Collman, F., Kemnitz, N., Wu, J., Castro, M., Falk, B., et al. (2021b). Seung-lab/cloud-volume: zenodo Release v1. *Zenodo* doi: 10.5281/zenodo.5671443

Stepanyants, A., and Chklovskii, D. B. (2005). Neurogeometry and potential synaptic connectivity. *Trends Neurosci.* 28, 387–394. doi: 10.1016/j.tins.2005.05.006

Stukalov, A., and Lin, D. (2021). *Clustering.jl*. Julia Statistics. Available online at: https://github.com/JuliaStats/Clustering.jl (accessed September 30, 2021).

Turner, N. L., Lee, K., Lu, R., Wu, J., Ih, D., and Seung, H. S. (2020). "Synaptic partner assignment using attentional voxel association networks," in *Proceeding of the 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, (IEEE), 1–5.

Uylings, H. B. M., and van Pelt, J. (2002). Measures for quantifying dendritic arborizations. *Netw. Comput. Neural Syst.* 13, 397–414.

Vishwanathan, A., Daie, K., Ramirez, A. D., Lichtman, J. W., Aksay, E. R. F., and Seung, H. S. (2017). Electron microscopic reconstruction of functionally identified cells in a neural integrator. *Curr. Biol.* 27, 2137.e3–2147.e3. doi: 10.1016/j.cub.2017.06.028

Vishwanathan, A., Ramirez, A. D., Wu, J., Sood, A., Yang, R., Kemnitz, N., et al. (2020). Modularity and neural coding from a brainstem synaptic wiring diagram. *bioRxiv [preprint]* 2020.10.28.359620. doi: 10.1101/2020.10.28.359620

Vishwanathan, A., Ramirez, A. D., Wu, J., Sood, A., Yang, R., Kemnitz, N., et al. (2021). Predicting modular functions and neural coding of behavior from a synaptic wiring diagram. *bioRxiv [preprint]* 2020.10.28.359620.

Wanner, A. A., Genoud, C., Masudi, T., Siksou, L., and Friedrich, R. W. (2016). Dense EM-based reconstruction of the interglomerular projectome in the zebrafish olfactory bulb. *Nat. Neurosci.* 19, 816–825. doi: 10.1038/nn.4290

Wu, J., He, Y., Yang, Z., Guo, C., Luo, Q., Zhou, W., et al. (2014). 3D BrainCV: simultaneous visualization and analysis of cells and capillaries in a whole mouse brain with one-micron voxel resolution. *NeuroImage* 87, 199–208. doi: 10.1016/j.neuroimage.2013.10.036

Wu, J., Silversmith, W. M., Lee, K., and Seung, H. S. (2021). Chunkflow: hybrid cloud processing of large 3D images by convolutional nets. *Nat. Methods* 18, 328–330. doi: 10.1038/s41592-021-01088-5

Yin, W., Brittain, D., Borseth, J., Scott, M. E., Williams, D., Perkins, J., et al. (2020). A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nat. Commun.* 11:4949. doi: 10.1038/s41467-020-18659-3

Zhao, T., Olbris, D. J., Yu, Y., and Plaza, S. M. (2018). NeuTu: software for collaborative, large-scale, segmentation-based connectome reconstruction. *Front. Neural Circuits* 12:101. doi: 10.3389/fncir.2018.00101

Zhao, T., and Plaza, S. M. (2014). *Automatic Neuron Type Identification by Neurite Localization in the Drosophila Medulla. ArXiv14091892 Cs Q-Bio*. Available online at: http://arxiv.org/abs/1409.1892 (accessed November 18, 2021).

Conflict of Interest: HS has financial interests in Zetta AI LLC. This study received assistance from Google, Amazon, and Intel. These companies were not involved in the study design, collection, analysis, interpretation of data, the writing of this article, or the decision to submit it for publication.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.