Biologically-Constrained Graphs for Global Connectomics Reconstruction

Supplementary Material

Brian Matejek*[1], Daniel Haehn[1], Haidong Zhu[2], Donglai Wei[1], Toufiq Parag[3], and Hanspeter Pfister[1]

[1]Harvard University [2]Tsinghua University [3]Comcast Research

1. Node Generation and Reduction

Traditional two-step agglomeration strategies tend to create very small segments at locations where the affinities are noisy, usually at very thin locations. In particular, the waterz agglomeration strategy tends to create many small segments (over 85% of segments are smaller than $0.01\,\mu m^3$). Many of these small segments are completely contained within a single z slice. These *singletons* often occur on dendritic spines and other thin locations for neuronal processes. Figure 1 shows an over-segmented dendrite with two enlarged spines each containing several singletons each. To reduce the number of singletons, we first identify all segments that lie entirely in one z slice. We then superimpose these singleton segments onto the neighboring slices and merge the singleton with any other segments that have an Intersection over Union score above 0.30. This threshold reduces a large number of the singletons while introducing very few merge errors.

After removing most of the singletons, we further reduce the number of nodes in our graph by identifying small segments and merging them with a large neighbor. We define the threshold for small based on the skeleton generation strategy used during edge generation. Our topological thinning algorithm for skeletonization [8] often erodes very small volumes to a single point. Since we need multiple points in the skeleton to generate our directional vectors, these nodes would not receive edges (Fig. 6, right). Therefore we analyze the number of expressive (non-single point) skeletons above and below several thresholds (Fig. 2). We find that over 84.1% of skeletons for segments larger than $0.010\,36\,\mu m^3$ are expressive and 90.5% of skeletons smaller are trivial. This volume corresponds to $20,000$ voxels in the PNI datasets. On three testing datasets, $50-80\%$ of all segments are below this threshold.

*Corresponding author, bmatejek@seas.harvard.edu

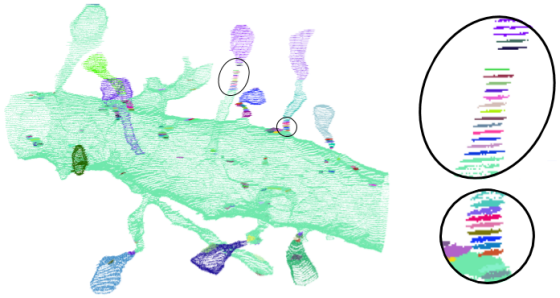

Figure 1. Current agglomeration strategies tend to produce a large number of singletons in thin locations. Here we show an over-segmented dendrite and two spines with many singletons.

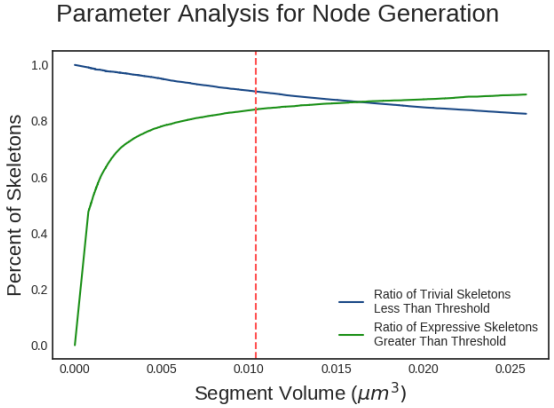

Figure 2. The percent of trivial and expressive skeletons decreases and increases respectively as the volume of segments increases. At $t_{vol} = 0.010\,36\,\mu m^3$ nearly 85% of larger segments have expressive skeletons and 90% of smaller ones are trivial.

2. Node Convolutional Neural Network

We experimented with twelve different network architectures and input configurations (Table 1). We evaluate region of interest (ROI) diameters of $800\,nm$, $1200\,nm$, and

800nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9482	0.9343	0.9332
(3, 60, 60, 20)	0.9575	**0.9427**	**0.9451**
(3, 68, 68, 22)	0.9644	0.9346	0.9369
(3, 76, 76, 24)	**0.9752**	0.9386	0.9403

1200nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9224	0.9096	0.9120
(3, 60, 60, 20)	0.9318	0.9144	0.9190
(3, 68, 68, 22)	0.9417	0.9264	0.9265
(3, 76, 76, 24)	0.9553	0.9257	0.9227

1600nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.8960	0.8851	0.8814
(3, 60, 60, 20)	0.9170	0.9097	0.9100
(3, 68, 68, 22)	0.9176	0.9043	0.9053
(3, 76, 76, 24)	0.9423	0.9085	0.9117

Table 1. We experiment with twelve different networks for the node CNN. We vary the input size to the network and the diameter of the cubic region of interest. Each experiment ran for 2,000 epochs. Our optimal configuration on the validation data uses an $800\,\mathrm{nm}$ ROI with an input size of (60, 60, 20).

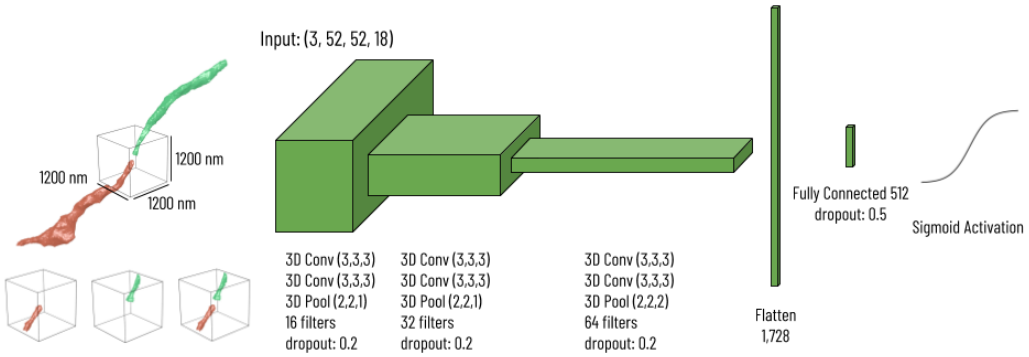

Figure 3. Both neural networks follow the same general architectures with three *VGG-style* convolution blocks with double convolutions followed by a max-pooling operation. Three input channels correspond to if a voxel has label one, label two, or either. The max-pooling of the convolution blocks is anisotropic for the first two and isotropic for the last.

$1600\,\mathrm{nm}$. We find that the overall accuracies on the training, validation, and testing datasets all decrease as the ROI increases. We also vary the input size to the network which in turn changes the number of nodes in the first fully connected layer. As the input size increases the training accuracy tends to increase but the validation and testing accuracies increase then decrease. Thus, we see that the highest validation accuracy occurs with a $400\,\mathrm{nm}$ ROI and a three channel input with (60, 60, 20) voxels in each channel. Each ROI from the input segmentation is upsampled and downsampled to fit into this input size. This allows us to use the same architecture sizes for new datasets with different resolutions. We find that very limited finetuning is needed to adapt a network to a new dataset.

The general architecture for both neural networks follows the same design (Fig. 3). Both have three *VGG-style* blocks with two convolutions of (3, 3, 3) followed by a max-pooling operation [1]. The first two max-pooling operations are anisotropic with downsampling only in the x and y dimensions. The final max-pooling is isotropic. A dropout of 0.2 follows each of these blocks, and there is a final dropout of 0.5 after the last fully connected layer. Each activation function is leaky ReLU after every convolution [4] with $\alpha = 0.001$. The final activation is a sigmoid function. We initialize all weights with Xavier initialization [3].

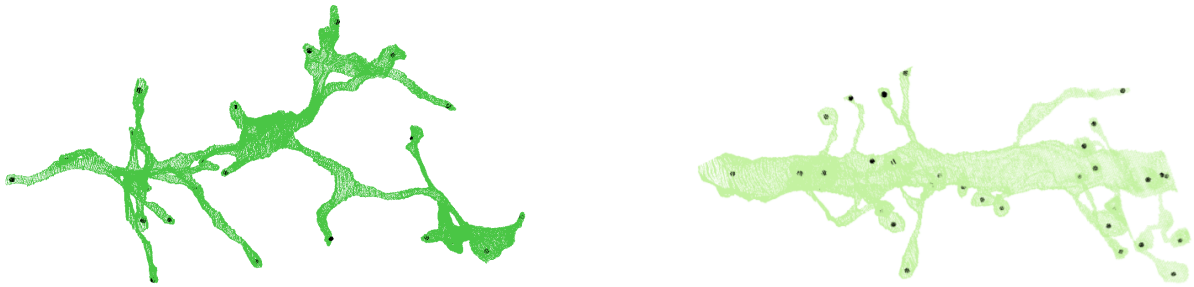

Figure 4. To evaluate various skeletonization methods, we create and publish a skeleton benchmark. We take a ground truth segmentation from the publically available Kasthuri dataset and label the endpoints for the 500 largest segments. Here we show two ground truth segments and their corresponding labeled endpoints.

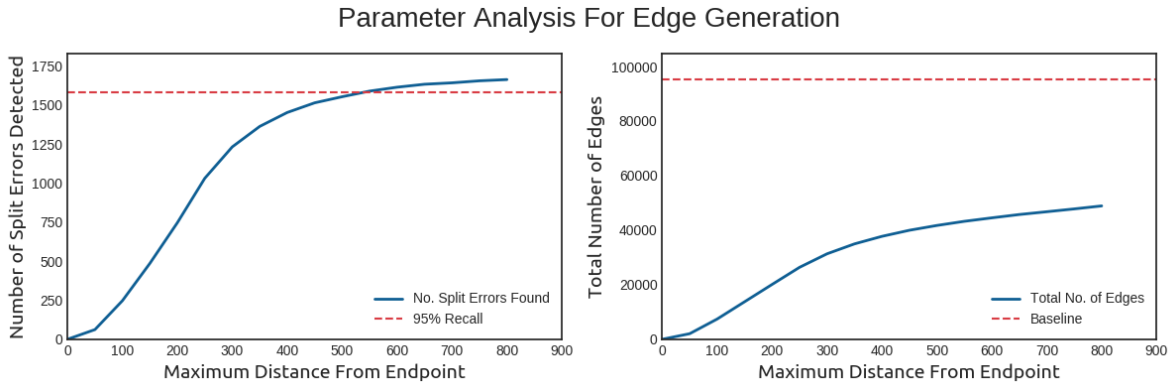

Figure 5. When considering values for t_{edge}—the maximal distance a voxel can be from an endpoint for edge generation—we look at the recall of true split errors and reduction in total edges. At $500\,\mathrm{nm}$ we have 95% recall compared to $800\,\mathrm{nm}$ with 12% fewer edges.

3. Skeleton Benchmark and Generation

Some research focuses on the use of skeletons for quicker connectomics analysis [13] and error correction [2]. Most connectomics papers use the TEASER algorithm [11] or a slight variant from the NeuTu package [12].[1] A significant amount of research in the computer graphics and volume processing communities considers the problem of extracting the medial axis from a 3D volume [6, 8, 9, 10]

Before this work, to our knowledge, no one has done an extensive analysis of various skeletonization approaches on connectomics data. We create and publish a benchmark dataset for skeletonization on connectomics datasets and evaluate three state-of-the-art 3D skeleton extraction techniques. For our benchmark, we consider the 500 largest segments for the ground truth from the Kasthuri training volume. These 500 segments correspond to 95.4% of the volume of the labeled ground truth data. For each of these segments, we identify all endpoints (Fig. 4). For each skeleton generation strategy, we identify all of the endpoints (i.e., those with one or fewer neighbors also belonging to the skeleton). For each segment and skeleton strategy, we look at all of the ground truth and generated endpoints and use a Hungarian matching algorithm to create a one-to-one mapping between the two sets [7]. Endpoint pairs closer than $800\,\mathrm{nm}$ are considered a match and ones farther are not.

We evaluate three different skeleton generation strategies on our benchmark dataset [6, 8, 11]. The medial axis algorithm from Lee et al. is the built-in scipy skeletonization strategy. We publish the benchmark dataset at https://www.rhoana.org/skeletonbenchmark and all code at https://www.rhoana.org/biologicalgraphs. The TEASER algorithm has two tunable parameters, *scale* and *buffer*, which indicate how much pruning to do during generation. Neither the medial axis algorithm or the topological thinning one have any parameters. However, we consider skeleton generation on segments downsampled to isotropic resolutions between $30\,\mathrm{nm}$ and $200\,\mathrm{nm}$ per sample. We achieve the highest F-score with the topological thinning approach [8] after downsampling each segment to a resolution of (80, 80, 80) (precision: 94.7%, recall: 86.7%, F-score: 90.5%). In total, nearly one thousand different parameter settings were evaluated on the benchmark dataset. The medial-axis algorithm and the topological thinning approach achieve similar results, but the topological thinning algorithm is an order of

[1] https://github.com/janelia-flyem/NeuTu

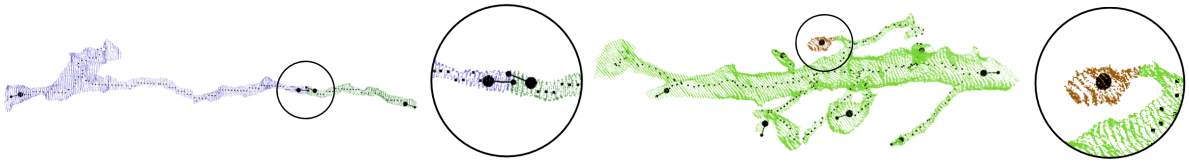

Figure 6. Here we show two more typical success and failure cases for the edge generation strategy. On the left we see a split neuronal process where both skeletons have endpoints with vectors pointing towards their wrongly split neighbor. On the right is an example of a failure case where the segment receives a trivial skeleton with a single endpoint. Since the skeleton is not expressive of the segment shape we have no corresponding vector and this pair of split segments do not receive an edge.

magnitudes faster (4.9 seconds versus 39.8 seconds on the volume downsampled to $80\,\mathrm{nm}$ in each dimension). The TEASER algorithm is the slowest with a running time of 306 seconds on that same volume.

We generate the endpoint vectors in the following way. Consider an endpoint with one neighbor. That neighbor joint is the parent of the endpoint. From the neighbor, we can find the grandparent of the endpoint (i.e., the neighbor of the joint that is not the endpoint). With one more iteration, we find the great-grandparent of an endpoint. The vector between the endpoint and its great-grandparent becomes the endpoint vector used to estimate the direction of the skeleton at endpoint termination.

4. Edge Generation

Figure 5 shows the effect on the number of true split errors identified and the number of total edges as a function of t_{edge}—the maximal allowable distance between a skeleton endpoint and the other neighboring volumes. We consider a wide range of possible distances ranging from $50\,\mathrm{nm}$ to $800\,\mathrm{nm}$ on four training datasets. Figure 5, left, shows the number of edges corresponding to split errors as a function of t_{edge}. The number increases quickly until around $300\,\mathrm{nm}$ before increasing more gradually until $800\,\mathrm{nm}$. We do not consider distances farther than $800\,\mathrm{nm}$ since we found the remaining missing edges are where the algorithm itself cannot find them (paper, Fig. 7, right; supplemental Fig. 6, right). The dotted line shows the location of 95% recall from the $t_{edge} = 800\,\mathrm{nm}$ results. Our proposed method uses a value of $500\,\mathrm{nm}$ since that is roughly at the 95% recall location.

As we can see from Fig. 5, right, the number of edges in our graph increases greatly for every increase in distance until around $400\,\mathrm{nm}$ where the increase becomes more gradual. For that graph, the baseline is determined from the adjacency matrix. A distance of $500\,\mathrm{nm}$ has approximately 12% fewer total edges than using $t_{edge} = 800\,\mathrm{nm}$.

Figure 6 shows an additional success and failure case of the proposed edge generation strategy. On the left, we see a neuronal process incorrectly segmented into two. The circled region shows that each segment has a nearby endpoint with a vector pointing towards the incorrectly split neighbor. On the right, we see a failure case where the small segment receives a singleton skeleton which is not expressive of the overall shape. This occurs on a spine where most of the spine is correctly merged with the dendrite and just the top end of the spine is segmented. Since there are no neighboring skeleton endpoints, these two neighboring segments do not receive an edge.

5. Edge Weights

There are two components to determining the weights for each edge in the graph. First, we need to generate probabilities that two segments belong to the same neuronal process. Second, we need to transform these probabilities into weights for edges.

Edge CNN. The edge CNN which determines if two large segments belong to the same neuron has the same general structure of the node CNN (Fig. 3). There are three *VGG-style* convolution blocks [1] with convolutions of size (3, 3, 3) followed by a max-pooling operation (anisotropic for the first two blocks and isotropic for the third). All weights have Xavier initializations [3] and each activation is leaky ReLU with $\alpha = 0.001$ [4].

We consider a series of architectures, cubic regions, and initial conditions for the edge CNN and choose the one with best validation loss (Table 2). We consider three different input sizes, three different diameters for the cubic region-of-interest (ROI), and finetune a network on the node CNN and train one from scratch. Amazingly, the node CNN produces reasonable results for the new input, with accuracies of 96.3%, 96.6%, and 94.4% on the two PNI testing and one Kasthuri testing datasets. The best validation accuracy has a ROI diameter of $1200\,\mathrm{nm}$ and an input size of (52, 52, 18) voxels. As before, the cubic region is extracted from the input segmentation and upsampled and downsampled as needed to fill the input channels for the network.

From Probabilities to Weights. After generating a probability that two nearby segments belong to the same neuron we need to transform the probability into a weight. As discussed in the main paper, we use the following equation to transform the probabilities into edge weights [5]:

800nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9736	0.9543	0.9576
(3, 52, 52, 18)	0.9750	0.9541	0.9582
(3, 60, 60, 20)	0.9722	0.9586	0.9598
(3, 60, 60, 20)	0.9771	0.9601	0.9626
(3, 68, 68, 22)	0.9740	0.9565	0.9587
(3, 68, 68, 22)	**0.9784**	0.9611	0.9633

1200nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9761	**0.9637**	**0.9638**
(3, 52, 52, 18)	0.9679	0.9553	0.9552
(3, 60, 60, 20)	0.9706	0.9543	0.9546
(3, 60, 60, 20)	0.9703	0.9540	0.9549
(3, 68, 68, 22)	0.9694	0.9544	0.9543
(3, 68, 68, 22)	0.9763	0.9618	0.9629

1600nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9597	0.9476	0.9501
(3, 52, 52, 18)	0.9516	0.9377	0.9388
(3, 60, 60, 20)	0.9643	0.9511	0.9526
(3, 60, 60, 20)	0.9589	0.9447	0.9454
(3, 68, 68, 22)	0.9566	0.9363	0.9367
(3, 68, 68, 22)	0.9630	0.9482	0.9476

Table 2. We decide upon our final edge generation network after looking at eighteen different network configurations. We vary the input size among three options, consider three different cubic ROI diameters, and train both from scratch and finetuning with the node network. The finetuned results are in the light gray rows. The best validation results occur with a cubic region of $1200\,\mathrm{nm}$ in diameter and an input size of (52, 52, 18).

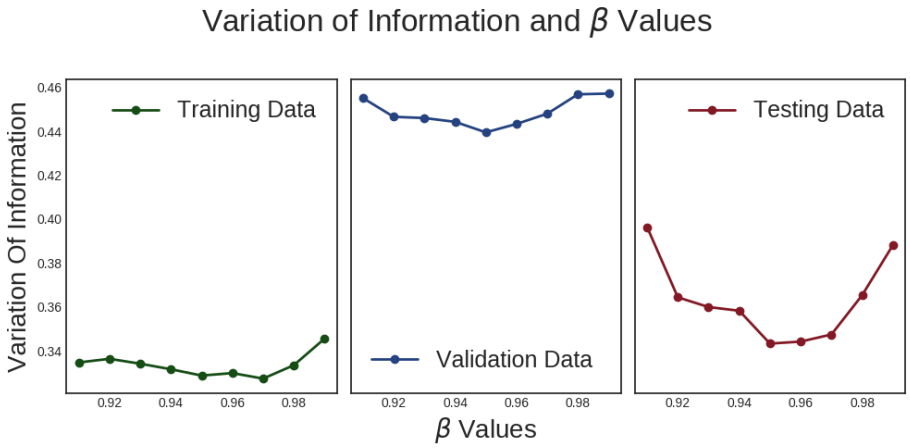

Figure 7. We consider a wide range of possible β values and see how each affects the total variation of information score. A β value of 0.95 creates the biggest reduction in variation of information on three validation datasets.

$$w_e = \log \frac{p_e}{1 - p_e} + \log \frac{1 - \beta}{\beta} \tag{1}$$

Figure 7 shows the equation as a function of p_e and β. β is a tunable parameter that encourages over- or under-segmentation. Based on the validation data we use $\beta = 0.95$ (Fig. 8). We find that this β value creates the largest reduction in variation of information on three PNI validation datasets.

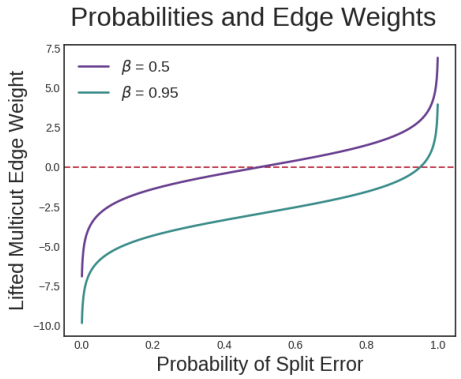

Figure 8. This function converts probabilities between 0 and 1 to edge weights. β is a tunable parameter that encourages over- or under-segmentation. Based on the results on three validation datasets we set $\beta = 0.95$.

References

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[2] K. Dmitriev, T. Parag, B. Matejek, A. Kaufman, and H. Pfister. Efficient correction for em connectomics with skeletal representation. In *British Machine Vision Conference (BMVC)*, 2018.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015.

[6] T.-C. Lee, R. L. Kashyap, and C.-N. Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[7] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[8] K. Palágyi. A sequential 3d curve-thinning algorithm based on isthmuses. In *International Symposium on Visual Computing*, pages 406–415. Springer, 2014.

[9] K. Palágyi and A. Kuba. A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, 19(7):613–627, 1998.

[10] K. Palágyi, G. Németh, and P. Kardos. Topology preserving parallel 3d thinning algorithms. In *Digital Geometry Algorithms*, pages 165–188. Springer, 2012.

[11] M. Sato, I. Bitter, M. A. Bender, A. E. Kaufman, and M. Nakajima. Teasar: Tree-structure extraction algorithm for accurate and robust skeletons. In *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, pages 281–449. IEEE, 2000.

[12] T. Zhao, D. Olbris, Y. Yu, and S. M. Plaza. Neutu: Software for collaborative, large-scale, segmentation-based connectome reconstruction. *Frontiers in Neural Circuits*, 12:101, 2018.

[13] T. Zhao and S. M. Plaza. Automatic neuron type identification by neurite localization in the drosophila medulla. *arXiv preprint arXiv:1409.1892*, 2014.