Biologically-Constrained Graphs for Global Connectomics Reconstruction

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

Abstract

Most current state-of-the-art connectome reconstruction pipelines have two major steps: initial pixel-based segmentation with affinity prediction and watershed transform, and refined segmentation by merging over-segmented regions. These methods rely only on local context and are typically agnostic to the underlying biology. Since a few merge errors can lead to several incorrectly merged neuronal processes, these algorithms are currently tuned towards over-segmentation producing an overburden of costly proofreading. We propose a third step for connectomics reconstruction pipelines to refine an over-segmentation using both local and global context with an emphasis on adhering to the underlying biology. We first extract a graph from an input segmentation where nodes correspond to segment labels and edges indicate potential split errors in the over-segmentation. To increase throughput and allow for large-scale reconstruction, we employ biologically inspired geometric constraints based on neuron morphology to reduce the number of nodes and edges. Next, two neural networks learn these neuronal shapes to aid the graph construction process further. Lastly, we reformulate the region merging problem as a graph partitioning one to leverage global context. We demonstrate the performance of our approach on four real-world connectomics datasets with an average variation of information improvement of 21.3%.

1. Introduction

By studying connectomes–wiring diagrams extracted from the brain containing every neuron and the synapses between them–neuroscientists hope to understand better certain neurological diseases, generate more faithful models of the brain, and advance artificial intelligence [12, 15]. To this end, neuroscientists produce high-resolution images of brain tissue with electron microscopes where every synapse, mitochondrion, and cell boundary is visible [19]. Since these datasets now exceed a petabyte in size, manual tracing of neurons is infeasible and automatic segmentation techniques are required.

Current state-of-the-art automatic 3D reconstruction approaches typically use pixel-based convolutional neural networks (CNNs) and watershed transforms to generate an initial over-segmentation [24, 37, 42], followed by region merging steps [11, 21, 25, 30, 35]. Flood-filling networks combine these two steps into one by gradually expanding segments from a seed voxel [18]. However, all of these above strategies make decisions using only the local context and do not consider the global ramifications to individual merges. Therefore, a small number of compounding merge errors can create an under-segmentation with several neuronal processes labeled as one neuron. Since correcting such *merge errors* is computationally challenging, current methods typically favor over-segmentation where a neuronal process is segmented into multiple labels. Unfortunately proofreading these *split errors*, while easier, still remains onerous [33].

We propose a third step for connectomics reconstruction workflows to refine these over-segmentations and close the gap between automatic and manual segmentation. We reformulate the region merging problem as a graph partitioning one to leverage global context during the agglomeration process. Thus far the computational burden associated with global optimization strategies remains their biggest drawback despite some research into parallelizing the computation [2]. Performing the graph partitioning step after an existing agglomeration technique allows us to capture larger shape context when making decisions. Furthermore, the amount of computation significantly decreases as the input method correctly segments a large number of supervoxels. The remaining *split errors* typically occur in places where a neuronal process becomes quite thin or the corresponding image data noisy—difficult locations to reconstruct using only the local context from images and affinities.

When constructing our graph, we employ geometric constraints guided by the underlying biological morphology to reduce the number of nodes and edges. Due to their biological nature, over-segmented regions should be con-

*Corresponding author, bmatejek@seas.harvard.edu

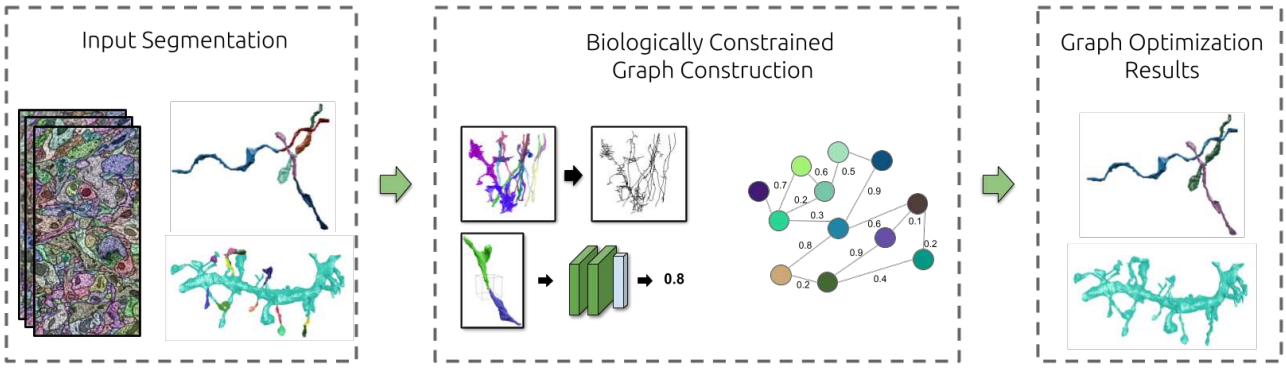

Figure 1. Most current state-of-the-art segmentation pipelines consist of affinity generation with watershed transform and region merging (left). We follow these existing methods by constructing a graph derived from their segmentation by enforcing geometric constraints inspired by the underlying biology and learning typical neuronal morphologies (center). Our graph formulation allows us to partition the graph with a global optimization strategy to produce an improved segmentation (right).

nected with specific geometric and topological properties in mind. For example, among other biological considerations, L-shaped junctions and arrow-shaped junctions are rare in neuronal structures. We can both use and learn these shape priors to produce a more accurate region merging strategy.

Our region merging framework consists of several steps to first construct a graph from an input segmentation and then to partition the graph using a global optimization strategy (Fig. 1). We first identify segments that are clearly over-segmented based on our knowledge of the span of neuronal processes and use a trained CNN to merge these segments with larger ones nearby. Remaining segments receive a node in our graph. We then generate skeletons for each segment to produce a simple yet expressive representation of the underlying shape of a given segment (Fig. 1, center). From these skeletons, we identify potential segments to merge, which in turn receive a corresponding edge in the graph. Another CNN classifier learns the local structural shapes of neurons and produces probabilities that two segments belong to the same neuron. Finally, we employ a graph optimization algorithm to partition the graph into an improved reconstruction (Fig. 1, right). Our graph formulation creates a formal description of the problem enabling a diverse range of optimization strategies in the future.

This work makes three main contributions: first, a method to extract biologically-inspired graphs from an input segmentation using hand-designed geometric constraints and machine-learned neuronal morphologies; second, a top-down framework to correct split errors in an input segmentation; last, a reduction of variation of information on state-of-the-art inputs by 21.3% on four datasets.

2. Related Work

Initial Pixel-based Segmentation Methods. There are two main approaches to segmenting electron microscopy images at the voxel-level. In the first, 2D or 3D convolutional neural networks are trained to produce an intermediate representation such as boundary [7, 16, 21, 37] or affinity maps [24, 39]. Advancements in architecture designs (e.g., 3D U-Net [6]), model averaging techniques [40], segmentation-specific loss functions (e.g., MALIS [4]), and data augmentation strategies [25] have greatly improved the results for these intermediate representations. Afterwards, clustering techniques such as watershed [8, 10, 42] or graph partition [1] transform these intermediate representations into a segmentation. In the second approach, neural networks [18, 28] are trained recursively to grow the current estimate of a binary segmentation mask, which is further extended to handle multiple neurons [29]. Despite impressive segmentation accuracies, the computational burden of this approach remains a limitation as the network needs to infer each segment separately.

Agglomeration Strategies. Agglomeration methods are parameterized by the similarity metric between adjacent segments and merging strategy. For the similarity metric, Lee *et al*. [25] and Funke *et al*. [11] rely solely on the predicted affinities and define the metric as the mean affinity between segments. Classification-based methods generate the probability to merge two segments from handcrafted [17, 21, 30, 34, 42] or learned features [3]. Niko *et al*. [23] use the information about post- and pre-synaptic connections to refine the multicut algorithm and prevent axons and dendrites from merging. For the merging strategy, most methods use variants of hierarchical agglomeration [21, 30, 34, 35, 42] to greedily merge a pair of regions at a time. Other methods formulate agglomeration as reinforcement learning [17] and superpixel partitioning problems [2]. More recently, flood-filling networks [18] use different seeding strategies with the same network from the

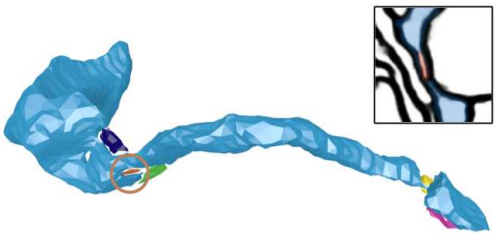

Figure 2. The above neuronal process is incorrectly segmented into several labels. Five of the segments are very small indicating that they must merge with a nearby larger segment. Frequently these small segments are artifacts of noisy affinities around locations where a process becomes quite thin.

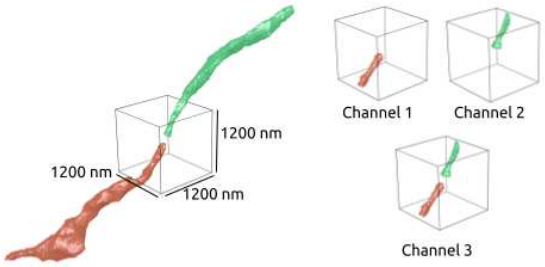

Figure 3. Both networks take three channels as input corresponding to if a particular voxel belongs to segment one, segment two, or either segment. This particular example is input to the edge CNN to determine if two segments belong to the same neuronal process.

initial segmentation step to agglomerate regions.

Error-correction Methods. Although significant advancements in the above methods produce impressive results, there are still errors in the segmentations. These errors are corrected either manually with human proofreading [14, 22] or automatically [43]. Since correcting errors is a computationally expensive task, various research explores how to use machine learning to improve human efficiency [13], automatic detection of error regions [36, 43], or reduce the search space via skeletonization [9]. However, these methods rely only on local context for decision-making and do not enforce biological constraints on their corrections.

3. Biologically-Constrained Graphs

Most current graph-based approaches assign a node to every unique label in the volume with edges between segments that have at least one neighboring pair of voxels. However, as the image volumes grow in size, the number of edges under such an approach increases dramatically. We employ hand-crafted geometric constraints based on the underlying biology to reduce the number of nodes and edges. Furthermore, we learn neuron morphologies with two neural networks to aid in the graph generation process.

3.1. Node Generation

Current pipelines that agglomerate regions based on the affinity predictions alone produce a large number of tiny segments (e.g., 86.8% of the segments produced by the waterz algorithm on a representative dataset contain fewer than $9,600$ voxels corresponding to a volume of approximately $0.01\,\mu\text{m}^3$). Since these strategies use only the mean affinity between two supervoxels, noise in the affinity generation process produces these small artifacts. In particular, these segments frequently occur in regions where a neuronal process becomes quite thin leading to low affinities between voxels (Fig. 2). We can leverage additional information about the underlying biology to identify and correct these segments: namely that neurons are quite large and should not contain few voxels when segmented. Figure 2 shows an example neuronal process over-segmented into six distinct components, five of which are relatively small. Each of these segments had sufficiently low mean affinities with its neighbors.

We identify these small segments and merge them before graph construction to reduce the number of nodes (and edges). We flag any segment whose volume is less than t_{vol} cubic microns as *small* and create a list of nearby *large* segments as potential merge candidates. The simplest method to absorb these segments is to agglomerate them with a non-flagged neighbor with the highest mean affinity. However, these segments arise because of inaccuracies in the affinities. We employ two methods to merge these nodes based on the geometry of the small segments themselves. Some agglomeration strategies produce several "singleton" segments that are completely contained within one image slice. We link these singletons together across several slices by considering the Intersection over Union when superimposing two adjacent slices. Second, we train a neural network to learn if two segments, one small and the other large, belong to the same neuron.

Looking at the local shape around two segments can provide significant additional information over just the raw image data or affinities alone. Often split errors occur at regions with either image artifacts or noisy affinities; however, the segment shapes provide additional information. We extract a small cube with diameter d_{node} nanometers around each *small–large* segment pair. We train a feed-forward 3D CNN to learn the neuron morphology and predict which pairs belong to the same neuron. The CNN takes as input three channels corresponding to if the voxel belongs to the small segment, the large segment, or either segment (Fig. 3). Our network contains three *VGG-style* convolution blocks [5] and two fully connected layers before a final sigmoid activation. The network parameters are further discussed in Sec. 4.2. Each small segment is merged with exactly one nearby large segment to prevent a merge

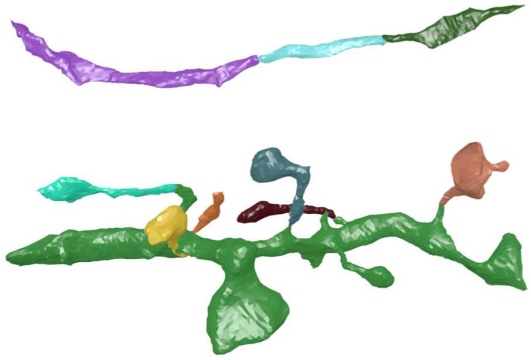

Figure 4. Two typical instances of split errors in connectomics segmentations. In the top image, the neuronal process is split multiple times at some of its thinnest locations. On the bottom, multiple spines are split from the dendrite.

error from connecting two distinct neurons completely.

3.2. Edge Generation

Each remaining segment in the volume has a large number of adjacent neighbors (28 per segment averaged over three gigavoxel datasets). We use a geometric prior on the split errors to reduce the number of considered errors greatly. Most split errors follow one of two modalities: either a neuronal process is split into two or more parts across its primary direction (Fig. 4, top) or several spines are broken off a dendrite (Fig. 4, bottom).

We generate skeletons for each segment to create a simple yet expressive representation of a volume's underlying shape. For example, this approach allows us to quickly identify all of the dendritic spines in a segment with minimal computation (Fig. 5). Some previous research focuses on the development and use of skeletons in the biomedical and connectomics domains for quicker analysis [38, 41] and error correction [9]. Topological thinning and medial axis transforms receive a significant amount of attention in the computer graphics and volume processing communities [26, 32].

We first downsample each segment using a max-pooling procedure to a resolution of $(X_{res}, Y_{res}, Z_{res})$ nanometers before generating the skeletons. This process does not cause significant detail loss since the finest morphological features of neurons are on the order of $100\,\mathrm{nm}$ [36]. In fact, the produced skeletons more closely follow the underlying geometry since the boundaries of these segments are quite noisy. We use a sequential topological thinning algorithm [31] to gradually erode the boundary voxels for each segment until only a skeleton remains. Figure 5 shows two example segments with their corresponding skeletons. The larger spheres in the skeleton correspond to endpoints. We generate a vector at each endpoint to indicate the direction of our skeleton before endpoint termination.

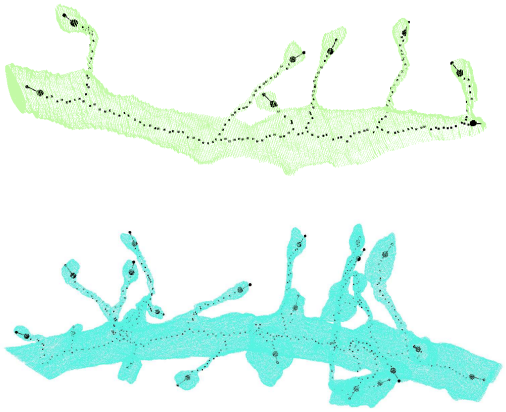

Figure 5. Two example skeletons produced by a topological thinning algorithm [31]. The larger spheres represent endpoints and the vectors protruding from them show the direction of the skeleton at endpoint termination.

When generating the edges for our graph, we exploit the aforementioned split error modalities which follow from the underlying biological structure of neurons. To identify these potential *split error* locations, we use the directional vectors at each skeleton endpoint. For each endpoint v_e in a given segment S_e we consider all voxels v_n within a defined radius of t_{edge} nanometers. If that voxel belongs to another segment S_n that is locally adjacent to S_e and the vector between v_e and v_n is within θ_{max} degrees of the directional vector leaving the skeleton endpoint, nodes S_e and S_n receive an edge in the graph. θ_{max} is set to approximately $18.5°$; this value follows from the imprecision of the endpoint vector generation strategy.

3.3. Edge Weights

To generate the merge probabilities between two segments we use a CNN similar to the one discussed in Section 3.1. We extract a small cube of diameter d_{edge} nanometers around each potential merge location found in the edge generation step. Again, we train a new feed-forward 3D CNN with three channels encoding whether a voxel belongs to each segment or either (Fig. 3). The network follows the same general architecture with three *VGG-style* convolution layers followed by two fully connected layers and a final sigmoid activation.

We next convert these probabilities into edge weights with the following weighting scheme [20]:

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

where p_e is the corresponding merge probability and β is a tunable parameter that encourages over- or under-segmentation. Note high probabilities transform into positive weights. This follows from our optimization strategy

Table 1. We show results on four testing datasets, two from the PNI volumes, one from the Kasthuri volume, and one on the SNEMI3D challenge dataset. We use four PNI volumes for training and three for validation. We further finetune our neural networks on separate training data for both the Kasthuri and SNEMI3D volumes.

Dataset	Brain Region	Sample Resolution	Dimensions	Segmentation
PNI	Primary Visual Cortex	$3.6 \times 3.6 \times 40\,\mathrm{nm}^3$	$2048 \times 2048 \times 256$	Zwatershed and Mean Agg [25]
Kasthuri	Somatosensory Cortex	$6 \times 6 \times 30\,\mathrm{nm}^3$	$1335 \times 1809 \times 338$	Waterz [11]
SNEMI3D	Somatosensory Cortex	$3 \times 3 \times 30\,\mathrm{nm}^3$	$1024 \times 1024 \times 100$	Waterz [11]

(discussed below) which minimizes an objective function and therefore should collapse all positive weighted edges.

3.4. Graph Optimization

Our graph formulation enables us to apply a diverse range of graph-based global optimization strategies. Here, we reformulate the partitioning problem as a multicut one. There are two primary benefits to this minimization strategy: first, the final number of segments depends on the input and is not predetermined; second, the solution is globally consistent (i.e., a boundary remains only if the two corresponding nodes belong to different segments) [20].

We use the greedy-additive edge contraction method to produce a feasible solution to the multicut problem [20]. Following their example, we use the more general lifted multicut formulation where all non-adjacent pairs of nodes receive a "lifted" edge and a corresponding edge weight indicating the long-range probability that two nodes belong to the same neuron. Ideally, these weights perfectly reflect the probability that two nodes belong to the same neuron by considering all possible paths between the nodes in the graph. Unfortunately, such computation is expensive, so we create a lower estimate of the probability by finding the shortest path on the negative log-likelihood graph (i.e., each original edge weight w_e is now $-\log w_e$) and setting the probability equal to e raised to the distance [20].

4. Experiments

We discuss the datasets used for evaluation and the various parameters from the previous section.

4.1. Datasets

We evaluate our methods using four datasets with different resolutions, acquisition techniques, and input segmentation strategies (Table 3.2). The PNI volumes were given to us by the authors of [43] and contain nine separate volumes imaged by a serial section transmission electron microscope (ssTEM). We use four of these volumes to train our networks and tune parameters, three for validation, and the last two for testing. These image volumes have an initial segmentation produced by a variant of a 3D U-Net followed by zwatershed and mean agglomeration [25].

The Kasthuri dataset is freely available online[1] and represents a region of the neocortex imaged by a scanning electron microscope (SEM). We divide this volume into training and testing blocks. We initially use a 3D U-Net to produce affinities and agglomerate with the waterz algorithm [11].

Although our proposed method is designed primarily for large-scale connectomics datasets, we evaluate our method on the popular SNEMI3D challenge dataset.[2] Our initial segmentation strategy is the same for both the SNEMI3D and Kasthuri datasets.

4.2. Parameter Configuration

Here we provide the parameters and CNN architectures discussed in Section 3. The supplemental material provides additional experiments that explore each of these parameters and network architectures in further detail.

Node Generation. To determine a suitable value for t_{vol}—the threshold to receive a node in the graph—we consider the edge generation step which requires expressive skeletons. Skeletons generated through gradual boundary erosion [31] tend to reduce small segments to a singular point removing all relevant shape information. After exploring various threshold values on four training datasets we set $t_{vol} = 0.010\,36\,\mu\mathrm{m}^3$.

Skeletonization Method. To evaluate various skeleton generation approaches we create and publish a skeleton benchmark dataset.[3] We evaluate three different skeleton approaches with varying parameters on this benchmark dataset [26, 31, 38]. Downsampling the data to 80 nanometers in each dimension followed by a topological thinning algorithm [31] produces the best results.

Edge Generation. During edge generation, we want to minimize the total number of edges while maintaining a high recall on the edges corresponding to *split errors*. After considering various thresholds, we find that $t_{edge} = 500\,\mathrm{nm}$ guarantees both of these attributes. When transforming our probabilities into edge weights, we use $\beta = 0.95$ to reduce the number of false merges further.

CNN Training. Of the nine PNI datasets, we use four for training and three for validation. We experimented with various network architectures and input cube sizes. Our node network receives a cube with $d_{node} = 800\,\mathrm{nm}$ which is then sampled into a voxel grid of size $(60, 60, 20)$. Our edge

[1] https://neurodata.io/data/kasthuri15/

[2] http://brainiac2.mit.edu/SNEMI3D/home

[3] http://rhoana.org/skeletonbenchmark

Table 2. Our proposed method reduces the total variation of information by 20.9%, 28.7%, 15.6%, and 19.8% on four testing datasets. The variation of information split decreases significantly, achieving a maximum reduction of 45.5% on the second PNI testing dataset.

Dataset	Total VI			VI Split		VI Merge	
	Baseline	Proposed	Decrease	Baseline	Proposed	Baseline	Proposed
PNI Test One	0.491	0.388	-20.9%	0.418	0.273	0.073	0.115
PNI Test Two	0.416	0.297	-28.7%	0.368	0.200	0.049	0.097
Kasthuri Test	0.965	0.815	-15.6%	0.894	0.681	0.071	0.134
SNEMI3D	0.807	0.647	-19.8%	0.571	0.438	0.236	0.209

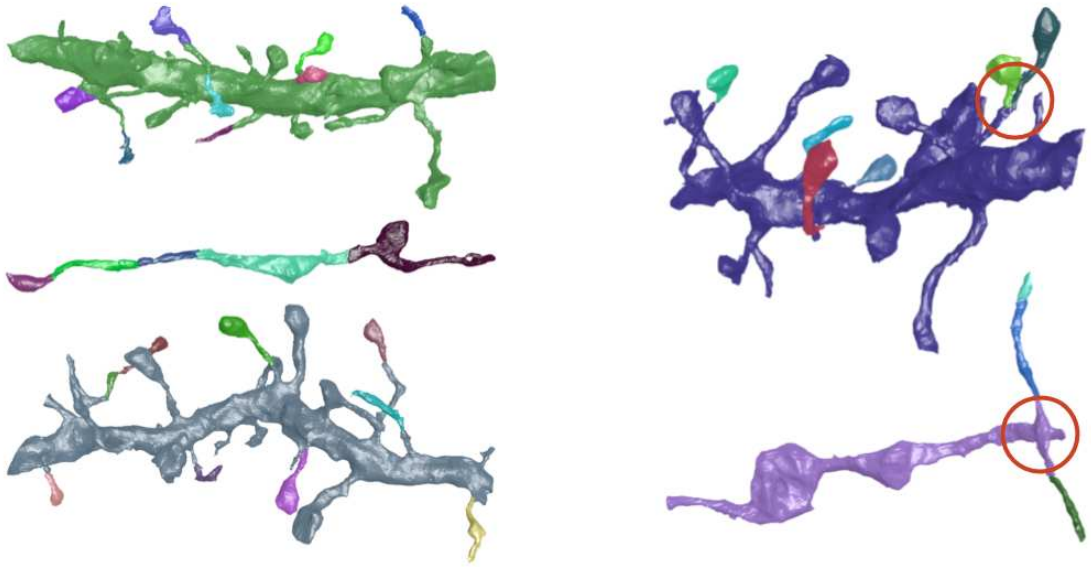

Figure 6. Here we show three success (left) and two failure (right) cases for our proposed methods. On the left, we see two dendrites with eight spines each correctly merged. Correcting these types of splits errors is particularly essential for extracting the wiring diagram since synaptic connections occur on the spines. In between these examples, we show a typical neuronal process initially split at numerous thin locations. Circled on the top right is an incorrectly merged spine to the dendrite. We correctly connect five spines but we accidentally merge two spines to the same location once. Below that is an example where a merge error in the input segmentation causes an error.

network receives a cube with $d_{edge} = 1200\,\mathrm{nm}$ which is similarly sampled into a voxel grid of size $(52, 52, 18)$

We train each network on the PNI data for 2,000 epochs. There are 20,000 examples per epoch with an equal representation of ones that should and should not merge. We employ extensive data augmentation by randomly rotating the input around the z-axis and reflecting over the xy-plane. For the Kasthuri and SNEMI3D data, we finetune the pre-trained network for 500 epochs.

4.3. Error Metrics

We evaluate the performance of the different methods using the split variation of information (VI) [27]. The split and merge variation of information scores quantify over- and under-segmentation respectively using the conditional entropy. The sum of the two entropies gives the total variation of information. For our CNNs, a true positive indicates a corrected split error and a false positive a merge error introduction.

5. Results

We provide quantitative and qualitative analysis of our method and ablation studies comparing the effectiveness of each component.

5.1. Benchmark Comparison

Table 2 shows the total variation of information improvement of our method over our input segmentations on four test datasets. We reduce the total variation of information on the two PNI, Kasthuri, and SNEMI3D datasets by 20.9%, 28.7%, 15.6%, and 19.8% respectively. Our VI split scores decrease by 34.5%, 45.5%, 23.8%, and 23.3% on the four datasets. Our proposed method only merges segments together and does not divide any into multiple components, and thus our VI merge scores can only increase. However, our input segmentations are very over-segmented and have a small VI merge score at the start. Our algorithm increases the VI merges (i.e., it makes some wrong merge decisions) but the overall decrease in VI split overcomes the slight in-

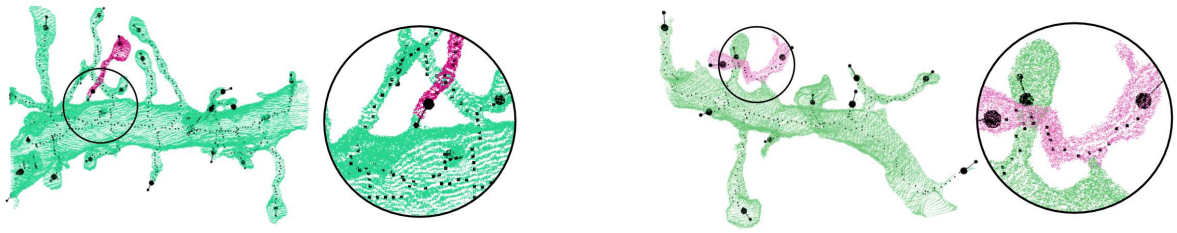

Figure 7. One success (left) and one failure (right) of our proposed biologically-constrained edge generation strategy. In the left instance, the broken spine has a skeleton endpoint with a vector directed at the main process. In the right example, two spines are split from the dendrite but merged together in the input segmentation. The skeleton traverses near the broken location without producing an endpoint.

creases in VI merge. On the SNEMI3D dataset, we generate multiple baselines and proposed segmentations by varying the merging threshold in the waterz algorithm. We show the results on the best baseline compared to the best-corrected segmentation, and thus the VI merge can decrease for this dataset.

Figure 6 shows five examples from our proposed method, three correct (left) and two failures (right). Here, we see two example dendrites with eight spines each correctly reconnected to the neuronal process. Fixing these types of split errors is crucial for extracting the wiring diagram from the brain: electrical signal from neighboring cells is propagated onwards through post-synaptic densities located on these spines. Between these two dendrites, we show a typical neuronal process split into multiple segments at locations where the process becomes quite thin. Our edge generation step quickly identifies these locations as potential split errors, and our CNN predicts that the neuronal process is continuing and not terminating. On the top right, we show an example dendrite where we correctly merge five spines. However, in one location (circled) we accidentally merge one additional spine causing a merge error. Below that, we show an error caused by a merge error in the input segmentation. The purple neuronal process is incorrectly merged at one location with a perpendicular traversing process (circled). We merge other segments with the perpendicular process causing an increase in VI merge.

5.2. Empirical Ablation Studies

Here, we elaborate on the effectiveness of each component of our method on three of the datasets and compare against relevant baselines.

Node Generation. Table 3 summarizes the success of our node generation strategy in terms of correctly merging small segments to larger ones from the same process. We compare our results against the following simple baseline: how many small labels are correctly merged if they receive the same label as the adjacent large segment with which it shares the highest mean affinity. Our method significantly outperforms the baseline on the PNI datasets. The baseline performs poorly as expected since the input segmentation agglomeration strategy initially opted not to merge these small segments based on the affinities alone. In each case, we correctly merge between 76 and 85% of small segments. The waterz agglomeration strategy produces many more small segments than the mean agglomeration method. Interestingly, the baseline is much higher for this strategy, indicating that a simple post-processing method of merging small segments based on a thresholded affinity might be justified.

Table 3. Our proposed node generation strategy that merges small segments into nearby larger ones outperforms the baseline strategy. In our best instance, we correctly merge 444 small segments while only incorrect merging 75.

Dataset	Baseline	Proposed
PNI Test One	305 / 521 (36.9%)	686 / 169 (80.2%)
PNI Test Two	185 / 281 (39.7%)	444 / 75 (85.5%)
Kasthuri Test	4,514 / 4,090 (52.5%)	6,623 / 2,020 (76.6%)

Edge Generation. There are two main components to edge generation: skeletonization and location of potential split errors. We created a skeleton benchmark dataset for connectomics segmentations and labeled the endpoints for 500 ground truth segments. The utilized skeletonization approach has a precision of 94.7% and a recall of 86.7% for an overall F-score of 90.5% on the benchmark dataset.

Figure 7 shows some qualitative examples of where our method succeeds (left) and fails (right). Our method correctly establishes edges whenever one of the neuronal processes has a skeleton endpoint and directional vector in the vicinity of the error (left). In this particular example, the broken spine has an endpoint vector pointing directly at the corresponding dendrite. On the right, we see a failure where two spines are connected to one another causing the skeleton to have no endpoints at the break.

Table 4 provides the quantitative results for our edge generation method. The simple baseline strategy is to use the adjacency graph from the segmentation. That is, two nodes receive an edge if the corresponding segments have a pair of neighboring voxels. We notice that the adjacency graph creates a large number of edges between neuronal processes that should not merge. In contrast, our proposed method reduces the graph size by around 60% on each of

Table 4. Our edge generation strategy reduces the number of edges in the graph by around 60% on each of the three datasets. Impressively 80% of the true split errors remain after the edge pruning operations.

Dataset	Baseline	Proposed	Edge Recall
PNI Test One	528 / 25,619	417 / 10,074	79.0% / 39.3%
PNI Test Two	460 / 30,388	370 / 11,869	80.4% / 39.1%
Kasthuri Test	1,193 / 43,951	936 / 18,168	78.5% / 41.3%

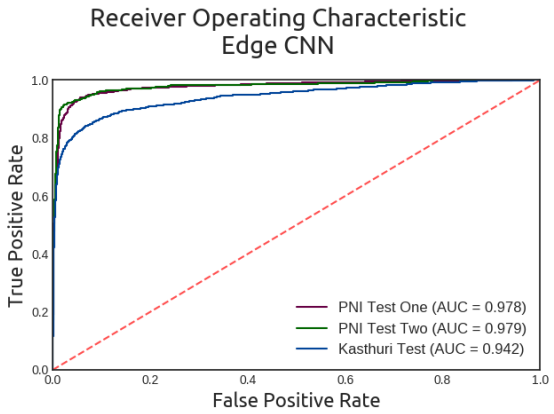

Figure 8. The receiver operating characteristic (ROC) curve for our learned edge features for three test datasets.

Table 5. Using a global graph optimization strategy prevents segments from merging incorrectly over a traditional greedy approach. Our average decrease in VI merge over the baseline is 15.1% with a maximum decrease of 23.6%.

Dataset	Baseline	Proposed	Decrease
PNI Test One	0.127	0.115	-9.4%
PNI Test Two	0.127	0.097	-23.6%
Kasthuri Test	0.153	0.134	-12.4%

the three datasets. Similarly, our recall of true split errors is around 80% on each dataset.

We provide the results of our edge CNN in Figure 8. Overall our network performs well on each of our datasets with accuracies of 96.4%, 97.2%, and 93.4% on the PNI and Kasthuri datasets respectively.

Graph Partitioning. Lastly, we quantify the benefits to using a global graph partitioning strategy over a standard agglomeration technique. As a baseline, we merge regions together using only the local context from our CNN classifier. To create a fair comparison with our proposed method, we merge all segments whose predicted merge scores exceed 95% (a corollary to the chosen β value). Table 5 shows the improvement in variation of information merge over a greedy agglomeration approach. The VI merge score decreases by 15.1% on average when using a global optimization strategy.

5.3. Computational Performance

All performance experiments ran on an Intel Core i7-6800K CPU 3.40 GHz with a Titan X Pascal GPU. All code is written in Python and is freely available[4]. We use the Keras deep learning library for our neural networks with Theano backend and cuDNN 7 acceleration for CUDA 8.0. Table 6 shows the running time for each step of our proposed method on the PNI Test Two dataset ($2048 \times 2048 \times 256$). Our method achieves a throughput of 1.66 megavoxels per second.

Table 6. Running times on a gigavoxel dataset.

Step	Running Time
Node Feature Extraction	73 seconds
Node CNN	208 seconds
Skeleton Generation	34 seconds
Edge Feature Extraction	208 seconds
Edge CNN	109 seconds
Lifted Multicut	13 seconds
Total	10.75 minutes

6. Conclusions

We propose a third step for connectomics reconstruction workflows to refine over-segmentations produced by typical state-of-the-art reconstruction pipelines. Our method uses both local and global context to improve on the input segmentation using a global graph optimization strategy. For local context, we employ geometric constraints based on the underlying biology and learn typical neuron morphologies. Performing the graph optimization after initial segmentation allows us to capture larger shape context when making decisions. We improve on state-of-the-art segmentation methods on four different datasets, reducing the variation of information by 21.3% on average.

Our graph formulation provides a formal description of the problem and enables a wide range of optimization strategies in the future. Our current implementation makes use of the lifted multicut formulation. However, our method can easily be extended to a wide range of other graph partitioning strategies. For example, with progress in automatic identification of neuron type (e.g., excitatory or inhibitory) we can introduce additional constraints to the global optimizer to prevent different types from merging.

Acknowledgements. We thank Jonathan Zung and Sebastian Seung from the Princeton Neuroscience Institute for sharing their results and data with us. We also thank Kálmán Palágyi for providing us with topological thinning code. This research was supported in part by NSF grants IIS-1447344 and IIS-1607800, and by IARPA contract D16PC00002.

[4]http://rhoana.org/biologicalgraphs

References

[1] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, pages 778–791. Springer, 2012.

[2] T. Beier, C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature methods*, 14(2):101, 2017.

[3] J. A. Bogovic, G. B. Huang, and V. Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013.

[4] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems*, pages 1865–1873, 2009.

[5] 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.

[6] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[7] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[8] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[9] 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.

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 2004.

[11] J. Funke, F. D. Tschopp, W. Grisaitis, C. Singh, S. Saalfeld, and S. C. Turaga. A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs. *arXiv preprint arXiv:1709.02974*, 2017.

[12] D. Haehn, J. Hoffer, B. Matejek, A. Suissa-Peleg, A. K. Al-Awami, L. Kamentsky, F. Gonda, E. Meng, W. Zhang, R. Schalek, et al. Scalable interactive visualization for connectomics. In *Informatics*, volume 4, page 29. Multidisciplinary Digital Publishing Institute, 2017.

[13] D. Haehn, V. Kaynig, J. Tompkin, J. W. Lichtman, and H. Pfister. Guided proofreading of automatic segmentations for connectomics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. W. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2466–2475, 2014.

[15] M. Helmstaedter. The mutual inspirations of machine learning and neuroscience. *Neuron*, 86(1):25–28, 2015.

[16] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstädter, K. Briggman, W. Denk, J. Bowden, J. Mendenhall, W. Abraham, K. Harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and S. Seung. Boundary learning by optimization with topological constraints. In *Proc. IEEE CVPR 2010*, pages 2488–2495, 2010.

[17] V. Jain, S. C. Turaga, K. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung. Learning to agglomerate superpixel hierarchies. In *Advances in Neural Information Processing Systems*, pages 648–656, 2011.

[18] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605, 2018.

[19] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

[20] 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.

[21] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister. Rhoananet pipeline: Dense automatic neural annotation. *arXiv preprint arXiv:1611.06973*, 2016.

[22] S. Knowles-Barley, M. Roberts, N. Kasthuri, D. Lee, H. Pfister, and J. W. Lichtman. Mojo 2.0: Connectome annotation tool. *Frontiers in Neuroinformatics*, (60), 2013.

[23] N. Krasowski, T. Beier, G. Knott, U. Köthe, F. A. Hamprecht, and A. Kreshuk. Neuron segmentation with high-level biological priors. *IEEE transactions on medical imaging*, 37(4):829–839, 2018.

[24] K. Lee, A. Zlateski, V. Ashwin, and H. S. Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary prediction. In *Advances in Neural Information Processing Systems*, pages 3573–3581, 2015.

[25] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[26] 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.

[27] M. Meila. Comparing clusterings by the variation of information. In *Colt*, volume 3, pages 173–187. Springer, 2003.

[28] Y. Meirovitch, A. Matveev, H. Saribekyan, D. Budden, D. Rolnick, G. Odor, S. Knowles-Barley, T. R. Jones, H. Pfister, J. W. Lichtman, et al. A multi-pass approach to large-scale connectomics. *arXiv preprint arXiv:1612.02120*, 2016.

[29] Y. Meirovitch, L. Mi, H. Saribekyan, A. Matveev, D. Rolnick, and N. Shavit. Cross-classification clustering: An efficient multi-object tracking technique for 3-d instance segmentation in connectomics. June 2019.

[30] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

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

[32] K. Palágyi, E. Balogh, A. Kuba, C. Halmai, B. Erdőhelyi, E. Sorantin, and K. Hausegger. A sequential 3d thinning algorithm and its medical applications. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 409–415. Springer, 2001.

[33] T. Parag. What properties are desirable from an electron microscopy segmentation algorithm. *arXiv preprint arXiv:1503.05430*, 2015.

[34] T. Parag, A. Chakraborty, S. Plaza, and L. Scheffer. A context-aware delayed agglomeration framework for electron microscopy segmentation. *PLOS ONE*, 10(5):1–19, 05 2015.

[35] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga, X. Zhang, B. Matejek, L. Kamentsky, J. W. Lichtman, and H. Pfister. Anisotropic em segmentation by 3d affinity learning and agglomeration. *arXiv preprint arXiv:1707.08935*, 2017.

[36] D. Rolnick, Y. Meirovitch, T. Parag, H. Pfister, V. Jain, J. W. Lichtman, E. S. Boyden, and N. Shavit. Morphological error detection in 3d segmentations. *arXiv preprint arXiv:1705.10882*, 2017.

[37] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[38] 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.

[39] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural computation*, 22(2):511–538, 2010.

[40] T. Zeng, B. Wu, and S. Ji. Deepem3d: approaching human-level performance on 3d anisotropic em image segmentation. *Bioinformatics*, 33(16):2555–2562, 2017.

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

[42] A. Zlateski and H. S. Seung. Image segmentation by size-dependent single linkage clustering of a watershed basin graph. *arXiv preprint arXiv:1505.00249*, 2015.

[43] J. Zung, I. Tartavull, and H. S. Seung. An error detection and correction framework for connectomics. *CoRR*, abs/1708.02599, 2017.