Guided Proofreading of Automatic Segmentations for Connectomics

Anonymous CVPR submission

Paper ID 0947

Abstract

Automatic cell image segmentation methods in connectomics produce merge and split errors, which require correction through proofreading. Previous research has identified the visual search for these errors as the bottleneck in interactive proofreading. To aid error correction, we develop two classifiers to recommend candidate merge and split errors and their corrections to the user. These classifiers are informed by training a convolutional neural network with known errors in automatic segmentations against expert-labeled ground truth. Our classifiers detect potentially-erroneous regions by considering a large context region around a segmentation boundary. Corrections can then be performed as yes/no decisions resulting in faster correction times than previous methods. We evaluate our approach on connectomics datasets of different species and compare correction performance of novice and expert users against different existing systems. We report significant improvements compared to pure automatic and pure manual proofreading.

1. Introduction

In connectomics, neuroscientists annotate neurons and their connectivity within 3D volumes to gain insight into the functional structure of the brain. Rapid progress in automatic sample preparation and electron microscopy (EM) acquisition techniques has made it possible to image large volumes of brain tissue at nanometer resolution. With a voxel size of $4 \times 4 \times 40$ nm^3, a 1 mm^3 volume is 1 petabyte of data. With so much data, manual annotation is not feasible, and automatic annotation methods are needed [10, 19, 22, 15].

Automatic annotation by segmentation and classification of brain tissue is challenging [1] and all available methods make errors. This leads to the results being *proofread* by humans. This crucial task serves two purposes: 1) to correct errors in the segmentation, and 2) to increase the body of labeled data from which to train better automatic segmentation methods. Recent proofreading tools provide intuitive user interfaces to browse segmentation data in 2D and 3D and to identify and manually correct errors [28, 11, 17, 8]. Many kinds of errors exist, such as inaccurate boundaries, but the most common are *split errors*, where a single segment is labeled as two, and *merge errors*, where two segments are labeled as one (Fig. 1). With user interaction, split errors can be joined, and the missing boundary in a merge error can be defined with manually-seeded watersheds [8]. However, even with semi-automatic correction tools, the visual inspection to find errors takes the majority of the time [24].

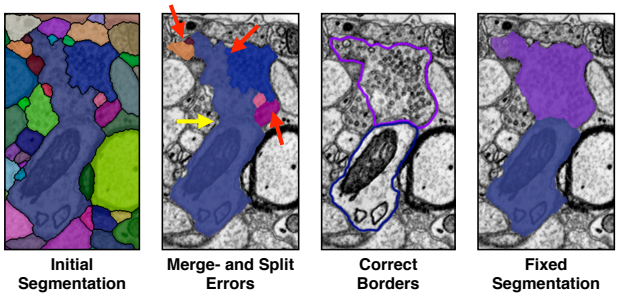

Figure 1. The most common proofreading corrections are fixing split errors (red arrows) and merge errors (yellow arrow). A fixed segmentation matches the cell borders.

Our goal is to automatically detect split and merge errors to reduce visual inspection time. Further, we wish to propose corrections to the user to accept or reject, to reduce correction time. We name this process *guided proofreading*.

First, we learn a classifier for split errors with a convolutional neural network (CNN). This takes as input patches of membrane segmentation probabilities, cell segmentation masks, and boundary masks, and outputs a probabiltiy score. As we must process large data, this classifier only operates on cell boundaries, which reduces computation over methods which analyze every pixel. For merge errors, we invert and reuse the split classification network, instead asking it to rate a set of generated candidate boundaries which hypothesize a split. We compute corrections for both types of errors.

Second, we propose a greedy algorithm for guided proofreading. Possible erroneous regions are sorted by their score, and a correction is generated for each region. Then, a user works through this list of regions and corrections. In a forced

choice setting, the user either selects a correction or skips it to advance to the next region. In an automatic setting, errors with a high probability can be automatically corrected first, given an appropriate probability threshold, after which the user would take over. Finally, to test the limits of performance, we create an oracle that only accepts corrections improving the segmentation based on knowledge of the ground truth. This equals perfect proofreading.

Third, we evaluate these methods on multiple connectomics datasets. For the forced choice setting, we perform a quantitative user study with 20 novice users who have no previous experience of proofreading EM data. We ask participants to proofread a small segmentation volume in a fixed time frame. In a between-subjects design, we compare guided proofreading to the semi-automatic *focused proofreading* approach by Plaza [25]. In addition, we compare against the manual interactive proofreading tool *Dojo* by Haehn *et al*. [8]. We also asked four domain experts to use guided proofreading and focused proofreading for comparison.

We state our contributions:

- A CNN-based boundary classifier for split errors, plus a merge error classifier which inverts the split error classifier. This is used to propose merge error corrections, removing the need to manually draw the missing edge. These classifiers perform well without much training data, which is expensive to collect for connectomics data.

- A guided proofreading approach to correcting segmentation volumes, and an assessment scenario comparing forced-choice interaction with automatic and oracle proofreading.

- A quantitative user study assessing guided proofreading, which shows that our method is able to reduce segmentation error faster than state-of-the-art semi-automatic tools, across both novice and expert users.

Our method applies to all existing automatic segmentation methods which produce a label map. As such, we believe that guided proofreading is a promising direction to proofread segmentations more efficiently to help better tackle large volumes of connectomics imagery.

2. Related Work

Automatic Segmentation. Multi-terabyte EM brain volumes require automatic segmentation [10, 19, 21, 22], but can be hard to classify due to ambiguous intercellular space: the 2013 IEEE ISBI neurites 3D segmentation challenge [1] showed that existing algorithms which learn from expert-segmented training data still exhibit high error rates.

Many works tackle this problem. NeuroProof [2] decreases error rates by learning a agglomeration on over-segmentations of images, based on a random forest classifier. Vazquez-Reina *et al*. [30] consider whole EM volumes rather than a per section approach, then solve a fusion problem with a global context. Kaynig *et al*. [14] propose a random forest classifier coupled with an anisotropic smoothing prior in a conditional random field framework with 3D segment fusion. Bogovic *et al*. [5] learn 3D features unsupervised, and show that they can be better than by-hand designs.

It is also possible to learn segmentation classification features directly from images with CNNs. Ronneberger *et al*. [26] use a contracting/expanding CNN path architecture to enable precise boundary localization with small amounts of training data. Lee *et al*. [18] recursively train very deep networks with 2D and 3D filters to detect boundaries.

All these approaches make good progress; however, in general, proofreading is still required to correct errors.

Interactive Proofreading. While proofreading is very time consuming, it is fairly easy for humans to perform corrections through splitting and merging segments. One expert tool is Raveler, introduced by Chklovskii *et al*. [6, 11]. Raveler is used today by professional proofreaders, and it offers many parameters for tweaking the process. Similar systems exist as products or plugins to visualization systems, *e.g*. V3D [24] or AVIZO [28].

Recent papers have attacked the problem of proofreading massive datasets through crowdsourcing with novices [27, 4, 7]. One popular platform is EyeWire, by Kim *et al*. [16], where participants earn virtual rewards for merging oversegmented labeling to reconstruct retina cells.

Between expert systems and online games sit Mojo and *Dojo*, by Haehn *et al*. [8, 3], which use simple scribble interfaces for error correction. Dojo extends this to distributed proofreading via a minimalistic web-based user interface. The authors define requirements for general proofreading tools, and then evaluate the accuracy and speed of Raveler, Mojo, and Dojo through a quantitative user study (Sec. 3 and 4) [8]. Dojo had the highest performance. In this paper, we use Dojo as a baseline for interactive proofreading, and so we extend the Haehn *et al*. experiment.

All interactive proofreading solutions require the user to find potential errors manually, which takes the majority of the time [24, 8]. Recent works propose computer-aided proofreading systems which help reduce the time spent in this visual search task.

Computer-aided Proofreading. Uzunbas *et al*. showed that potential labeling errors can be found by considering the merge tree of an automatic segmentation method [29]. The authors track uncertainty throughout the automatic labeling by training a conditional random field. This segmentation

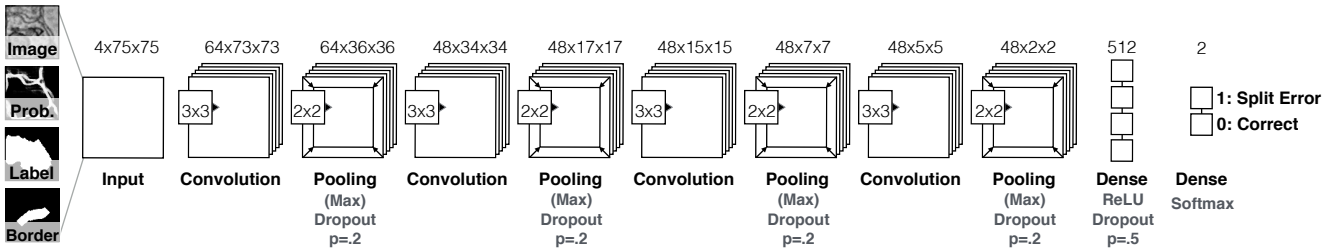

Figure 2. We build the guided proofreading classifiers using a traditional CNN architecture. The network is based on four convolutional layers, each followed by max pooling as well as dropout regularization. The 4-channel input patches are rated as either correct splits or as split errors.

technique produces uncertainty estimates, which inform potential regions for proofreading to the user. While this applies to isotropic volumes, more work is needed to apply it to anisotropic volumes, like most connectomics datasets.

Karimov *et al*. propose guided volume editing [12], which measures the difference in histogram distributions in image data to find potential split and merge errors in the corresponding segmentation. This lets expert users correct labeled computer-tomography datasets, using several interactions per correction. To correct merge errors, the authors create a large number of superpixels within a single segment and then successively group them based on dissimilarities. We were inspired by this approach but generate single watershed boundaries to handle the intracellular variance in high-resolution EM images (Sec. 3).

Most closely related to our approach is the work of Plaza, who proposed *focused proofreading* [25]. This method generates affinity scores by analyzing a region adjacency graph across slices, then finds the largest affinities based on a defined impact score. This yields edges of potential split errors which can be presented to the proofreader. Plaza reports that additional manual work is required to find and correct merge errors. Focused proofreading builds upon NeuroProof [2] as its agglomerator, and is open source with integration into Raveler. As the closest related work, we wish to use this method as a baseline to evaluate our approach (Sec. 4). However, as Haehn *et al*. showed that Raveler is less performant than Dojo for novice users, we separate the backend affinity score calculation from the expert-level front end, and present our own interface (Sec. 4).

3. Method

3.1. Split Error Detection

We build a split error classifier with output p using a convolutional neural network (CNN) to check whether an edge within an existing automatic segmentation is valid ($p = 0$) or not ($p = 1$). Rather than analyzing every input pixel, the classifier operates only on segment boundaries which requires less pixel context and is faster. In contrast to Bogovic *et al*. [5], we work with 2D slices rather than 3D volumes. This enables proofreading prior or in parallel to a computationally expensive alignment of individual EM images.

Convolutional Neural Network Architecture. Split error detection of a given boundary is really a binary classification task since the boundary is either correct or erroneous. However, in reality the score p is between 0 and 1. The classification complexity arises from hundreds of different cell types in connectomics data rather than from the classification decision itself. Intuitively, this yields a wider architecture with more filters rather than a deeper architecture with more layers. We explored different architectural configurations - including residual networks [9] - by performing a brute force parameter search and comparing precision and recall (see supplementary materials). Our final CNN configuration for split error detection is composed of four convolutional layers, each followed by max pooling as well as dropout regularization to prevent overfitting due to limited training data. Fig. 2 shows the CNN architecture for split error detection.

Classifier Inputs. To train the CNN for split error detection, we take boundary context information into consideration for the decision making process. For this, we use a 75×75 pixel patch at the center of an existing boundary. This covers approximately 80% of all boundaries in real-world connectomics data with nanometer resolution. If the boundary length is not fully covered, we sample up to 10 non-overlapping patches along the boundary and combine the resulting score by weighted averaging based on boundary length coverage per patch. Similar to Bogovich *et al*. [5], we use grayscale image data, corresponding boundary probabilities, and a single binary mask combining the two neighboring labels as features for our CNN. However, we observed that the boundary probability information generated from EM images is often misleading due to noise or artifacts in the data. This can result in merge errors within the automatic segmentation. To better direct our classifier to train on the true boundary, we extract the border between two segments. We then dilate this border

by 5 pixels to consider slight edge ambiguities and use this binary mask as an additional feature to create a stacked 4-channel input patch. Fig. 3 shows examples of correct and erroneous feature patches and their corresponding automatic segmentation and ground truth.

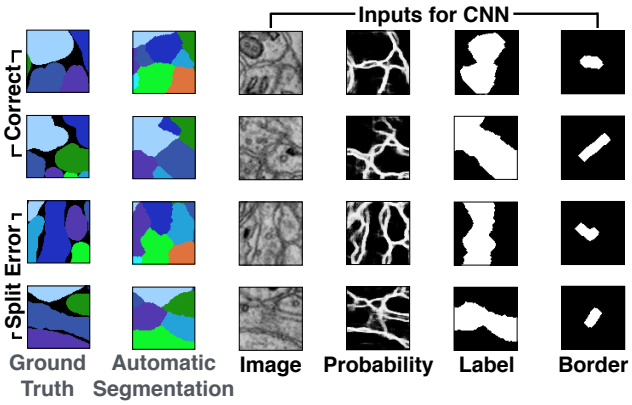

Figure 3. Example inputs for learning correct splits and split errors as reflected in the segmentation relative to the ground truth. Image, membrane probabilities, merged binary labels, and a dilated border mask are combined to 4-channel input patches.

3.2. Merge Error Detection

Identification and correction of merge errors is more challenging than finding and fixing split errors, because we must look inside segmentation regions for missing or incomplete boundaries and then propose the correct boundary. However, we can reuse the same trained CNN for this task. Similar to guided volume editing by Karimov *et al*. [12] we generate potential borders within a segment. For each segmentation label, we dilate the label by 20 pixel and generate 50 potential boundaries through the region by randomly placing watershed seed points at opposite sides of the label boundary. For watershed, we use the inverted grayscale EM image as features. This yields 50 candidate splits.

Dilation of the segment prior to watershed is motivated by our observation that the generated split tends to attach to the real membrane boundary. These boundaries are then individually rated using our split error classifier. For this, we invert the probability score such that a correct split (previously encoded as $p = 0$) is most likely a candidate for a merge error (now encoded as $p = 1$). In other words, if a generated boundary is ranked as correct, it probably should be in the segmentation. Fig. 4 illustrates this procedure.

3.3. Error Correction

We use the proposed classifiers in combination to perform corrections of split and merge errors in automatic segmentations. For this, we first perform merge error detection for all existing segments in a dataset and

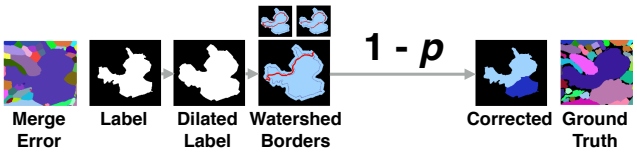

Figure 4. Merge errors are identified by generating randomly seeded watershed borders within a dilated label segment. These borders then are individually rated using the split error CNN by inverting the probability score. This way, a confident rating for a correct split most likely indicates the missing border of the merge error and can be used for correcting the labeling.

store the inverted rankings $1 - p$ as well as potential corrections. After that, we perform split error detection and store the ranking p for all neighboring segments in the segmentation. We then sort the merge and split error rankings individually from highest to lowest. For error correction, we first loop through the potential merge error regions and then through the potential split error regions. During this process, each error is now subject to a yes/no decision which can be provided in different ways:

Selection oracle. If ground truth data is available, the selection oracle *knows* whether a possible correction improves an automatic segmentation. This is realized by simply comparing the outcome of a correction using a defined measure. The oracle only accepts corrections which improve the automatic segmentation - others get discarded. This equals perfect proofreading.

Automatic selection with threshold. The decision whether to accept or reject a potential correction is done by comparing rankings to a threshold p_t. If the inverted score $1 - p$ of a merge error is higher than a threshold $1 - p_t$, the correction is accepted. Similarly, a correction is accepted for a split error if the ranking p is higher than p_t. Our experiments have shown that the threshold p_t is the same for merge and split errors which makes sense for a balanced classifier, trained on equal numbers of correct and error patches.

Forced choice setting. A user is presented with the choices of either accepting or rejecting a correction. This way, all potential split errors are seen. Inspecting all merge errors is not possible for users due to the sheer amount of generated borders. We therefore only present merge errors which satisfy $1 - p_t$.

In all cases, a decision has to be made to advance to the next possible erroneous region. If a merge error correction was accepted, the newly found boundary is added to the segmentation data. This partially updates

the merge error and split error ranking with respect to the new segment. If a split error correction was accepted, two segments are merged in the segmentation data and the disappearing segment is removed from all error rankings. We then perform merge error detection on the now larger segment and update the ranking. We also update the split error rankings to include all new neighbors and re-sort. The error with the next highest ranking is now subject to the yes/no decision.

3.4. User Interface

Guided proofreading is integrated into an existing workflow for large connectomics data. The system is web-based and is designed with a minimalistic user interface showing three components. We show the outline of the current labeling of a cell boundary and its proposed correction overlaying the EM image data. For the user, it is not possible to distinguish the current labeling and the proposed correction to avoid selection bias. We also show a solid overlay of the current and the proposed labeling. In addition, we show the image without overlays to provide an unoccluded view. User interaction is simple and involves one mouse click on either the current labeling or the correction. After interaction, the next potential error is shown. Figure 5 shows the user interface.

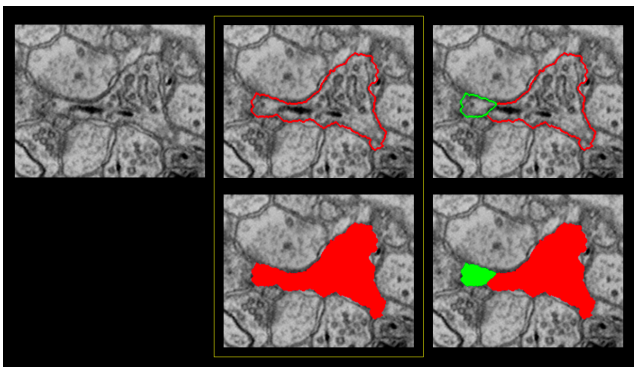

Figure 5. Segmentation correction with the guided proofreading user interface. An image without overlays is shown on the left. A split error (right) and its correction (center) are the possible choices for the user. Hovering highlights the current selection with a yellow border, and a mouse click confirms it by advancing to the next potential error.

4. Evaluation

We evaluate guided proofreading on multiple different real-world connectomics datasets of different species. All datasets were acquired using either via serial section electron microscopy (ssEM) or serial section transmission electron microscopy (ssTEM). We perform experiments for error correction based on a selection oracle, automatic selection with threshold, and the forced choice setting. The latter is designed as a between-subjects user study with novice and expert participants.

4.1. Datasets

L. Cylinder. We use the left part of the 3-cylinder mouse cortex volume of Kasthuri *et al*. [13] ($2048 \times 2048 \times 300$ voxels). The tissue is dense mammalian neuropil from layers 4 and 5 of the S1 primary somatosensory cortex of a healthy mouse and was acquired using ssEM. The resolution of our dataset is $3 \times 3\,\text{nm}^2$ per pixel, and the section thickness is 30 nm. The image data and a manually-labeled expert segmentation is publicly available as ground truth for the entire dataset[1].

AC4 subvolume. This dataset is part of a publicly available dataset of mouse cortex that was published for the ISBI 2013 challenge "SNEMI3D: 3D Segmentation of neurites in EM images". The resolution of this dataset is $6 \times 6 \times 30$ nm^3/voxel and it was acquired using ssEM. Haehn *et al*. found the most representative subvolume ($400 \times 400 \times 10$ voxels) of this dataset with respect to the distribution of object sizes [8]. This subvolume was used for Haehn *et al*. experiment on interactive proofreading tools for connectomics. We use the data, labeled ground truth and findings of their study which are published online[2].

CREMI A/B/C. As part of the MICCAI 2016 challenge on circuit reconstruction from electron microscopy images (CREMI) six ssTEM datasets (each $1250 \times 1250 \times 125$ voxels) were made publicly available[3]. Since only three include manually labeled ground truth, we use these three volumes for our experiments. The volumes are part of adult fruit fly (Drosophila melanogaster) brain. The resolution of all three datasets is $4 \times 4 \times 40$ nm^3/voxel.

Automatic segmentation pipeline. We use a state-of-the-art method to create a dense automatic segmentation of the data. Membrane probabilities are generated using a convolutional neural network based on the U-net architecture [26]. The probabilities are used to seed watershed and generate an oversegmentation using superpixels. Agglomeration is then performed by the GALA active learning classifier [23].

4.2. Classifier Training

To train our split error classifier, we use the L. Cylinder dataset. We use the first 250 sections of the data for training and validation. For n-fold cross validation, we select

[1]The Kasthuri 3-cylinder mouse cortex volume is available at https://software.rc.fas.harvard.edu/lichtman/vast/.

[2]The AC4 subvolume and results of the interactive proofreading experiment are available at http://rhoana.org/dojo/.

[3]The MICCAI CREMI challenge data is available at http://www.cremi.org.

one quarter of this data and re-select after each epoch. We minimize cross-entropy loss and update using stochastic gradient descent with Nesterov momentum [20]. To generate training data, we identify correct regions and split errors in the automatic segmentation by intersection with ground truth regions. This is required since extracellular space is not labeled in the ground truth but in our dense automatic segmentation. From these regions, we sample 112,760 correct and 112,760 split error patches with 4-channels as described above. The patches are normalized and to further augment our training data, we rotate patches within each mini-batch by $k * 90$ degrees with randomly chosen k. The training parameters such as filter size, number of filters, learning rate, and momentum are the result of intuition and experience, studying recent machine learning research, and a brute force parameter search within a limited range (see supplementary material). The first row of table 1 lists the final parameters. Our CNN configuration results in approximately 170,000 learnable parameters. We assume that training has converged if the validation loss does not decrease for 50 epochs. We test the CNN by generating a balanced set of 8,780 correct and 8,780 error patches using unseen data of the left cylinder dataset and report testing results in the second row of table 1.

Guided Proofreading	**Focused Proofreading**
Parameters	**Parameters**
Filter size: 3x3	Iterations: 3
No. Filters 1: 64	Learning strategy: 2
No. Filters 2-4: 48	Mito agglomeration: Off
Dense units: 512	Threshold: 0.0
Learning rate: 0.03-0.00001	
Momentum: 0.9-0.999	
Mini-Batchsize: 128	
Results	**Results**
cost [m]: 383	cost [m]: 217
Val. loss: 0.0845	
Val. acc.: 0.969	
Test. acc.: 0.94	Test. acc.: 0.99
Prec./Recall: 0.94/0.94	Prec./Recall: ?/?
F1 Score: 0.94	F1 Score: ?

Table 1. Training parameters, cost and results of our guided proofreading classifier versus focused proofreading by Plaza [25]. Both methods were trained on the same mouse brain dataset using the same hardware (Tesla K40 graphics card).

4.3. Baseline Comparisons

Interactive proofreading. A comparison of interactive proofreading tools by Haehn *et al*. concludes that novices perform best when using Dojo [8]. We use the publicly available findings of their user study as a baseline comparison. More specifically, we use the data of the Dojo user with the reported best performance.

Computer-aided proofreading. We compare against focused proofreading by Plaza [25]. Focused proofreading performs graph analysis on output from NeuroProof [2]. For training, we therefore replace GALA in our automatic segmentation pipeline with NeuroProof but use exactly the same input data including membrane probabilities. We obtained the best possible parameters for NeuroProof by consulting the developers and report them in table 1.

4.4. Experiments

Selection oracle evaluation.
Automatic method evaluation.
Forced choice user experiment.

5. Quantitative Results

6. Conclusions

References

[1] IEEE ISBI challenge: SNEMI3D - 3D segmentation of neurites in EM images. http://brainiac2.mit.edu/SNEMI3D, 2013. Accessed on 11/01/2016. 1, 2

[2] Neuroproof: Flyem tool, hhmi / janelia farm research campus. https://github.com/janelia-flyem/NeuroProof, 2013. Accessed on 03/15/2106. 2, 3, 6

[3] A. K. Al-Awami, J. Beyer, D. Haehn, N. Kasthuri, J. W. Lichtman, H. Pfister, and M. Hadwiger. Neuroblocks - visual tracking of segmentation and proofreading for large connectomics projects. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):738–746, Jan 2016. 2

[4] J. Anderson, S. Mohammed, B. Grimm, B. Jones, P. Koshevoy, T. Tasdizen, R. Whitaker, and R. Marc. The Viking Viewer for connectomics: Scalable multi-user annotation and summarization of large volume data sets. *Journal of Microscopy*, 241(1):13–28, 2011. 2

[5] J. A. Bogovic, G. B. Huang, and V. Jain. Learned versus hand-designed feature representations for 3d agglomeration. *CoRR*, abs/1312.6159, 2013. 2, 3

[6] D. B. Chklovskii, S. Vitaladevuni, and L. K. Scheffer. Semi-automated reconstruction of neural circuits using electron microscopy. *Current Opinion in Neurobiology*, 20(5):667 – 675, 2010. Neuronal and glial cell biology New technologies. 2

[7] R. J. Giuly, K.-Y. Kim, and M. H. Ellisman. DP2: Distributed 3D image segmentation using micro-labor workforce. *Bioinformatics*, 29(10):1359–1360, 2013. 2

[8] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis 2014)*, 20(12):2466–2475, 2014. 1, 2, 5, 6

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[10] 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. 1, 2

[11] Janelia Farm. Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2014. Accessed on 11/01/2016. 1, 2

[12] A. Karimov, G. Mistelbauer, T. Auzinger, and S. Bruckner. Guided volume editing based on histogram dissimilarity. *Computer Graphics Forum*, 34(3):91–100, May 2015. 3, 4

[13] 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. 5

[14] V. Kaynig, T. Fuchs, and J. Buhmann. Neuron geometry extraction by perceptual grouping in sstem images. In *Proc. IEEE CVPR*, pages 2902–2909, 2010. 2

[15] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Medical image analysis*, 22(1):77–88, 2015. 1

[16] J. S. Kim, M. J. Greene, A. Zlateski, K. Lee, M. Richardson, S. C. Turaga, M. Purcaro, M. Balkam, A. Robinson, B. F. Behabadi, M. Campos, W. Denk, H. S. Seung, and EyeWirers. Space-time wiring specificity supports direction selectivity in the retina. *Nature*, 509(7500):331336, May 2014. 2

[17] 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. 1

[18] K. Lee, A. Zlateski, A. Vishwanathan, and H. S. Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary detection. *arXiv preprint arXiv:1508.04843*, 2015. 2

[19] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen. A modular hierarchical approach to 3D electron microscopy image segmentation. *Journal of Neuroscience Methods*, 226(0):88 – 102, 2014. 1, 2

[20] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983. 6

[21] 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. 2

[22] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz. Graph-based active learning of agglomeration (GALA): A python library to segment 2D and 3D neuroimages. *Frontiers in Neuroinformatics*, 8(34), 2014. 1, 2

[23] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8, 2014. 5

[24] H. Peng, F. Long, T. Zhao, and E. Myers. Proof-editing is the bottleneck of 3D neuron reconstruction: The problem and solutions. *Neuroinformatics*, 9(2-3):103–105, 2011. 1, 2

[25] S. M. Plaza. Focused Proofreading: Efficiently Extracting Connectomes from Segmented EM Images, Sept. 2014. 2, 3, 6

[26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2, 5

[27] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. CATMAID: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009. 2

[28] R. Sicat, M. Hadwiger, and N. J. Mitra. Graph abstraction for simplified proofreading of slice-based volume segmentation. In *EUROGRAPHICS Short Paper*, 2013. 1, 2

[29] M. G. Uzunbas, C. Chen, and D. Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical Image Analysis*, 27:31 – 44, 2016. Discrete Graphical Models in Biomedical Image Analysis. 2

[30] A. Vázquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation fusion for connectomics. In *Proc. IEEE ICCV*, pages 177–184, Nov 2011. 2