Proofreading of Automatic Segmentations in Connectomics

Daniel Haehn*
Advisor: Hanspeter Pfister
Harvard University

1 Introduction

Our research targets the intersection of computer science and neuroscience — particularly in the field of *connectomics*. Connectomics aims to completely reconstruct the wiring diagram of the mammalian brain including billions of nerve cells and their connections [10, 14]. Scientists hope to better understand mental illnesses, learning disorders and neural pathologies by decoding this network [11]. To analyze neuronal connectivity at the level of individual synapses (i.e., connections between nerve cells), electron microscopy (EM) image stacks are processed. These images are acquired at nano-meter resolution. This results in volume sizes of hundreds of terabytes which then get automatically labeled and classified. Severe noise and artifacts present in the data make this procedure difficult and lead to errors. Therefore, the automatic labeling requires *proofreading* by humans. In fact, over 120 manual corrections are required on average per cubic micron of tissue [9]. These are mainly *split errors*, where a single segment is labeled as two, and *merge errors*, where two segments are labeled as one (Fig. 1). Split errors can be corrected by joining labels and the missing boundary in merge errors can be identified [6].

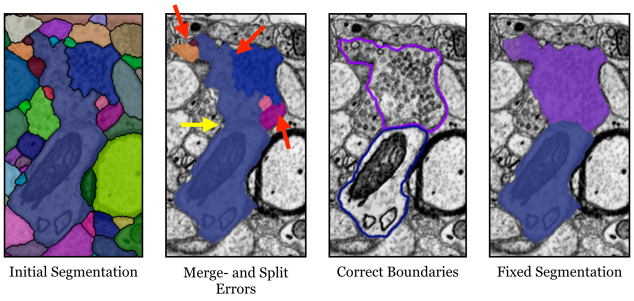


Figure 1: The most common proofreading corrections are fixing split errors (red arrows) and merge errors (yellow arrow).

To support this correction task, interactive proofreading tools and semi-automatic methods exist but are mostly geared towards domain experts [6, 15]. In connectomics, we instead need novice users to perform proofreading because of the large amounts of data — ideally in a distributed setting. Our studies have shown that this is quite challenging: In a controlled experiment, inexperienced users worsened the initial labeling instead of improving it [6]. Then, we discovered that simply finding errors is the most challenging step and takes the majority of time. This lead us to recently incorporate a machine learning based error suggestion method by training a convolutional neural network (CNN). The network detects and ranks split errors as well as merge errors. Likely errors and possible corrections are then suggested to the proofreader, correctable via point-and-click. Our initial results indicate that this enhances the proofreading performance. We now plan to incorporate a novel active learning approach to further increase the results; however, the ultimate solution as a combination of machine learning + user interface + visualization still has to be found.

*e-mail: haehn@seas.harvard.edu

2 Related Work

Janelia Farms identified the need for proofreading methods in 2010 [4]. The authors introduced Raveler [7], a software targeting expert users by offering many parameters for tweaking the proofreading process at the cost of a higher complexity. Raveler also includes a simple 3D renderer to validate corrections across slices.

The editing bottleneck for existing imperfect automatic segmentations was further specified by Peng et al. [12]. The authors developed software which supports three-dimensional proofreading and envision crowdsourcing to tackle large amounts of data.

EyeWire [1] exists since 2012 and is a popular online platform where users annotate retinal neurons. The platform is set up as an online game and participants earn virtual rewards for merging superpixels to reconstruct the data.

To simplify proofreading of small volumes, Sicat et al. proposed a graph abstraction method to guide users to problematic regions indicating the need for corrections [15]. Later, Plaza suggested a focused proofreading approach using heuristics to concentrate on certain regions of image data [13]. The main driver of his approach is the size of the segment — ignoring very small regions. By doing so, Plaza limits user decisions to yes/no questions and suggests crowdsourcing as a potential platform for proofreading.

In 2014, we developed two proofreading tools: Mojo and Dojo. Mojo provides a simple scribble interface for error correction, and Dojo extends this for distributed proofreading via a minimalistic web-based user interface. We defined requirements for general proofreading tools, and then evaluated the accuracy and speed of Raveler, Mojo, and Dojo through a quantitative user study (Sec. 3 and 4) [6]. Al-Awami et al. then integrated Dojo into the Neuroblocks proofreading management system [2], which tracks and visualizes progress among multiple users.

In 2015, Karimov et al. proposed a guided volume editing approach using histogram dissimilarity [8]. Measuring differences in histogram distributions helps to find potential errors and to suggest possible corrections. These promising results on Computed-Tomography Angiography datasets are targeted towards expert users. Most recently, Uzunbas et al. showed that potential labeling errors can be found by considering the merge tree of an automatic segmentation method [16]. The authors track uncertainty through automatic labeling and then present potential regions for proofreading. This requires access to the segmentation merge tree, which is not always available. However, none of these methods leverage machine learning to enhance the proofreading performance.

3 Preliminary Methods

We explored methods for fully interactive proofreading and compared different available software products. Since correcting segmentations is especially difficult for novice users, we started looking into ways on how to use machine learning to support proofreading.

3.1 Interactive Proofreading

A powerful proofreading tool is crucial for enhancing segmentations efficiently and effectively. Particularly, communicating the three dimensional property of EM stacks is essential for identifying segmentation errors in 3D and for confirming the correctness of changes in the automatic segmentation. Existing software solutions tend to be geared towards domain experts and often have a steep learning curve. One example is Raveler [7], a stand-alone proofreading tool developed at the Janelia Farms Research Campus. However, in connectomics, the huge scale of data requires that proofreading is crowdsourced. This means that proofreading really has to be performed by non-domain-experts and novice users.

In collaboration with neuroscientists at the Harvard Center for Brain Science we have developed two software tools to increase the efficiency, accuracy and usability for proofreading large and complex EM data. First, we have developed Mojo — a powerful stand-alone software application for experienced as well as non-expert users with advanced semi-automatic proofreading features. Building on the experience we gained from Mojo, we developed Dojo ("Distributed Mojo") — a web-based, distributed proofreading application (Fig. 2). Dojo is geared towards non-expert users and offers easy access through a web browser. Additional features include 3D volume rendering and parameter free tools for collaborative proofreading.

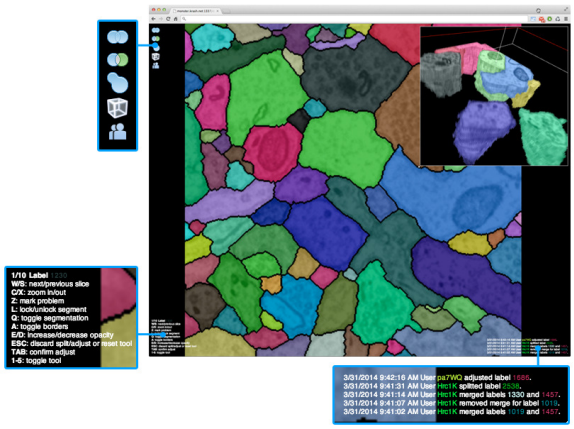


Figure 2: The user interface of Dojo consists of the 2D slice view (center), the 3D volume renderer (top right corner), the toolbox (top left corner), additional textual information (bottom left corner) and an activity log including all connected users (bottom right corner).

In a paper published at IEEE Vis 2014, we defined a set of requirements and design guidelines for visual proofreading applications [6]. We think that proofreading tools should be as minimalistic and as easy-to-use as possible. In addition, 3D visualization is especially important for novice users to grasp an understanding of the volumetric EM data. We evaluated our claims by performing a comparative user study between Mojo, Dojo and Raveler. 30 participants with no former experience in proofreading EM data were recruited and randomly assigned to each tool. The participants were asked to proofread a small representative subvolume of connectomics data for 30 minutes. Our results showed that novice users have problems performing this task (Sec. 4).

3.2 Machine Learning Supported Proofreading

With simple user interaction, split and merge errors can be corrected. However, even with semi-automatic correction tools, the visual inspection of the data to find the errors in the first place takes the majority of the time. Our goal was to add automatic detection of split and merge errors to proofreading tools. Instead of the user visually inspecting the whole image volume, we designed automatic classifiers that detect split and merge errors in 2D segmentations.

Inspired by work proposed by Bogovic et al. for label agglomeration [3], we built a split error classifier using a convolutional neural network. Our CNN checks the boundaries of an existing automatic segmentation. We use multiple channels of context information of the boundary for the decision making process such as the gray scale EM image, a probability mask, a label binary mask and a border binary mask (Fig. 3). Using this information, the CNN provides a probability for a split error for each boundary in automatically labeled data. Two variations of the CNN were trained: *RGBANet* with a combined 4-channel input and *MergeNet* with four separate input channels which then get concatenated. We sample up to 10 decision points for each boundary. The probabilities of the decision points then get weighted by the length of the boundary and averaged. A greedy algorithm finally merges neighboring regions sequentially, starting with the highest probability score.

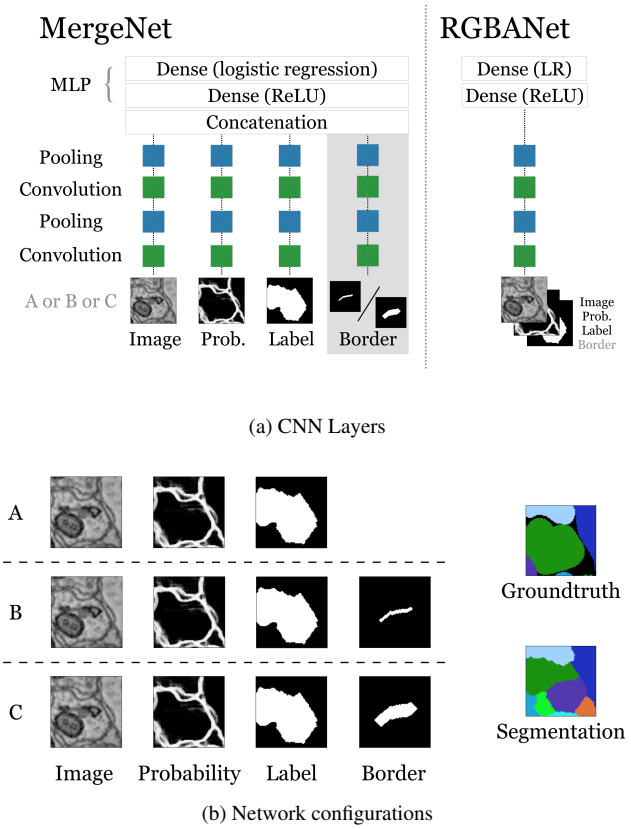


Figure 3: (a) Our network architecture with up to four input channels, each with two convolutional and two pooling layers. MergeNet includes four separate input channels and RGBANet uses a combined 4-channel input. (b) We trained three different network configurations with three and four inputs: A) image, boundary map probability, and merged binary mask; B) A configuration extended with a small border mask, to focus on the specific boundary in question; C) A configuration extended with a large border mask.

For merge errors, we generate randomly seeded watershed splits in a cell and then let our split error network rate their borders. If the CNN detects a boundary with a very low split score, then the boundary should have been in the segmentation and the region is a candidate for a merge error. This way, we were able to use one

network for both split errors and merge errors.

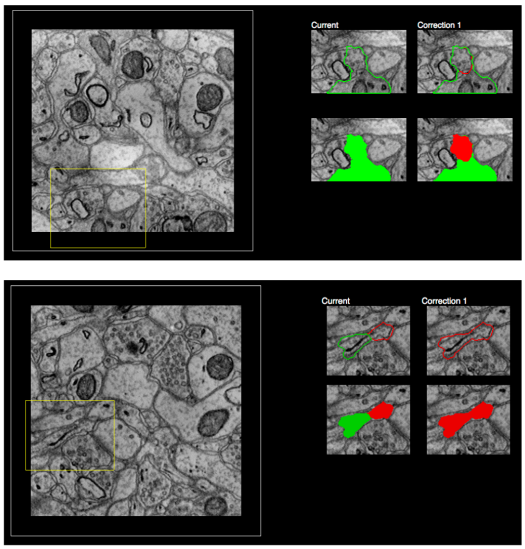


Figure 4: The recommendation user interface showing the current labeling of a cell boundary and the proposed correction as well as an overview of the whole image. A merge error is shown in the top and a split error is shown in the bottom.

We trained our CNN with 500k+ erroneous and correct patches of a manually labeled large connectomics volume (2048 x 2048 x 250 voxels). The training resulted in validation loss of 0.07 and test accuracy of 90.8%. These numbers lead to our initial hope that our network in combination with a greedy algorithm would be able to perform proofreading fully automatically. Unfortunately, this was not the case and it turned out that the user still has to be in the loop (Sec. 4). For this purpose, we designed a very simple user interface which shows a proofreading error and a potential correction — one at a time (Fig. 4). The CNN then recommends likely errors and their corrections for proofreading within the interface. We call this approach *Guided Proofreading*.

4 Preliminary Results

First, we compared the interactive proofreading tools Mojo, Dojo and Raveler through a quantitative user study. Second, we used our machine learning based classifier to recommend likely errors for correction and compared the results against Dojo users.

4.1 Interactive Proofreading

We conducted a quantitative user study to evaluate the performance and usability of three proofreading tools. In particular, we evaluated how nearly untrained non-experts can correct an automatic segmentation using Mojo, Dojo and Raveler. We designed a between-subjects experiment and asked the participants to proofread a small dataset in a fixed time frame of 30 minutes. We used the most representative sub-volume of a larger freely available real-world data set where expert ground truth is available. We recruited participants from all fields with no experience in electron microscopy data or proofreading of such. As baseline, we asked two domain experts to label the same sub-volume from scratch using manual segmentation in the same fixed time frame.

We measured proofreading performance using the similarity measure variation of information (VI) between the ground truth and the proofreading output. The participants performed slightly better with Dojo (Fig. 5). Interestingly, the majority of participants made the initial segmentation worse rather than correcting it.

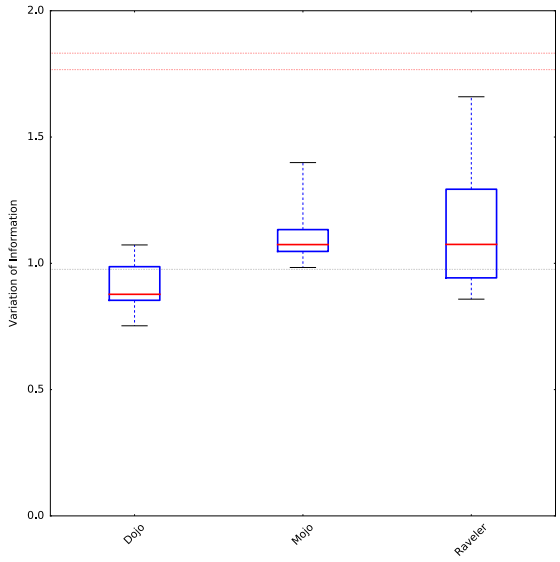


Figure 5: We compared the three interactive proofreading tools Dojo, Mojo and Raveler. Here we show the VI similarity measure for each tool after proofreading for 30 minutes (lower is better). Participants using Dojo achieved a lower VI than subjects with the other tools. These results are statistically significant. The gray line shows the VI of the automatic segmentation before proofreading. The red lines show the VI of the experts' manual segmentation after 30 minutes.

This let us conclude the following: a) proofreading for novices is hard without further guidance, and b) a minimalistic and easy-to-use software tool enables better corrections. The detailed results are available in our paper [6] and the developed tools are available as open source software[1].

4.2 Machine Learning Supported Proofreading

We repeated the experiment described above but now let our classifier recommend likely errors for proofreading. Since the participants performed best using Dojo, we compared against the average and the best Dojo user. One novice user and two expert users were asked to perform proofreading using the recommendation interface (Fig. 4). The experimental conditions were exactly the same as in our interactive proofreading study.

Again, we measured proofreading performance using VI between the ground truth and the proofreading output. The recommendation interface enabled both novice and expert users to perform better than the best Dojo user (Fig. 6). This was not surprising since the classifier suggested where to proofread first.

To further evaluate our classifier, we performed the experiment fully automatic and with a simulated perfect user (error rate 0) and random recommendations. The fully automatic approach did not yield good results due to the complexity of the data — our greedy algorithm made things worse after a probability threshold was reached. The performance of random recommendations showed that the ranking by our classifier indeed is valuable.

[1]Our interactive proofreading software and a live demo are available at http://rhoana.org/dojo/.

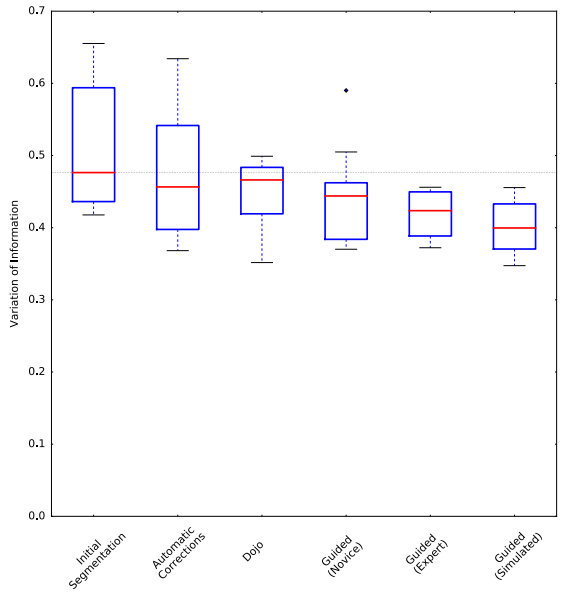


Figure 6: Guided Proofreading vs. Interactive Proofreading: We compare distributions of VI measures across 10 sections for the initial automatic segmentation, a fully-automatic correction of recommended errors based on a threshold of acceptance, the best participant in our Dojo user study, a novice user and two experts using our recommendation user interface and our simulated user. Lower scores are better.

Our work on machine learning supported proofreading is currently being extended by incorporating active learning.

5 Proposed Work

Proofreading will always be necessary since every automatic method makes errors. In addition, the human will always be the driving force behind every proofreading attempt. Based on our observations so far, we think that the ultimate proofreading tool is a combination of machine learning, user interface design and (information-)visualization.

We currently extend our trained CNN by adding an *active learning* component inspired by generative adversarial nets [5]. To improve our network and identify a possible pattern in wrong classifications, we use a second CNN (Fig. 7). This network is trained to distinguish inputs from our classifier and from human labeled annotations. Ideally, both inputs should be non-distinguishable — meaning that if the new network does not classify well, our original classifier does. However, outputs identified as originating from our original classifier will be re-evaluated by an oracle and re-added as training data to our original classifier.

Our hope is that the new network identifies a certain classification pattern in our original trained network. By re-adding these patches as training data, our original classifier will be fine tuned. Finally, we think that additional gain in proofreading performance can be made by adding visualizations to the user interface. We want to explore different visualizations to a) reduce the correction time for each error and b) aid the correction decisions (especially for novice users). Finally, our future proofreading solution will be used by hundreds of high school students in the Boston area to help process the large amounts of connectomics data.

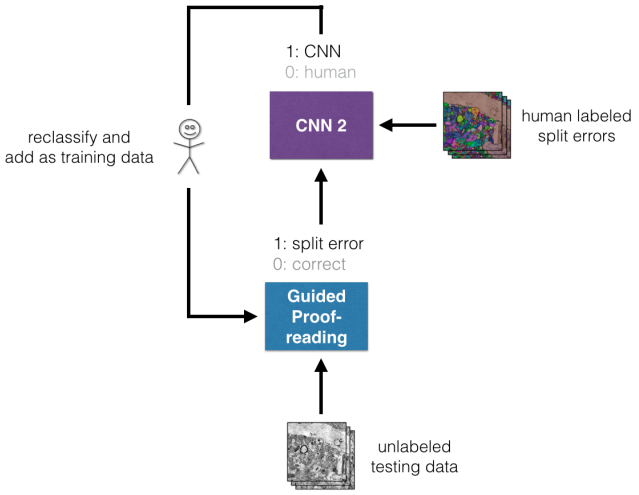


Figure 7: Our proposed active learning approach: *CNN 2* (purple) distinguishes whether a split error was identified by a human or by our guided proofreading classifier (blue). Errors detected as identified by our classifier are then re-evaluated by a human. If *CNN 2* classifies well, there is most likely a pattern of wrong classifications which then gets re-used as training data for our classifier.

6 Discussion

The task of automatic cell boundary segmentation is difficult, and trying to improve such segmentations automatically as a post-process through split and error correction is, in principle, no different than trying to improve the underlying cell boundary segmentation. Due to the task difficulty, manual proofreading of connectomics segmentations is necessary, but it is a time consuming and error-prone task.

However, there is value in being able to recommend to users possible regions for correction, as the time cost of proofreading is dominated by the visual search for errors. Besides optimizing our CNN performance by leveraging an active learning approach, we think that adding (information-)visualization components can improve the results and/or reduce the time spent analyzing errors. The visualizations have to be simple and reduce rather than add complexity. Hopefully, this will ultimately lead to expert-comparable proofreading performance for novice users.

7 Conclusion

Large amounts of images, such as in connectomics, require a lot of manpower to correct the automatic labeling. The proofreading task itself can not be done automatically and specifically requires the help of non-experts. Currently existing proofreading tools do not let novice users proofread efficiently — even with semi-automatic features. The perfect proofreading solution most likely requires a combination of machine learning, user interface design and visualization.

References

[1] Eyewire. http://eyewire.org/.

[2] 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.

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

[4] 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.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[6] 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 (Proceedings IEEE SciVis 2014)*, 20(12):2466–2475, 2014.

[7] Janelia Farm. Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2014. Accessed on 31/03/2016.

[8] 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.

[9] 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.

[10] J. W. Lichtman and W. Denk. The big and the small: Challenges of imaging the brain's circuits. *Science*, 334(6056):618–623, 2011.

[11] J. W. Lichtman, J. Livet, and J. R. Sanes. A technicolour approach to the connectome. *Nature reviews. Neuroscience*, 2008.

[12] 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.

[13] S. M. Plaza. Focused Proofreading: Efficiently Extracting Connectomes from Segmented EM Images, Sept. 2014.

[14] S. Seung. *Connectome: How the Brain's Wiring Makes Us Who We Are*. Houghton Mifflin Harcourt, Feb. 2012.

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

[16] 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.

Design and Evaluation of Interactive Proofreading Tools for Connectomics

Daniel Haehn, Seymour Knowles-Barley, Mike Roberts,
Johanna Beyer, Narayanan Kasthuri, Jeff W. Lichtman, and Hanspeter Pfister

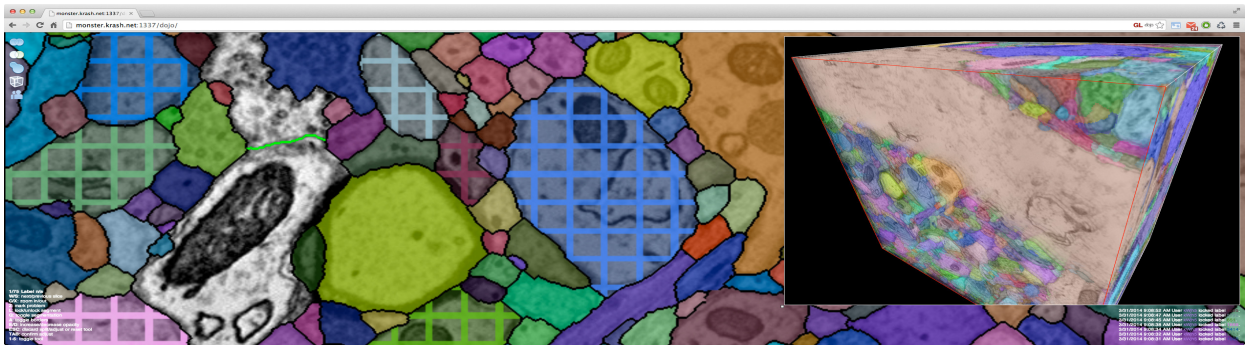


Fig. 1: Proofreading with Dojo. *We present a web-based application for interactive proofreading of automatic segmentations of connectome data acquired via electron microscopy. Split, merge and adjust functionality enables multiple users to correct the labeling of neurons in a collaborative fashion. Color-coded structures can be explored in 2D and 3D.*

Abstract—Proofreading refers to the manual correction of automatic segmentations of image data. In connectomics, electron microscopy data is acquired at nanometer-scale resolution and results in very large image volumes of brain tissue that require fully automatic segmentation algorithms to identify cell boundaries. However, these algorithms require hundreds of corrections per cubic micron of tissue. Even though this task is time consuming, it is fairly easy for humans to perform corrections through splitting, merging, and adjusting segments during proofreading. In this paper we present the design and implementation of *Mojo*, a fully-featured single-user desktop application for proofreading, and *Dojo*, a multi-user web-based application for collaborative proofreading. We evaluate the accuracy and speed of *Mojo*, *Dojo*, and *Raveler*, a proofreading tool from Janelia Farm, through a quantitative user study. We designed a between-subjects experiment and asked non-experts to proofread neurons in a publicly available connectomics dataset. Our results show a significant improvement of corrections using web-based *Dojo*, when given the same amount of time. In addition, all participants using *Dojo* reported better usability. We discuss our findings and provide an analysis of requirements for designing visual proofreading software.

Index Terms—Proofreading, Segmentation, Connectomics, Quantitative Evaluation

1 Introduction

In computer vision, image segmentation is the process of partitioning an image into several sub-regions or segments, where individual segments correspond to distinct areas or objects in the image. Automatic segmentation approaches eliminate user interaction, which is often the bottleneck of manual and semi-automatic segmentation techniques. However, automatic approaches are computationally expensive and need to be targeted towards very specific segmentation problems and data sets to achieve high-quality results. Furthermore, even optimized automatic segmentation algorithms usually exhibit higher error rates and are less accurate than manual expert segmentations. Manual pixel labeling, on the other hand, requires users to have domain-specific knowledge and is usually a tedious and time-consuming process. A powerful alternative to fully automatic or manual approaches are interactive semi-automatic techniques. They usually rely on minimal user input to achieve an initial result and allow users to further improve the segmentation by manual adjustments. *Proofreading* refers to the manual and semi-automatic correction of automatic segmentations as a post-processing step. In a proofreading tool, segments can be quickly joined or split to produce the correct segmentation faster than it would be possible with manual annotation. The combination of automatic segmentation and proofreading is the preferred option for large data sets where manual segmentation is not feasible.

Our work stems from a collaboration with neuroscientists in the field of *connectomics*. Connectomics aims to completely reconstruct the wiring diagram of the mammalian brain at nanometer resolution, comprising billions of nerve cells and their interconnections [32, 44]. By deciphering this vast network and analyzing its underlying properties, scientists hope to better understand mental illnesses, learning disorders and neural pathologies [33]. However, to analyze neuronal connectivity at the level of individual synapses (i.e., connections between nerve cells), high-resolution electron microscopy (EM) image stacks have to be acquired and processed (Fig. 2). These image stacks are typically on the order of hundreds of terabytes in size and often exhibit severe noise and artifacts. The huge size of these volumes makes (semi-)automatic segmentation approaches the only viable option, however, the complex structure of the data provides difficulties

- *Daniel Haehn, Johanna Beyer, and Hanspeter Pfister are with the School of Engineering and Applied Sciences at Harvard University. E-mail: {haehn,jbeyer,pfister}@seas.harvard.edu.*
- *Seymour Knowles-Barley, Narayanan Kasthuri, and Jeff W. Lichtman are with the Center for Brain Science at Harvard University. E-mail: seymourkb@seas.harvard.edu, bobby.kasthuri@gmail.com, jeff@mcb.harvard.edu.*
- *Mike Roberts is with the Computer Graphics Laboratory at Stanford University. E-mail: mlrobert@stanford.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

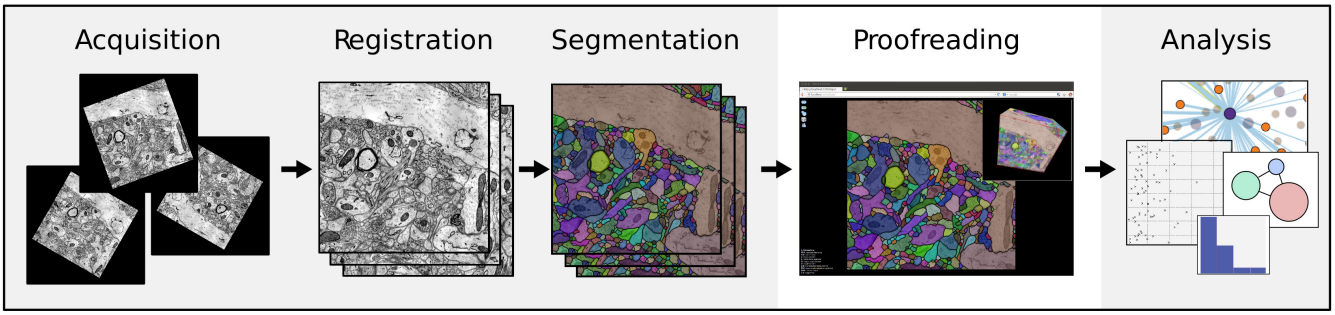


Fig. 2: ***Proofreading as part of the Connectome workflow.*** *Electron microscopy data of the mammalian brain gets acquired, registered and segmented. Since the output of the automatic segmentation algorithm is not perfect, proofreading is a mandatory stage before any analysis.*

for automatic segmentation. The resulting segmentations, on average, require over 120 manual corrections per cubic micron of tissue [30]. Previous research has focused on image acquisition, segmentation [28] and interactive visualization and analysis [16, 8, 7], but little research has focused on the proofreading stage [38].

A powerful proofreading tool is crucial for enhancing segmentations efficiently and effectively. Particularly, communicating the three dimensional property of EM stacks is essential for identifying segmentation errors in 3D and for confirming the correctness of changes in the automatic segmentation. Existing software solutions tend to be geared towards domain experts and often have a steep learning curve. One example is Raveler [23], a stand-alone proofreading tool recently developed at the Janelia Farm Research Campus. The huge scale of the data that needs to be segmented and proofread, however, requires that proofreading will have to be crowdsourced in the future. This, on the other hand, implies that proofreading will have to be performed in a distributed setting by non-domain-experts and novice users.

In collaboration with neuroscientists at the Harvard Center for Brain Science we have developed two novel tools to increase the efficiency, accuracy and usability for proofreading large, complex EM data. First, we have developed Mojo–a powerful stand-alone software application [30] for experienced as well as non-expert users that offers advanced semi-automatic proofreading features. Building on the experience we gained from Mojo, we developed Dojo ("Distributed Mojo")–a web-based, distributed proofreading application that includes collaborative features and is geared towards non-expert users. Dojo offers easy access through a web browser, no installation requirements, 3D volume rendering, and a clean user interface that improves usability, especially for novice users. To evaluate the effectiveness of Dojo, we perform a quantitative user study. The study targets non-experts from all fields with no previous knowledge of proofreading electron microscopy data. We compare Dojo against Mojo and Raveler on a representative sub-volume of connectomics data. The study is designed as a between-subjects experiment with very little training for all participants and a fixed time frame of thirty minutes to proofread the given dataset. As a baseline, we also asked two domain experts to label the same sub-volume from scratch using manual segmentation.

Our first contribution is a set of requirements and design guidelines for visual proofreading applications based on our interviews with domain experts and feedback from an initial deployment of Mojo to non-expert users, interns and high-school students. Our second contribution is the design and development of Mojo, a stand-alone software application for proofreading. Based on our experiences during our work on Mojo, we defined requirements for a successive software which aims at increasing the usability of a proofreading tool geared towards non-expert users. These thoughts have led to the development of web-based Dojo, the third contribution of this paper. Dojo is easier to use and adds 3D volume rendering and collaborative features. Our final contribution is our quantitative user study. We present statistically significant results showing that novice users of Dojo are able to proofread a given data set better and faster than with existing proofreading tools. Based on these results we present design guidelines that will help developers of future proofreading tools.

2 Related Work

Connectomics. Neuroscience and especially connectomics with its goals and challenges [32] has received a lot of attention recently. Seung [43] motivates the need for dense reconstruction of neuronal structures, underpinning the requirement for segmentation methods that automatically label huge volumes of high-resolution EM images.

Registration, segmentation and annotation for neuroscience. Currently, the main bottleneck for analyzing connectomics data is the need for registration and segmentation of the underlying image data (Fig. 2). EM images are acquired slice by slice by physically cutting a block of tissue. Registration has to be performed between slices and also between individual sub-tiles of a single slice. Several methods focus on registration of large EM data [5, 8, 42].

Segmentation methods for connectomics can be classified into manual [10, 12], semi-automatic [3, 20, 26, 27, 40, 47, 49], and automatic [22, 28, 34, 36, 37, 48] approaches. While initially manual and semi-automatic approaches were very popular, in recent years the need for automatic approaches that are scalable to volumes of hundreds of terabytes has become apparent. ITK-SNAP [49] supports semi-automatic segmentation methods based on active contours as well as manual labeling. Vazquez-Reina et al. [48] propose an automatic 3D segmentation of EM volumes by taking the whole volume into account rather than a section-to-section approach. They formulate the segmentation as the solution to a fusion problem with a global context. Kaynig et al. [28] propose a pipeline for the automatic reconstruction of neuronal processes that is based on a random forest classifier coupled with an anisotropic smoothing prior in a conditional random field framework and 3D segment fusion. This pipeline is integrated into the RhoANA open-source software available at http://www.rhoana.org. Automatic segmentation methods are more scalable than manual or semi-automatic approaches, but often require a clean-up or proofreading step where incorrect segmentations are fixed. In 2013 the IEEE ISBI challenge [2] called for machine learning algorithms for the 3D segmentation of neurites in EM data. The organizers provided test and training data, and a manual expert segmentation. While the results were quite impressive, all algorithms still exhibited a rather high error rate, motivating the need for proofreading. We use the data of that challenge in our user study.

Collaborative segmentation and annotation. EyeWire [3] is an online segmentation tool where novice users participate in a segmentation game to get points for segmenting neuronal structures using a semi-automatic algorithm. D2P [15] uses a micro-labor workforce approach (based on Amazon's Mechanical Turk) where boolean choice questions are presented to users and local decisions are combined to produce a consensus segmentation. Catmaid [41] and the Viking Viewer [6] are collaborative annotation frameworks for experts, that allow users to create skeleton segmentations for terabyte-sized data sets. However, they do not offer proofreading support.

Visualization and visual analytics for connectomics. A good overview of visualization for connectomics and human connectomics is given by Pfister et al. [39] and Margulies et al. [35] respectively. Most visualization frameworks for connectome data have only basic

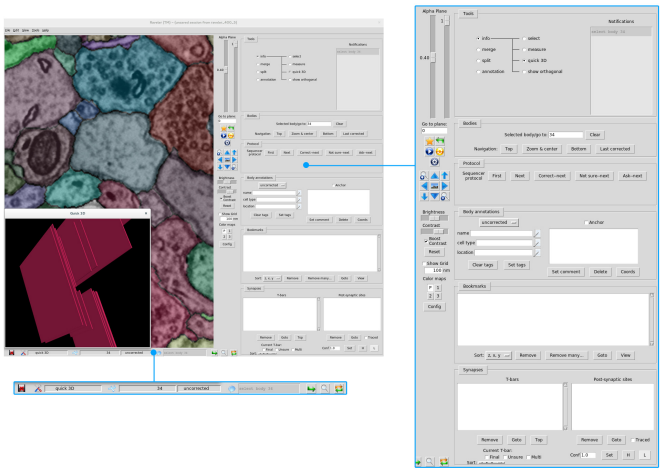


Fig. 3: ***User interface of Raveler by Janelia Farm.*** *The interface consists of the 2D slice view (center), the toolbox (right), additional textual information (bottom) and a simple 3D renderer showing bounding boxes of segments (bottom left).*

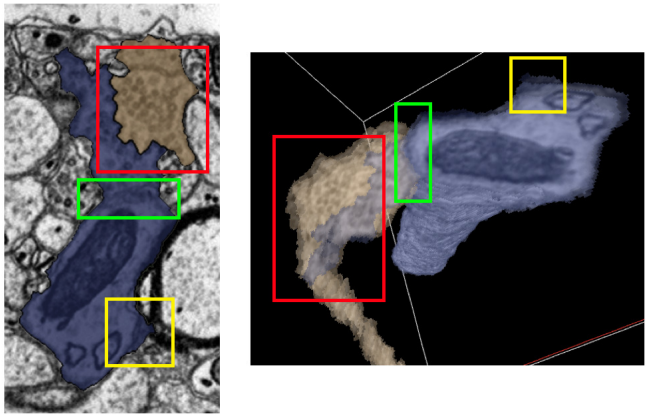


Fig. 4: ***Segmentation errors.*** *Common errors of automatic segmentation algorithms include merge errors (green), split errors (red) and boundary errors (yellow). The 2D slice visualization is shown on the left and Dojo's 3D volume rendering is shown on the right.*

support for 3D rendering [1] and focus on displaying network maps of connected brain regions [6]. The Connectome Viewer [13] offers a general processing and plug-in framework for visualization. Hadwiger et al. [16] propose a visualization-driven petavoxel volume rendering framework for high-resolution electron microscopy streams. For visual analysis of connectomics data, interactive query systems have been proposed [7, 31]. None of these systems, however, run in a distributed multi-user environment and many need high-performance workstations and modern GPUs for rendering and data processing.

Collaborative and web-based visualization of image data. Ginsburg et al. [14] propose a visualization system for connectome data based on WebGL [29]. They combine brain surface rendering with tractography fibers and render a 3D network of connected brain regions. The X toolkit [18] offers WebGL rendering for neuroimaging data, and SliceDrop [17] is a web-based viewer for medical imaging data that supports volume rendering and axis-aligned slice views. Jeong et al. [25] describe an online collaborative pathology viewer. None of these frameworks, however, support volume rendering of segmented data or interactively updating segmentations.

Proofreading. Until recently, visual proofreading methods for automatically generated segmentations have not received a lot of attention. Sicat et al. [46] propose a graph abstraction method to simplify proofreading. They construct a skeleton of the segmentation and identify potential problematic regions in the segmentation and guide users to these areas. Raveler [23] (Fig. 3), by Janelia Farms, is used for annotation and proofreading and uses a quadtree-based tiling system to support large data. It targets expert users and offers many parameters for tweaking the proofreading process at the cost of a higher complexity. In this paper we introduce two novel proofreading tools: Mojo (Sec. 4.1) and Dojo (Sec. 4.2); and compare them to Raveler in a quantitative user study (Sec. 5).

3 Visual Proofreading

In this section we introduce the overall workflow for proofreading high-resolution EM image stacks and discuss common issues (e.g., usability and scalability) before presenting a detailed requirement analysis of the necessary tasks in a scalable proofreading tool.

3.1 Proofreading Workflow

The visual proofreading workflow consists of three main steps:

1. Searching for structures containing segmentation errors.
2. Modifying the existing segmentation to fix the errors.
3. Confirming the correctness of the modified segmentation.

The first step – searching for segmentation errors – requires users to be very focused. This is especially true for volumetric data because tracking 3D structures in a 2D slice-based visualization involves constant switching between slices. Segmentation data is typically displayed as a colored overlay on top of the original image data. A common strategy to spot segmentation errors is to continuously toggle the visibility of this segmentation overlay to compare the labeled boundaries to the actual image data. During this search three different types of errors can be spotted: a) *merge errors or under-segmentation*, where two separate structures are erroneously connected; b) *split errors or over-segmentation*, where a single structure is erroneously split into several different segments; and c) *boundary errors*, where the boundaries of a labeled segment do not match the boundaries of the structure in the image data (Fig. 4).

The second step – modifying the existing segmentation – is mentally less demanding than the search phase but needs good tool support, to allow users to quickly and correctly modify the existing segmentation. The most common semi-automatic tools for correcting the segmentation correspond to the three error types: merge, split, and adjust. Once a segmentation error is spotted, the user chooses the appropriate tool to fix the error. Merge errors can be corrected by using the split tool to divide the segments. Split errors can be corrected by joining the segments using the merge tool. Boundary errors can be fixed by adjusting either the erroneous label or the neighboring labels. In connectomics data, these errors always look similar (e.g., a merge error fails to detect a dark boundary between two lighter structures). Therefore, after understanding these three error types it should be possible, even for non-expert users, to recognize and subsequently fix these errors.

The last step – confirming the correctness of the modified segmentation – requires the user to check if the modifications did fix the segmentation error, or if another correction step is necessary. Most proofreading tools only offer a 2D slice view for that task, however, we propose that an additional 3D view can significantly improve the speed and accuracy in which users can confirm their modifications.

3.2 Proofreading Issues

We identified several issues that have to be considered when designing proofreading tools for connectomics data. The main problems in proofreading tools usually correspond to usability issues, scalability issues, and failing to target the system towards the expected users, their needs and background knowledge. Developers of systems for experts can expect users to have a thorough domain background as well as a higher motivation to use the system, hence higher frustration threshold. Most proofreading tools are geared towards expert users and offer a full-featured segmentation framework and often a very complex user interface (Fig. 3). When designing a system for non-experts, however,

Requirement	Mojo	Dojo	Raveler
R1. Navigation	2D	2D+3D	2D+simple 3D
R2. E. Detection	manual	manual	manual
R3. E. Correction	advanced	parameter-free	very advanced
R4. Validation	2D	2D+3D	2D+simple 3D
R5. Collaboration	n/a	yes	n/a
R6. Deployment	download	web access	compilation
R7. GUI	complex	minimalistic	very complex

Table 1: ***Comparison of proofreading tools.*** *Features offered by Mojo, Dojo and Raveler are summarized in respect to the requirements as identified in Section 3.3.*

one has to pay particular attention to usability issues and guidance that will help novice users to perform their proofreading tasks.

Data set sizes in connectomics are increasing at a rapid pace. Currently our collaborators routinely acquire datasets of several terabytes, but in the future this will continue to grow. One solution for handing large data is to work on sub-volumes of the data. However, it is common that neuronal structures extend across long distances especially when images are in nanometer pixel resolution. These sub-volumes would then have to be merged in an extra step to guarantee that structures across block boundaries receive corresponding labels. The problem of combining segmentations is further complicated in settings where multiple users work on the same dataset concurrently. Without a collaboration mode, each user has to either work on a different sub-volume, or segmentations of different users might overlap. The former is not scalable to a system with potentially thousands of users while the latter can result in conflicts where the segmentation between users differs, and which will have to be solved in a post-processing step. Collaborative approaches would allow users to discuss unclear areas or conflicts directly during the proofreading process.

3.3 Requirement Analysis

We regularly met with our collaborating scientists to discuss their goals and define required user tasks for proofreading. We conducted informal as well as semi-structured interviews with them over the course of several months and identified several domain-specific proof reading tasks. After initial development, we installed Mojo at the lab of our domain experts, where it was used for a first informal user study. After this initial testing phase, Mojo was used by a larger group of non-experts (20+ high-school students and lab interns) to correct the segmentation of a small data set of a mouse cortex. The feedback acquired in this first deployment step led us to the development of Dojo, targeting a web-based, non-expert, multi-user environment.

We have identified the following general requirements for proofreading large-scale EM image stacks:

R1. Intuitive navigation inside the 3D image stack. Users have to be able to easily and intuitively navigate within the 3D data including zooming and navigating within a single slice and multiple slices. A 3D view is necessary to help non-expert users grasp the spatial relations in the data and should be able to display either the entire volume or only selected user-defined structures.

R2. Detection of segmentation errors. To quickly detect segmentation errors the user has to be able to easily switch between different rendering modes (i.e., showing the raw data, showing the segmentation data, showing the segmentation outline).

R3. Fast correction of errors. Correcting errors has to be as simple and easy as possible. Manual corrections have to be supported, but semi-automatic methods for splitting and merging should be the main interaction metaphors and have to be accurate, fast, and easy to use.

R4. Checking the correctness of modified segmentations. Non-expert users have to be able to quickly judge the correctness of their last modification. This requires visualizing and highlighting the modified segment in 2D as well as in 3D.

R5. Collaborative proofreading environment. The system has to support multiple simultaneous users. The amount of data that needs to be processed necessitates crowdsourcing the proofreading step. This means that multiple users have to be able to work on the same data set and giving users the ability to work together collaboratively.

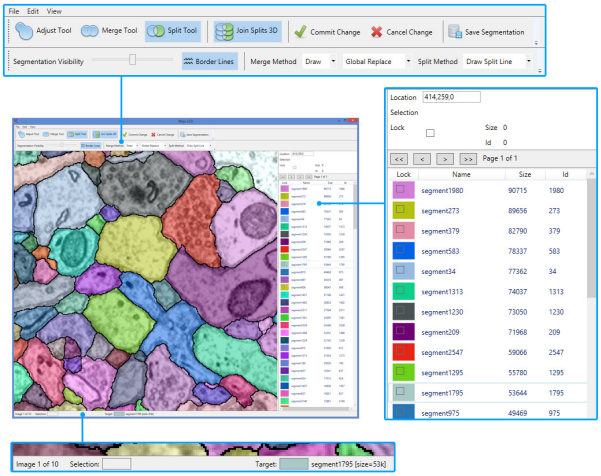


Fig. 5: ***User interface of Mojo.*** *The interface consists of the 2D slice view (center), the toolbox (top) and additional textual information as well as the label list (bottom, right).*

R6. Simple deployment. A crowdsourcing system has to support easy deployment and must not require any special hardware. High-schools, for example, do not allow installing any external software on their computers. To enable high-school students to work on proofreading, the system has to be web-based and should run in every browser.

R7. Minimalistic GUI. The final requirement targets the usability of the system. To support non-expert users, the user interface has to be simple and easy to understand. It is better to have fewer options that are well understood by its users, than to have a cluttered user interface that confuses novice users.

Table 1 summarizes the different features supported by each of the proofreading tools.

4 Technology

In the following section we describe the technology behind Mojo and Dojo. Mojo was developed as a powerful standalone application and is now used by domain experts and trained researchers to perform proofreading. To overcome the limitations of Mojo regarding installation, multi-platform support and hardware requirements, we also developed a web-based proofreading tool. Dojo does not require any installation and can be accessed with a web browser such as Google Chrome and also runs on tablets and smartphones.

4.1 Mojo - A Standalone Proofreading Tool

Mojo supports the computer-assisted correction of automatic segmentation results by providing a simple scribble interface for correcting merge and split errors in automatic segmentations. Mojo is capable of annotating any volumetric segmentation data, but works best when a reasonable automatic segmentation can be provided as a starting point. Mojo is written in C++ and C# and available as open source software at http://www.rhoana.org.

4.1.1 User Interface

The Mojo GUI (Fig. 5) reflects a typical desktop application, with a toolbar at the top of the window to control interface and view options, a 2D slice view window showing the current segmentation and EM data, and a segment list showing segment names, ids and sizes.

4.1.2 Proofreading Features

The Mojo interface provides split, merge and adjust tools that enable edit operations to be performed with simple, wide brush strokes. This allows annotation to be performed on a standard workstation, without the use of a tablet drawing or touchscreen device. The interaction mode of each tool can be customized in the Mojo toolbar.

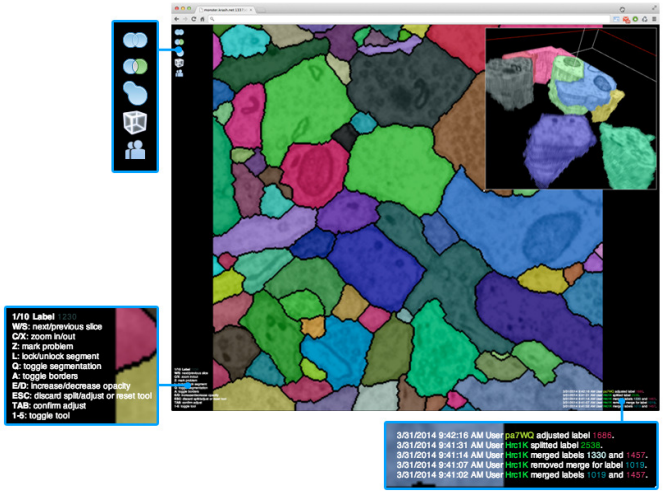


Fig. 6: ***User interface of Dojo.*** *The interface consists of the 2D slice view (center), the 3D volume renderer (top right corner), the toolbox (top left corner), additional textual information (bottom left corner) and an activity log including all connected users (bottom right corner).*

The merge tool offers two interaction modes; draw and click. In draw mode, the user draws a wide brush stroke over the objects to be merged. Any objects touched by this brush stroke are combined into a single object. In click mode the user selects a seed object with a left click and merges additional objects to the seed object with a right click. Merging operates on 2D segments, 3D connected objects, or all 3D objects in the volume, depending on the selected merge behavior.

The split tool offers three interaction modes. In boundary line splitting, the user simply draws a wide brush stroke over the membrane that represents the boundary. From the lines drawn, non-adjacent perimeter pixels are found and used as seed points for a watershed algorithm. This method is very fast to compute, and results can be adjusted interactively by adding or removing pixels from the boundary line. The remaining two modes are point-based split, which is similar to a live-wire segmentation where individual points are added to the split line; and region-based split, where seed regions are painted and a watershed boundary is found between them. Once one split has been performed, Mojo can predict how segments in adjacent slices will be split. User can navigate through the stack and quickly confirm or modify split operations while retaining 3D connectivity of the split objects.

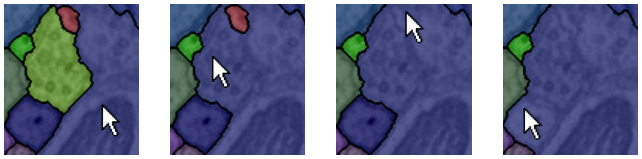
In addition to the merge and split tools, the adjust tool allows the user to manually draw a region and add it to the selected segment. This tool is useful when a combination of split and merge operations are required to correct a segment. Segments which have been fully corrected can be locked, which changes their appearance to a striped pattern and ensures that they will not be included in further actions.

4.1.3 Visualization

Mojo displays the original data in a single slice view and allows users to toggle the colored segmentation overlays. Boundaries of segmented structures can be enhanced by showing contours. Additionally, the user can zoom in and pan within a single slice or navigate between slices. When an object is selected, or the mouse hovers over an object, all parts of that object are highlighted to help the user navigate through the volume and to quickly identify all parts belonging to that object.

4.1.4 Data Handling

Mojo data is stored on the filesystem as a quadtree hierarchy of 2D image and segmentation tiles. Additional segment information, such as segment name and size, is stored in an sqlite database. To improve tile loading and processing times we maintain a tile index table to identify which tiles each segment id appears in. Additionally, we use a segment remap table to allow fast merge operations. Merging segment A with segment B can be achieved without having to modify the original segmentation tiles but by adding a look-up or redirection entry (e.g., A→B) into the segment remap table. The remap table is maintained in system memory on the GPU, so that stored segmentation tile ids can be loaded directly to the GPU and large volume changes can propagate quickly to the display. 2D merge, split and adjust operations modify segment tiles directly and the tile index table is updated accordingly.



(a) *select* **(b)** *click 1* **(c)** *click 2* **(d)** *click 3*

Fig. 7: ***Merge workflow.*** *Dojo merges segments via mouse clicks.*

4.2 Dojo - A Distributed Web-Based Approach

Dojo is a web-based proofreading tool consisting of an image server component written in python and an HTML5/JavaScript based user interface on the client side. This has the advantage that users can access the software by simply pointing their web-browser to the URL of the Dojo server. The main goal of Dojo's GUI and interaction design is to reduce complexity for non-expert users. Furthermore, Dojo was designed to be compatible with Mojo. Both tools use the same data structures, ensuring that annotated data from Mojo can be loaded into Dojo and vice-versa. The source code of Dojo and a demo installation are available at http://rhoana.org/dojo.

4.2.1 User Interface

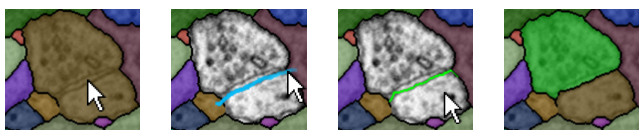
The graphical user interface of Dojo (Fig. 6) was designed with non-expert users in mind and aims to be minimalistic and clean. The 2D slice viewer uses the full window size while controls, information and help are moved to the corners to not disturb the data visualization and to provide a distraction-free environment. All textual information is kept small but still readable. The elements of the toolbox (i.e., split, merge, adjust) are presented as simple icons that show help tooltips upon hovering. Furthermore, to reduce interaction complexity, Dojo's proofreading tools are parameter-free and only require simple point-and-click mouse interaction. Additionally, all tools can be activated with keyboard shortcuts which are documented in the lower left corner for quick reference. The 3D volume rendering view is located in the upper right corner of the main window and can be resized as desired.

4.2.2 Proofreading Features

Dojo provides three proofreading tools, inspired by Mojo's toolbox described in Section 4.1.2. The key difference is that Dojo offers a single interaction mode for each tool, thus simplifying the interface.

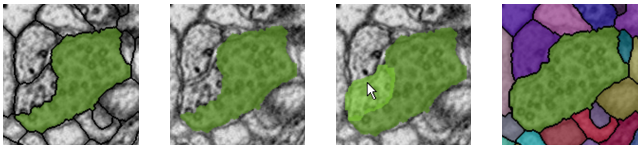
With the merge tool, the user clicks on the propagating structure and then on all segments which should be merged with it (Fig. 7). This action connects the segments across all slices (3D merge) and can also be used to merge segments which are located on different slices.

The split tool allows users to split a single segment into two or more segments by drawing a line across the segment that is to be split (Fig. 8). A split is confirmed by clicking in that part of the original



(a) *select* **(b)** *brush* **(c)** *confirm* **(d)** *result*

Fig. 8: ***Split workflow.*** *Users can split connected segments in Dojo by brushing over cell boundaries. The software then calculates the best split which can be accepted or discarded.*



(a) *select* **(b)** *no borders* **(c)** *paint* **(d)** *result*

Fig. 9: ***Adjust workflow.*** *Users can perform fine-grained pixel-wise adjustments by painting on the image data.*

object that should keep the original color. Under the hood, the splitting algorithm works differently to Mojo: Dojo uses all points of the segment to be split that are not part of the drawn line as seeds for a watershed algorithm, instead of only two perimeter pixels. Additionally, before computing the watershed algorithm, the original image data is blurred by a Gaussian and then contrast enhanced, which has experimentally been proven to generate better and more stable results.

The adjust tool works as in Mojo and lets users paint on the image data to extend a segmentation label (Fig. 9).

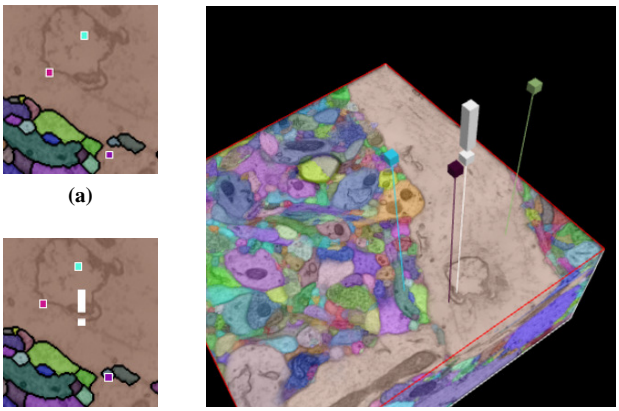
Merging uses the remap table data structure of Mojo and is computed solely on the client. Once a client adds or removes a merge, the merge table is sent to the server to keep all clients synchronized. No pixel data needs to be modified when merging segments. Splitting is performed on the server and does require pixel modifications. A split triggers a reload event for all clients to fetch new segmentation data for the specific slice. Users identify segmentation errors by constantly comparing the original image with the segmentation results. This tedious procedure results in a high mental workload for users but Dojo offers hotkeys for switching or adjusting opacity of the layers. Also, the volume rendering component of Dojo is useful for this task (Fig. 11). In the future, we want to integrate methods that provide user guidance by automatically identifying potential errors of such segmentations [37] and showing them to the user.

4.2.3 Visualization and Volume Rendering

In addition to the 2D slice view, Dojo provides full 3D volume rendering of the image stack based on WebGL. We leverage the X toolkit (XTK) for this purpose [18]. Since XTK is primarily used for medical imaging data sets which are smaller in size than EM data, we extended XTK to support 32 bit label overlays and raw image data of larger sizes. WebGL does not support 3D textures, therefore, volume rendering is based on 2D textures [9]. To circumvent memory and texture size limitations, the volume renderer limits the resolution of the loaded image slices to 512×512 for the xy-plane. This resolution is sufficient to display the 3D context of a structure and to gain a better spatial understanding of the data. The volume rendering can be activated by selecting an icon in the toolbox. Once active, the renderer displays the full image stack and segmentation volume or multiple selected segmentations. To enhance the users' orientation, the current slice position in the image stack is displayed as a red outline. Additionally, we have integrated collaborative features into the 3D view which are explained in more detail in Section 4.2.5 (Fig. 10c). Our volume renderer is built on top of WebGL and therefore requires a web browser with WebGL support. Nevertheless, recent advances in technology have brought WebGL to all major web browsers including most smartphones and tablets.

4.2.4 Data Handling - Large image data on the web

One key feature of Dojo is the support of large image and segmentation data. Transferring large amounts of image data between server and web client has previously been explored as part of several research projects [24, 41]. When designing Dojo we therefore evaluated different frameworks for interactively displaying large-scale images on the web. Most existing solutions are based on quadtree multi-resolution hierarchies that always load the currently requested resolution of the image data–low resolution levels for far-away and zoomed-out views and high resolution levels when the user zooms in. This approach



(a) **(b)** **(c)**

Fig. 10: ***Collaborative features of Dojo.*** *When active, the collaborative mode of Dojo facilitates proofreading the same data among multiple users. (a) 2D: cursors of other users are visible as colored squares if working on the same slice, (b) 2D: users can mark difficult regions with an exclamation mark and (c) 3D: cursors of all connected users and exclamation marks are visible as colored pins pointing to specific locations in the volume. Users can directly jump to that location by clicking on the pins.*

is well known and similar to Mojo's quadtree data structure. Unfortunately, most existing web-based image servers exhibit poor performance when loading different zoom levels so the user experience was not comparable to Mojo or other standalone software applications.

The only framework with a comparable performance was OpenSeaDragon [4]. OpenSeaDragon is based on the DeepZoom protocol and was initially developed by Microsoft but later open sourced as a community project. In our initial feasibility study we used an OpenSeaDragon viewer to connect to an IIP (Internet Imaging Protocol) image server to transfer images in a very performant way. The initial development stage of Dojo focused on using these technologies and it worked well for raw images. Unfortunately, there was no easy way of transferring segmentation data nor to display it as overlays. We initially added this functionality to OpenSeaDragon in close collaboration with their core developers, however, eventually had to abandon this path. While the basic integration of segmentation data worked, the viewer exhibited severe flickering artifacts and the image server did not support correct downsampling of segmentation masks. Another problem was the bit depth: our segmentation data can have up to 64 bit per pixel, which is not supported by the OpenSeaDragon framework. Based on these experiences we decided to a) develop our own client side image viewer and b) develop our own image server.

Using a custom web-client and image server let us optimize the transfer of images to our specific requirements. We now transfer high bit-per-pixel segmentation data directly as a zlib-compressed byte stream. This results in data sizes comparable to PNG encoding without any interpolation errors or loss of precision. Furthermore, the development of a custom image server and web client led to significant performance improvements. Zooming and scrolling through the image stack can now be done in real-time, even for large volumes.

4.2.5 Collaborative Features

Web pages inherently support connections from multiple clients (i.e., web browsers) and thus multiple users at the same time. By default, clients fetch information from and post information to the server. Usually, servers do not push information to all clients. Nevertheless, the latter can be achieved by using web sockets. We decided to use a web socket server in addition to standard HTTP to enable synchronization between all connected clients. Modifications of image data through merging, splitting and adjusting are sent to the server which then distributes them to each client. This can be a heavy operation depending on how many clients are connected. Thus, we limit the transfer to

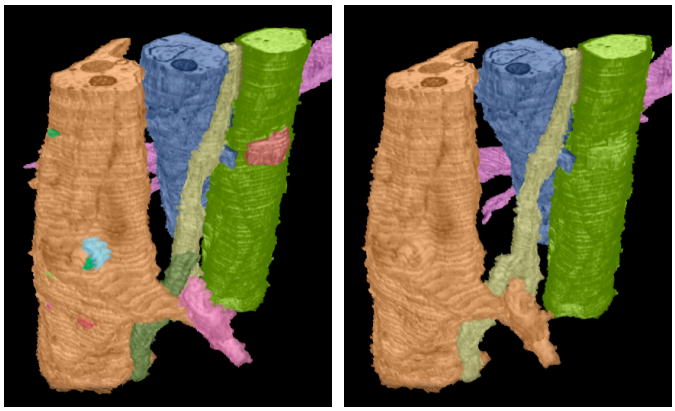


Fig. 11: ***Volume rendering of proofreading results in Dojo.*** *Prior to proofreading, incorrect segments are present (different colors). Right: After proofreading, neurons are smooth and uniformly colored.*

coordinates and meta information but never send binary data via web sockets. Additionally, each image operation gets stored as an entry in the activity log which is displayed on all connected clients, and located at the bottom right corner of the view.

Proofreading of larger volumes can be sped up considerably when multiple clients are working on the same volume. Hence, in addition to the synchronization of the current state of segmentation, we decided to add several collaborative features. Mouse cursors of all connected users are shared (Fig. 10). If two or more users work on the same tile in the image volume, the other users' cursors are shown as little colored squares. The colors are randomly assigned and each users' action is also colored in the activity log. If the 3D visualization is active, the cursors are also displayed as colored pins or needles pointing to a position of the volume. It is possible to click on a needle to jump to the position the other user is currently working on. In addition to cursor sharing, users can actively mark an area of the data to seek help from other users. When a user sets such a mark, all other users see a large exclamation mark in the 3D view and a small exclamation mark if they view the same slice. After navigating to that position and resolving the issue, it is possible for users to mark this problem as solved. Asking for help and marking an area as fixed is also reported in the activity log. Since these collaborative features can be distracting, they are optional and can be toggled by each individual client.

4.2.6 Limitations

Dojo has several limitations due to its minimalistic user interface and web-based nature (see Table 1): parameter-free operations do not allow complex operations as in Mojo and Raveler, the 3D view is less versatile than stand-alone GPU approaches due to memory restrictions resulting in downsampled textures, and the undo stack is limited.

5 User Study

We conducted a quantitative user study to evaluate the performance and usability of three proofreading tools. In particular, we evaluate how nearly untrained non-experts can correct an automatic segmentation using Mojo, Dojo and Raveler. We designed a between-subjects experiment and asked the participants to proofread a small dataset in a fixed time frame of 30 minutes. We used the most representative sub-volume of a larger freely available real-world data set where expert ground truth is available. We recruited participants from all fields with no experience in electron microscopy data or proofreading of such. As baseline, we asked two domain experts to label the same sub-volume from scratch using manual segmentation in the same fixed time frame.

5.1 Hypotheses

We proposed three hypotheses entering the study:

- **H.1 Proofreading will be measurably better with Dojo compared to other tools.** When presented with identical data, users of Dojo with no experience in proofreading and very short training, will perform significantly better (i.e., more accurately) than users of Mojo or Raveler.
- **H.2 Dojo's usability is higher than other tools.** Users of Dojo will report increased usability and that they like the system.
- **H.3 Given a fixed short time frame, proofreading by non-experts gives more correct results than completely manual annotations by experts.** In a fixed short time frame, inexperienced participants will generate a corrected data set which is more accurate and similar to the ground truth than a manual labeling from domain experts that was done in the same time frame.

5.2 Participants

Because we wanted to study how completely inexperienced users perform with the three proofreading tools, we recruited people of all occupations through flyers, mailing lists and personal interaction. Based on sample size calculation theory, we estimated the study sample size as ten users per proofreading tool including six potential dropouts [11, 21]. Nevertheless, all thirty participants completed the study ($N = 30$, 17 female; 20-40 years old, $M = 27$). Participants had no experience with electron microscopy data or proofreading of such and had never seen or used any of the three software tools. They received monetary compensation for their time.

5.3 Experimental Conditions

The three conditions in our study were the proofreading tools Mojo, Dojo and Raveler. Each participant proofread the same dataset in a time frame of 30 minutes. The requirement for participating was no experience in proofreading EM data. Since the three tools run on different platforms (except web-based Dojo which runs on all), we used three different machines with similar and standard, off-the-shelf hardware. Therefore, the only variable was the used software.

5.3.1 Dataset

We used a publicly available dataset of a mouse cortex (1024x1024x100 pixels) that was published for the ISBI 2013 challenge "SNEMI3D: 3D Segmentation of neurites in EM images". It was acquired with a serial section scanning electron microscope (ssSEM) with a resolution of 6x6x30 nm/pixel. We trained our automatic segmentation pipeline (Rhoana pipeline) on a similar dataset and used it to segment the data. Details of the segmentation pipeline are published in [28]. Manually labeled expert segmentation was available as a ground truth for the complete dataset. Since it is not feasible to let users proofread such a large volume, we cropped a sub-volume to 400x400x10 pixels. To find the most representative sub-volume (i.e., the sub-volume with the distribution of object sizes that is closest to the empirical mean distribution of object sizes in the volume) we calculated object size histograms, used them as features in a multi-dimensional vector space and chose the sub-volume with the minimal Euclidean distance to the centroid.

5.4 Procedure

Each study session started off with the participants signing the consent form and a basic demographic survey (age, sex, occupation, neuroscience background, scientific visualization background, familiarity with proofreading of segmentations and familiarity with EM data). Next, the participants were introduced to the Connectome project and its typical workflow (Fig. 2). Then, participants sat down at their randomly assigned study station which ran one of the three proofreading tools and the experimenter explained the tool. To demonstrate merging and splitting functionalities of each tool, a training dataset was loaded which was the second most representative sub-volume of the larger dataset with the same size of the test dataset.

There was no common region between the training data and the test data even though both were highly representative of the larger dataset. After explaining two merge and two split operations (average

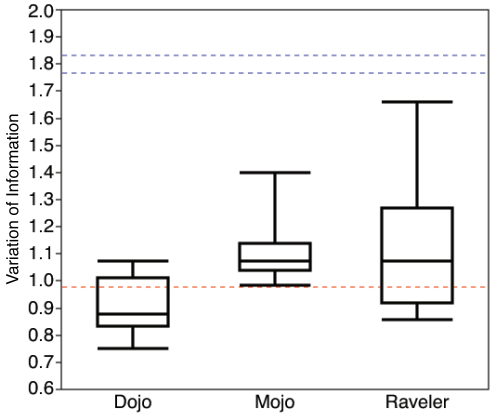


Fig. 12: ***Variation of information.*** *The VI similarity measure for each tool after proofreading for 30 minutes (lower is better). Participants using Dojo achieved a lower VI than subjects with the other tools. These results are statistically significant. The red line shows the VI of the automatic segmentation before proofreading. The blue lines show the VI of the experts' manual segmentation after 30 minutes.*

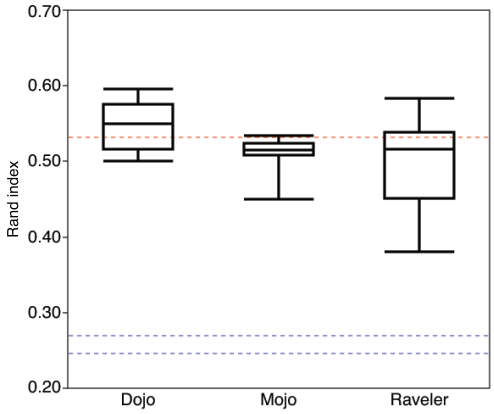


Fig. 13: ***Rand index.*** *The RI similarity measure for each tool after proofreading for 30 minutes (higher is better). Participants using Dojo achieved a higher RI than subjects with the other tools. These results are statistically significant. The red line shows the RI of the automatic segmentation before proofreading. The blue lines show the RI of the two experts' manual segmentation after 30 minutes.*

time about 5 minutes), the participants were asked to try the proofreading tool themselves for another 5 minutes. The experimenter then loaded the test dataset which was the same for each participant. For the next 30 minutes participants were asked to correct as many segmentation errors as possible but were warned that it was highly unlikely that they could finish proofreading the complete dataset in the given time frame. During the assessment, usage questions regarding the proofreading software were answered in a short manner.

After 30 minutes or if the participant decided to stop the experiment, the participants completed a qualitative questionnaire with ten questions regarding the software. Then, they answered the raw NASA-TLX standard questions for task evaluation [19]. At the end of the session, participants were presented their similarity scores in respect to the ground truth segmentation and could give general feedback and comments. The entire study session took approximately 60 minutes.

5.4.1 Expert Segmentations

To generate a baseline measure regarding hypothesis 3, we asked two experts to manually annotate the representative sub-volume from scratch. The experts used their software of choice, ITK-SNAP [49]. We set a time limit of 30 minutes and computed the similarity measures variation of information and Rand index as well as Edit Distance.

5.5 Design and Analysis

We used a single factor between-subject design with the factor *proofreading tool* (Mojo, Dojo, Raveler). The participants were randomly assigned to one of the three tools. From our group of participants ($N = 30$) we excluded two subjects (using Raveler and Mojo). One of the two subjects stated to be familiar with proofreading of EM data and for the other participant Mojo crashed after 20 minutes.

Our dependent measures were two measures of similarity between participants' results and the ground truth segmentation (manually labeled by experts, publicly available), the number of not performed merges and splits to fully correct the segmentation, as well as the participants' subjective responses recorded on a 7-point Likert scale.

Similarity was measured as *variation of information* (VI) (Fig. 12) and *Rand index* (RI) (Fig. 13) which are common benchmarks for clustering comparison in computer vision. VI is a measure of the distance between two clusterings, closely related to mutual information, but lower being better. Rand index is a measure of similarity, related to the accuracy, meaning higher scores are better. The number of not performed merges and splits to correct the segmentation is the *Edit Distance*, another common metric in computer vision. We calculated these measures and treated them as continuous variables. We analyzed these dependent variables using analysis of variance followed by parametric tests. For the subjective responses on Likert scales, we created sub-groups and performed ANOVA according to Holm's sequentially-rejective Bonferroni method [45] and parametric tests, if relevant.

6 Results and Discussion

The results of our user study demonstrate a modest advantage of Dojo regarding the quality of corrections. The difference in similarity was minor but statistically significant (Sec. 6.1). Another interesting but expected finding was that proofreading yielded better performance than manual segmentation of experts starting from scratch, when given the same short time frame (Sec. 6.3). Overall, participants reported better usability of Dojo compared to the other proofreading tools (Sec. 6.4), thus confirming our initial hypotheses.

6.1 Similarity

The initial segmentation for all participants was created using the Rhoana pipeline and had a VI of 0.98 and a RI of 0.53. The ten participants using Dojo had a mean VI of 0.90 ($SD = 0.10$). The nine participants using Mojo had a mean VI of 1.11 ($SD = 0.12$). The nine participants using Raveler had a mean VI of 1.12 ($SD = 0.25$). The ef-

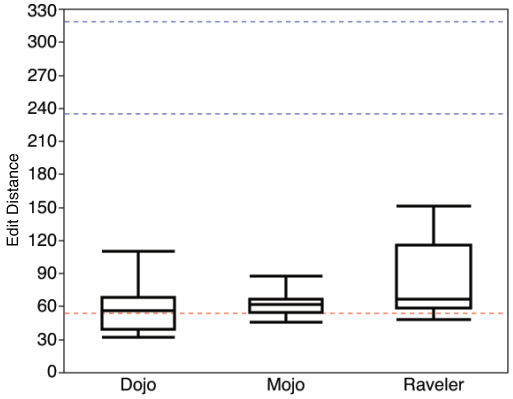


Fig. 14: ***Edit Distance.*** *The ED similarity measure for each tool after proofreading, in respect of the ground truth segmentation after 30 minutes (lower is better). On average, participants using Dojo reached a lower ED than subjects with the other tools. The red line shows the ED of the automatic segmentation before proofreading. The blue lines show the ED of the experts' manual segmentation after 30 minutes.*

fect of the proofreading tool, therefore, was significant, $F_{2,25} = 5.04$, $p = .015$. Post hoc comparisons (after Bonferroni correction) indicate that the mean VI for results with Dojo was significantly lower than for Mojo ($t_{25} = 2.06$, $p = .0411$) and also lower than the one for Raveler ($t_{25} = 2.06$, $p < .01$). Figure 12 shows this relation. We also analyzed the mean RI of participants using Dojo which was 0.55 ($SD = 0.03$). For participants using Mojo, the mean RI was 0.51 ($SD = 0.02$) and for participants who used Raveler, the mean RI was 0.5 ($SD = 0.06$). This difference was statistically significant, $F_{2,25} = 3.59$, $p = .043$. Further testing (after Bonferroni correction) showed, that only the difference between Dojo and Raveler was significant ($t_{25} = 2.06$, $p = .05$). The results are displayed in Figure 13.

6.2 Edit Distance

The edit distance metric (ED) reports how many operations have to be performed to reach the state of another labeled image. We calculated the ED for the proofread EM data as the sum of required 2D splits and 3D merges to reach the ground truth for each participant. These are the operations which can be performed by all three tools. The initial segmentation which was the input for all participants, had an ED of 54 (32 splits, 22 merges). A blank segmentation of our data had a ED value of 386, requiring 386 splits and 0 merges. The mean ED of participants for Dojo was 59.1 ($SD = 23.28$), for Mojo 62.9 ($SD = 12.03$) and for Raveler 83.22 ($SD = 37.03$). These results were not statistically significant but they follow the trend of better performance with Dojo. In fact, about half of the participants using Dojo were able to reduce the ED. Figure 14 shows this relation. Even though the improvement of the ED was not statistically relevant, the improvements of VI and RI allow us to **accept H1** and to confirm that complete novice users perform slightly better on a proofreading task using Dojo than with other tools. Interestingly, many participants were not able to improve the automatic segmentation but made it worse. In fact, only participants using Dojo were able to correct the segmentation on average. We believe that this is caused by several of factors:

1. **Three-dimensional thinking.** It is hard for untrained users to grasp the 3D structure of EM data. Dojo provides full 3D volume rendering to help users get a three dimensional intuition of the data, for single structures as well as for the whole EM stack.
2. **Difficulty identifying boundaries.** EM data can be very noisy and cell boundary detection needs to be practiced. From observations, it seemed that a large amount of time was spent by participants trying to identify boundaries.
3. **Time.** Participants tried to correct as much as possible in the given time frame. Even though they were told to only perform corrections when confident, they felt rushed. Therefore, we want to do a follow-up study without that fixed time frame.
4. **Usability of tools.** The usability of many existing proofreading tools is lacking. An overwhelming number of features and parameters are available and confuse users. Only Dojo provides parameter-free, easy-to-use tools.
5. **Almost zero training.** The participants of this study received, on purpose, nearly no training regarding the data or the software. We believe that training in the range of hours or days can drastically improve the performance of non-experts. Especially the manual detection of errors is very difficult for novices. Algorithms that guide the user to potential errors could greatly improve user performance.

6.3 Proofreading versus Manual Expert Segmentation

The two experts who were asked to manually label the data set from scratch in 30 minutes, did not reach the result of our proofreading participants starting from automatic segmentation (Figures 12, 13, 14). The mean values were VI=1.77, RI=0.26 and ED=277. These results were significant in comparison to Dojo ($t_{25} = 2.06$, $p < .0001$). This is no surprise since manual segmentation is very time consuming. Hence, we **accept H3**. In the given time frame experts were able to label 60.3% and 57.8% of the sub-volume. It is clear that given more time, the VI, RI and ED measures would improve rather quickly for manual expert segmentations.

6.4 Subjective Responses

All subjective responses were recorded on a 7 point Likert scale with *1=fully disagree* and *7=fully agree*. To ensure representative results, we grouped the questions and performed Holm's sequentially-rejective Bonferroni adjustment (N reported) before reporting statistic significance. We observed statistical significance for qualitative responses regarding usability: Participants stated that the tool is easy to use for Dojo $M = 4.5$ ($SD = 1.27$), Mojo $M = 4.11$ ($SD = 2.03$) and Raveler $M = 3.22$ ($SD = 1.09$) and that the tool is usable for Dojo $M = 5.1$ ($SD = 1.1$), for Mojo $M = 4.3$ ($SD = 1.87$) and for Raveler $M = 3.11$ ($SD = 1.27$). After adjustment with $N = 2$, being usable is statistically significant ($F_{2,25} = 4.57$, $p = .0408$) while further analysis only confirmed significance between Dojo and Raveler ($t_{25} = 2.06$, $p = .006$). We asked three questions regarding the visualization components: Participants stated that the slice visualizations were pleasing for Dojo $M = 5.8$ ($SD = 0.92$), Mojo $M = 4.7$ ($SD = 1.73$) and Raveler $M = 4.33$ ($SD = 1.58$), the segment visualizations were pleasing for Dojo $M = 5.5$ ($SD = 1.08$), Mojo $M = 5.11$ ($SD = 0.93$) and Raveler $M = 4.0$ ($SD = 1.5$) and additional information beside 2D was useful for Dojo $M = 5.0$ ($SD = 1.63$), Mojo $M = 4.0$ ($SD = 2.2$) and Raveler $M = 4.0$ ($SD = 1.63$). Unfortunately, none of these results were significant after adjustment (segment visualization was before). Nevertheless, we do believe that the 3D volume rendering of Dojo had impact on the superior quantitative performance reported in the previous section. Other interesting observations were that participants reported that the merge tool was easy to use: for Dojo $M = 5.8$ ($SD = 1.14$), Mojo $M = 4.89$ ($SD = 1.17$) and Raveler $M = 3.78$ ($SD = 1.39$). After adjustment with $N = 3$, this was statistically significant ($F_{2,25} = 6.37$, $p = .0174$). The NASA-TLX workload reported by the users did not yield any interesting results. As the overall conclusion, participants reported that they generally liked the used software for Dojo $M = 5.4$ ($SD = 1.17$), Mojo $M = 4.33$ ($SD = 1.66$) and Raveler $M = 3.67$ ($SD = 1.41$). This result was statistically significant with $N = 1$ ($F_{2,25} = 3.62$, $p = .0416$) but further analysis showed significance only between Dojo and Raveler ($t_{25} = 2.06$, $p = .0135$). Because of that and the usability findings, we **partially accept H2**. The qualitative as well as the quantitative evaluation was in favor of Dojo which also matches our observations during the user study.

7 Conclusion and Future Work

In this paper, we have presented an analysis and evaluation of proofreading tools for automatic segmentations of connectomics data. Based on this analysis and on our experience with Mojo, we developed Dojo, a web-based proofreading tool. Dojo provides several proofreading aids such as a clean and minimalistic user interface as well as 3D volume rendering. We performed a between-subjects user study regarding Dojo, Mojo and another proofreading tool called Raveler. The results of our quantitative evaluation confirm the need for easy-to-use and well-designed visualization features for proofreading tools but also show the need of user training regarding the proofreading task. The individual differences between the evaluated tools were not large due to study design limitations (see Section 6.2).

In the near future we will deploy Dojo to hundreds of high school students to proofread EM data in a collaborative fashion. Furthermore, we want to investigate in novel methods for simplifying cell boundary detection. Using interactive edge-detection to highlight boundaries could significantly improve the performance of non-expert users. We would also like to perform an in-depth user study without the time limitation to see how far proofreading can optimize faulty segmentations. Furthermore, we hope that offering an open source proofreading tool will promote the adoption of web-based scientific visualization and encourage more research in novel proofreading applications.

Acknowledgments

We would like to thank Steffen Kirchhoff for his input. This work was partially supported by NSF grant OIA-1125087, the NIMH Silvio Conte Center (P50MH094271) and NIH grant 5R01NS076467-04.

REFERENCES

[1] Rambo3d. http://openconnectome.github.io/Rambo3D/, 2012. Accessed on 31/03/2014.

[2] IEEE ISBI challenge: SNEMI3D - 3D segmentation of neurites in EM images. http://brainiac2.mit.edu/SNEMI3D, 2013. Accessed on 31/03/2014.

[3] Eyewire. http://eyewire.org/, 2014. Accessed on 31/03/2014.

[4] OpenSeaDragon. http://openseadragon.github.io/, 2014. Accessed on 31/03/2014.

[5] A. Akselrod-Ballin, D. Bock, R. Reid, and S. Warfield. Improved registration for large electron microscopy images. In *IEEE Int. Symp. on Biomedical Imaging (ISBI '09)*, pages 434–437, 2009.

[6] 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.

[7] J. Beyer, A. Al-Awami, N. Kasthuri, J. W. Lichtman, H. Pfister, and M. Hadwiger. ConnectomeExplorer: Query-guided visual analysis of large volumetric neuroscience data. *IEEE Trans. on Vis. and Computer Graphics (Proc. IEEE SciVis 2013)*, 19(12):2868–2877, 2013.

[8] J. Beyer, M. Hadwiger, A. Al-Awami, W.-K. Jeong, N. Kasthuri, J. W. Lichtman, and H. Pfister. Exploring the connectome: Petascale volume visualization of microscopy data streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, 2013.

[9] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *IEEE Symposium on Volume Visualization*, pages 91–98, 1994.

[10] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. J. Douglas. TrakEM2 software for neural circuit reconstruction. *PLoS ONE*, 7(6):e38011+, 2012.

[11] L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.

[12] J. C. Fiala. Reconstruct: A free editor for serial section microscopy. *Journal of Microscopy*, 218(1):52–61, 2005.

[13] S. Gerhard, A. Daducci, A. Lemkaddem, R. Meuli, J. Thiran, and P. Hagmann. The connectome viewer toolkit: An open source framework to manage, analyze, and visualize connectomes. *Frontiers in Neuroinformatics*, 5, 2011.

[14] D. Ginsburg, S. Gerhard, J. E. C. Calle, and R. Pienaar. Realtime visualization of the connectome in the browser using WebGL. *Frontiers in Neuroinformatics*, (95), 2011.

[15] 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.

[16] M. Hadwiger, J. Beyer, W.-K. Jeong, and H. Pfister. Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach. *IEEE Trans. on Visualization and Computer Graphics (Proceedings IEEE SciVis'12)*, 18(12):2285–2294, 2012.

[17] D. Haehn. Slice:drop: Collaborative medical imaging in the browser. In *ACM SIGGRAPH 2013 Comp. Anim. Festival*, pages 1–1, 2013.

[18] D. Haehn, N. Rannou, B. Ahtam, E. Grant, and R. Pienaar. Neuroimaging in the browser using the X Toolkit. *Frontiers in Neuroinformatics*, 2012.

[19] S. G. Hart and L. E. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139 – 183. 1988.

[20] M. Helmstädter, K. L. Briggman, and W. Denk. High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nature Neuroscience*, 14(8):1081–1088, 2011.

[21] W. Hwang and G. Salvendy. Number of people required for usability evaluation: The 10+-2 rule. *Commun. ACM*, 53(5):130–133, May 2010.

[22] 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 *Proceedings of IEEE CVPR 2010*, pages 2488–2495, 2010.

[23] Janelia Farm. Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2014. Accessed on 31/03/2014.

[24] A. Janke, H. Waxenegger, J. Ullmann, and G. Galloway. Tissuestack: A new way to view your imaging data. *Frontiers in Neuroinformatics*, (3).

[25] W. Jeong, J. Schneider, A. Hansen, M. Lee, S. G. Turney, B. E. Faulkner-Jones, J. L. Hecht, R. Najarian, E. Yee, J. W. Lichtman, and H. Pfister. A collaborative digital pathology system for multi-touch mobile and desktop computing platforms. *Comp. Graphics Forum*, 32(6):227–242, 2013.

[26] W.-K. Jeong, J. Beyer, M. Hadwiger, R. Blue, C. Law, A. Vázquez-Reina, C. Reid, J. W. Lichtman, and H. Pfister. SSECRETT and NeuroTrace: Interactive visualization and analysis tools for large-scale neuroscience datasets. *IEEE Computer Graphics and Applications*, 30(3):58–70, 2010.

[27] E. Jurrus, S. Watanabe, R. J. Giuly, A. R. Paiva, M. H. Ellisman, E. M. Jorgensen, and T. Tasdizen. Semi-automated neuron boundary detection and nonbranching process segmentation in electron microscopy images. *Neuroinformatics*, 11(1):5–29, 2013.

[28] V. Kaynig, A. Vázquez-Reina, S. Knowles-Barley, M. Roberts, T. Jones, N. Kasthuri, E. Miller, J. W. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. In *arXiv: 1303.7186 [q-bio.NC]*, 2013.

[29] Khronos Group. WebGL specification. http://www.khronos.org/registry/webgl/specs, 2014. Accessed on 31/03/2014.

[30] 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.

[31] A. Kuß, S. Prohaska, B. Meyer, J. Rybak, and H.-C. Hege. Ontology-based visualization of hierarchical neuroanatomical structures. In *Proceedings of Visual Computing for Biomedicine*, pages 177–184, 2008.

[32] J. W. Lichtman and W. Denk. The big and the small: Challenges of imaging the brain's circuits. *Science*, 334(6056):618–623, 2011.

[33] J. W. Lichtman, J. Livet, and J. R. Sanes. A technicolour approach to the connectome. *Nature reviews. Neuroscience*, 2008.

[34] 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.

[35] D. S. Margulies, J. Böttger, A. Watanabe, and K. J. Gorgolewski. Visualizing the human connectome. *NeuroImage*, 80(0):445 – 461, 2013.

[36] 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.

[37] 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.

[38] 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.

[39] H. Pfister, V. Kaynig, C. P. Botha, S. Bruckner, V. J. Dercksen, H.-C. Hege, and J. B. Roerdink. Visualization in connectomics. In *arXiv:1206.1428v2 [cs.GR]*, 2012.

[40] M. Roberts, W.-K. Jeong, A. Vázquez-Reina, M. Unger, H. Bischof, J. W. Lichtman, and H. Pfister. Neural process reconstruction from sparse user scribbles. In *Proceedings of MICCAI 2011*, pages 621–628, 2011.

[41] 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.

[42] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. *Bioinformatics*, 26(12):i57–i63, 2010.

[43] S. Seung. Reading the book of memory: Sparse sampling versus dense mapping of connectomes. *Neuron*, 62(1):17–29, 2009.

[44] S. Seung. *Connectome: How the Brain's Wiring Makes Us Who We Are*. Houghton Mifflin Harcourt, Feb. 2012.

[45] J. P. Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46(1):561–584, 1995.

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

[47] C. Sommer, C. Straehle, U. Kothe, and F. A. Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *IEEE Int. Symp. on Biomedical Imaging: From Nano to Macro*, pages 230–233, 2011.

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

[49] P. A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. C. Gee, and G. Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.

Guided Proofreading of Automatic Segmentations in Connectomics

Daniel Haehn[1] Verena Kaynig[1,2] James Tompkin[1] Jeff W. Lichtman[2] Hanspeter Pfister[1]

[1]Harvard Paulson School of Engineering and Applied Sciences
[2]Harvard Center for Brain Science
Cambridge, MA 02138, USA
haehn@seas.harvard.edu

Abstract

Automatic cell image segmentation methods in connectomics produce split *and* merge *errors, which require correction through proofreading. To aid error correction, we develop two classifiers to recommend candidate 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 classifier detects potentially-erroneous regions by considering a large context region around a segmentation boundary. With recommendations, proofreading of mouse cortex electron microscopy image segmentations can reduce variation of information scores on two different datasets from 0.48 to 0.40, which is an improvement on both pure automatic and pure manual proofreading.*

1. Introduction

In connectomics, neuroanatomists 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 $\approx 6\,nm$ per pixel to identify cells, synapses, and vesicles. For $25\,nm$ thick sections, a $1\,mm^3$ volume of brain contains 10^{15} voxels, or 1 petabyte of data. With so much data, manual annotation is infeasible, and automatic annotation methods are needed [14, 22, 25, 19].

Automatic annotation by segmentation and classification of brain tissue is challenging [1]. The state of the art uses supervised learning with convolutional neural networks [11], or potentially even unsupervised learning [9]. Typically, cell membranes are detected in 2D images, and the resulting region segmentation is grouped into geometrically-consistent cells across registered sections. Cells may also be segmented across registered sections in 3D directly. Using dynamic programming techniques [23] and a GPU cluster, these classifiers can segment ≈ 1 terabyte of data per hour [17], which is sufficient to keep up with the 2D data capture process on state-of-the-art electron microscopes (though 3D registration is still an expensive offline operation).

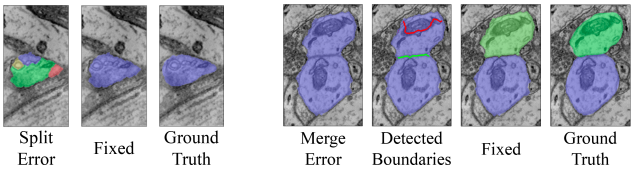


Figure 1: Split and merge error examples, their corrections, and their ground truths.

All automatic methods make errors, and we are left with large data which needs *proofreading* by humans. This crucial task serves two purposes: 1) to correct errors in the segmentation, and 2) to provide a large body of labeled data 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 [31, 15, 20, 13, 16, 32]. Many kinds of errors exist, such as inaccurate boundaries, but the most common are *split errors*, where a single cell is labeled as two, and *merge errors*, where two cells 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 [13]. However, even with semi-automatic correction tools, the visual inspection to find errors in the first place takes the majority of the time.

Our goal is to add automatic detection of split and merge errors to proofreading tools. Instead of the user visually inspecting the whole data volume carefully to spot errors, we design automatic classifiers that detect split and merge errors in 2D segmentations. Then, a proofreading tool can recommend regions with a high probability of an error to the user, and suggest corrections to accept or reject. We call

this process *Guided Proofreading*.

Given a membrane segmentation from a fast automatic method, our classifiers operate on the boundaries of whole cell regions. Compared to techniques that must analyze every input pixel, we reduce the data analysis to the boundaries only. This allows us to employ wider convolutional neural networks that take regional context and multiple input channels into account. One reason to classify errors on 2D images lies with the cost of 3D registration. This is often slow as it requires non-linear image alignment [4, 30]. However, typically segmentation results are local decisions at the cell level. In this case, 3D reconstruction is unnecessary and, instead of waiting for the 3D output, proofreading can start immediately to maximize error correction before cell grouping occurs across sections.

We quantitatively validate our approach with variation of information (VI) versus ground truth expert segmentations. While unintuitive, VI is a common metric to evaluate region segmentation performance [24]. We compare our approach to an existing proofreading tool that provides only manual error correction [13]. With a simulated user, our automatic error suggestions and corrections decrease VI from 0.4764 to 0.3996, which is in contrast to pure manual or pure automatic methods that can both *increase* VI. On a second larger dataset, our method decreases VI from 0.4847 to 0.3946. As a consequence, we are able to provide tools to proofread segmentations more efficiently, and so better tackle large volumes of connectomics imagery.

2. Related Work

Automatic Segmentation Multi-terabyte EM brain volumes require automatic segmentation [14, 22, 24, 25], 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.

NeuroProof [2] tries to decrease error rates with interactive learning of agglomeration of over-segmentations of images, based on a random forest classifier. Vazquez-Reina et al. [33] propose automatic 3D segmentation by taking whole EM volumes into account rather than a per section approach, then solving a fusion problem with a global context. Kaynig et al. [18] propose a random forest classifier coupled with an anisotropic smoothing prior in a conditional random field framework with 3D segment fusion. It is also possible to learn segmentation classification features directly from images with CNNs. Ronneberger et al. [28] use a contracting/expanding CNN path architecture to enable precise boundary localization with small amounts of training data. Lee et al. [21] recursively train very deep networks with 2D and 3D filters to detect boundaries.

These approaches make good progress; however, in general, proofreading is required to improve them through generating more ground-truth segmentations.

Arganda-Carreras et al. [7] posed the ISBI 2D EM segmentation challenge in 2012, where a 30-image corpus of fly cell 'in/out' labels was used to train boundary detection. However, mouse brain EM data is more difficult than the ISBI 2012 challenge, as it contains significant intercellular space which is hard to classify.

Bogovic et al. [9] learn 3D features, and show even that unsupervised learning can produce better features than hand-designs. We extend the features reported by Bogovic et al. but limit the classification to 2D images rather than 3D. In this case, reconstruction is unnecessary and, instead of waiting for the alignment of the 3D output, proofreading can start immediately to maximize throughput.

Collaborative Interactive Segmentation Recent works attack the problem of massive volume segmentation through crowd-sourcing[29, 6]. EyeWire [3] asks novice users to participate in a segmentation game with neuronal structures using a semi-automatic algorithm. D2P [12] uses a micro-labor workforce approach where local boolean decisions are combined to produce a consensus segmentation. In general, our goal is to correct the output of a segmentation which is thought to be good; hence, our tool would be used after learning a segmentation model to direct user attention to correct likely erroneous areas.

Proofreading Tools Janelia Farms identified the need for proofreading methods in 2010 [10]. The authors introduced Raveler [15], a software targeting expert users by offering many parameters for tweaking the proofreading process at the cost of a higher complexity. Raveler also includes a simple 3D renderer to validate corrections across slices.

The editing bottleneck for existing imperfect automatic segmentations was further specified by Peng et al. [26]. The authors developed software which supports three-dimensional proofreading and envision crowdsourcing to tackle large amounts of data.

EyeWire [3] exists since 2012 and is a popular online platform where users annotate retinal neurons. The platform is set up as an online game and participants earn virtual rewards for merging superpixels to reconstruct the data.

To simplify proofreading of small volumes, Sicat et al. proposed a graph abstraction method to guide users to problematic regions indicating the need for corrections [31].

Later, Plaza suggested a focused proofreading approach using heuristics to concentrate on certain regions of image data [27]. The main driver of his approach is the size of the segment — ignoring very small regions. By doing so, Plaza limits user decisions to yes/no questions and suggests crowdsourcing as a potential platform for proofreading.

In 2014, Haehn et al. developed two proofreading tools: Mojo and Dojo. Mojo provides a simple scribble interface

for error correction, and Dojo extends this for distributed proofreading via a minimalistic web-based user interface. The authors defined requirements for general proofreading tools, and then evaluated the accuracy and speed of Raveler, Mojo, and Dojo through a quantitative user study (Sec. 3 and 4) [13]. Al-Awami et al. integrated Dojo into the Neuroblocks proofreading management system [5], which tracks and visualizes progress among multiple users.

In 2015, Karimov et al. proposed a guided volume editing approach using histogram dissimilarity [16]. Measuring differences in histogram distributions helps to find potential errors and to suggest possible corrections. These promising results on Computed-Tomography Angiography datasets are targeted towards expert users.

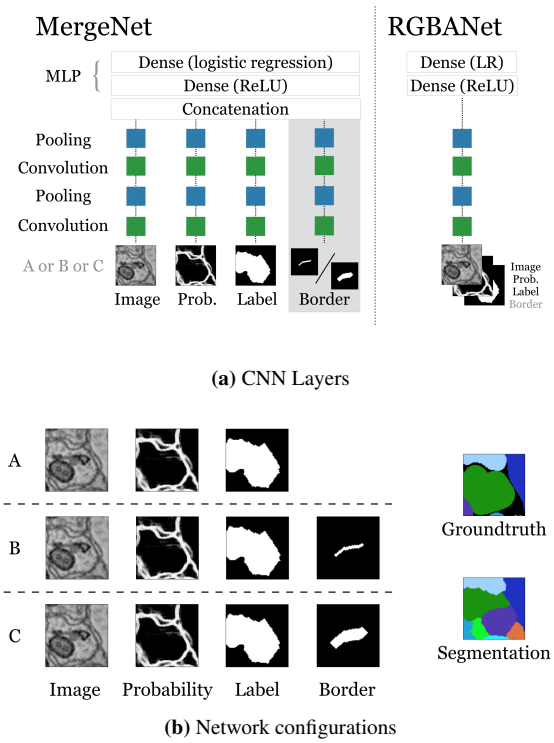
Recently, Uzunbas et al. showed that potential labeling errors can be found by considering the merge tree of an automatic segmentation method [32]. The authors track uncertainty through automatic labeling and then present potential regions for proofreading. This requires access to the segmentation merge tree, which is not always available.

However, none of these methods leverage machine learning to enhance the proofreading performance.

3. Method

We build a split error classifier using a convolutional neural network (CNN) to check the boundaries of an existing automatic segmentation. For each boundary, the CNN provides a probability that points sampled along the boundary have caused a split error. For each boundary, we sample up to 10 decision points, where the decision points are spread evenly over the boundary length, given that their context windows do not overlap. These probabilities are then weighted by the length of the boundary within the context over the total boundary length, and averaged. A greedy algorithm then merges neighboring regions sequentially, starting with the highest probability score. Following each merge, neighboring boundaries are re-evaluated for split errors. Correcting a split error is as simple as merging the two bordering labels.

Identification and correction of merge errors is more challenging, 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. For each segmentation label, we generate 30 potential boundaries through the region by placing watershed seed points at opposite sides of the label boundary and generating the corresponding split. Then, we check to see whether any potential edge is classified as a split error. If the CNN detects a boundary with a very low split error score, then the boundary should have been in the segmentation and the region is a candidate for a merge error.



(a) CNN Layers

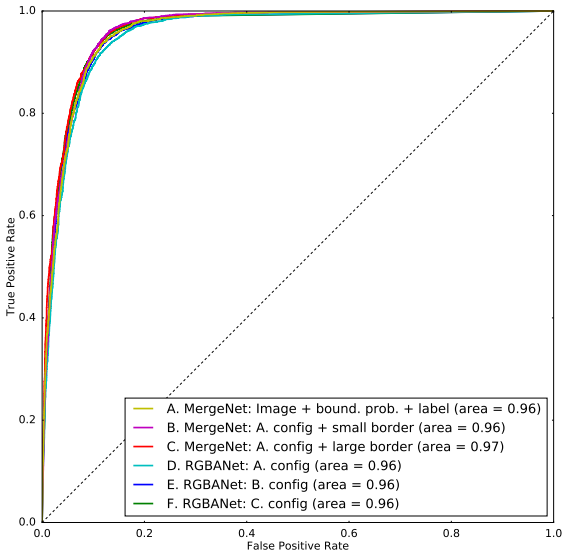
(b) Network configurations

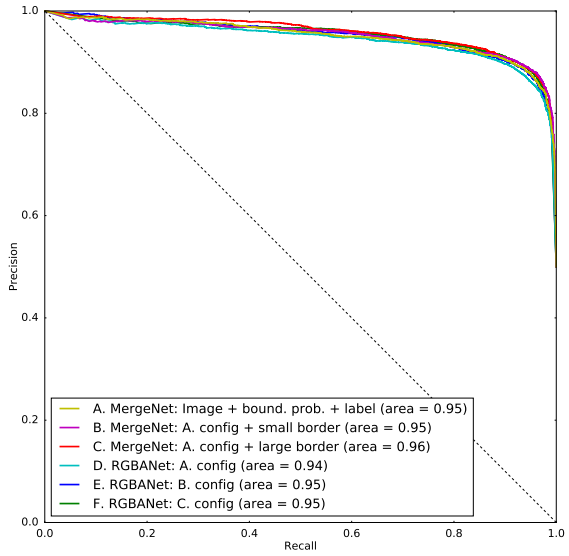
Figure 2: (a) Our network architecture with up to four input channels, each with two convolutional and two pooling layers. MergeNet includes four separate input channels and RGBANet uses a 4-channel input. (b) We trained three different network configurations with three and four inputs: A) image, boundary map probability, and merged binary mask; B) A configuration extended with a small border mask, to focus on the specific boundary in question; C) A configuration extended with a large border mask.

3.1. Convolutional Neural Network Design

To train a CNN for split error detection, we take multiple channels of boundary context information into consideration for the decision making process. We pass multiple inputs into the CNN windowed around a particular decision pixel: the input grayscale image patch, the corresponding boundary probability map patch, and two corresponding binary mask patches for the segmented regions at either side of the boundary. Following Bogovic et al. [9], these two masks can be combined into a single mask with comparable performance (configuration A, Fig. 2b). The network then leverages these multiple input patches to identify and correct errors made by the previous membrane detection network and automatic segmentation pipeline.

One way to combine these inputs is to treat them as a 4-channel input (‘*RGBANet*’), so that alignment between the input image and the segmentation masks is not lost throughout the convolutions. However, training a boundary-classifying network can be difficult due to rigid ground-





	Network	Validation loss	Test acc.	Prec./Recall	F1 Score
A.	MergeNet: Image + boundary prob. + seg. label	0.073	0.906	0.906/0.906	0.907
B.	MergeNet: A config. + small border overlap ($d = 1$)	0.070	0.911	0.911/0.911	0.912
C.	MergeNet: A config. + large border overlap ($d = 5$)	0.070	0.908	0.908/0.908	0.909
D.	RGBANet: A config.	0.058	0.895	0.895/0.895	0.894
E.	RGBANet: B config.	0.054	0.907	0.907/0.907	0.908
F.	RGBANet: C config.	0.058	0.905	0.905/0.905	0.904

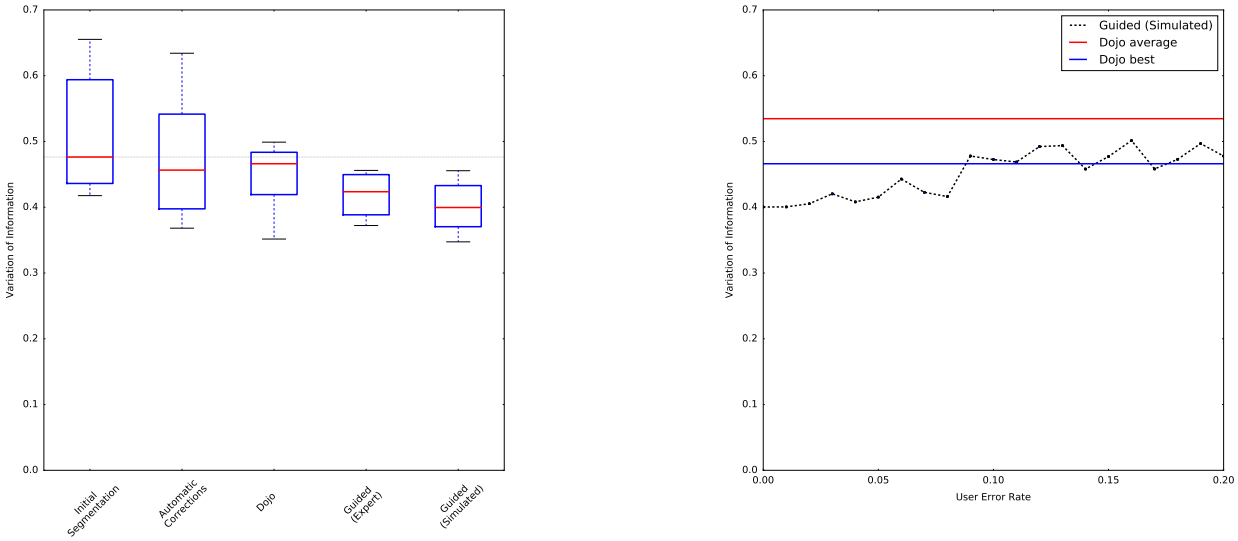
Figure 3: Network design training evaluation. Adding an extra channel containing a binary mask of just the border slightly increases performance in both network configurations. Due to slightly larger areas under the curves for both receiver operating characteristic and precision/recall, we choose configuration MergeNet C to evaluate VI against human performance.

truth segmentations, which often differ substantially from automatic segmentation regions in ambiguous extra-cellular space. To cope with this variation, our network is based on multiple separate input channels ('*MergeNet*', Fig. 2a). Each of the input patches is connected individually to a 2-layer network, with each layer consisting of convolutional and pooling layers. The output of these networks is then combined by a fully-connected multi-layer perceptron (MLP) with one hidden layer and a two-class logistic regression output layer. The intuition for this multiple input channel approach is that we want to allow variation in the input and masks independently to accommodate potential error, and then for the hidden layers to discover appropriate combinations of the relevant features learned separately for the different input channels.

To better direct both networks to train on the true boundary edge, which in many cases is missing from the boundary probability map and hence is the cause of merge errors, we additionally pass as input a second binary mask. This mask contains the true boundary edge (configuration B, Fig. 2b). To consider slight edge ambiguities, we also test a version of this network where the true boundary mask has been dilated by 5 pixels (configuration C).

3.2. Training

To train the network, we use the blue 3-cylinder mouse cortex volume of Kasthuri et al. [17] ($2048 \times 2048 \times 300$ voxels). The tissue is dense mammalian neuropil from layers 4 and 5 of the S1 primary somatosensory cortex of a healthy mouse. The resolution of our dataset is $3\,nm$ per pixel, and the section thickness is $30\,nm$. A manually-labeled expert segmentation is available as ground truth for the entire dataset. We use the first 250 sections of the data for training and validation (split 0.25) and the last 50 for testing. To generate training data, we identify correct regions and split errors in the automatic segmentation by intersection with ground truth regions. From these regions, we sample 266,088 correct regions and 266,088 split error



(a) Interactive (Dojo) vs. Guided Proofreading

(b) Simulated User Error Rate

Figure 4: (a) We compare distributions of VI measures across 10 sections for five cases: the initial automatic segmentation, a fully-automatic correction of recommended errors based on a threshold of acceptance, the best user in the Haehn et al. experiment with Dojo, two experts using our system, and our simulated user. Lower scores are better. (b) We test our simulated user with different error rates and compare against the best user as well as the average user performance in Dojo.

patches. We define patches of size 75×75 to cover approximately 80% of all boundaries in our segmentation output.

We train our networks using the following parameters: learning rate $lr = 0.03$ (iteratively decreasing until $lr = 0.00001$), momentum $m = 0.9$ (iteratively increasing until $m = 0.999$), filter size $fs = 13 \times 13$, and number of filters $fn = 16$ for MergeNet ($fn_1 = 64, fn_2 = 48$ for RGBANet). This results in approximately 1.5 million parameters for all network configurations. For regularization, we use dropout layers after each pooling layer with $p = 0.2$. We assume that the training has converged if the validation loss does not decrease for 50 epochs. The network is specified using the deep learning libraries Lasagne and Theano [8], and trained on a Tesla K40m graphics card.

Figure 3 presents validation loss function scores (cross validation), test accuracy percentages, precision/recall scores, and F1 scores. We also show receiver operating characteristics and precision/recall curves. We observe a slightly increased performance for networks using the MergeNet configuration. From this, we select configuration MergeNet C to evaluate against human performance in a VI improvement experiment.

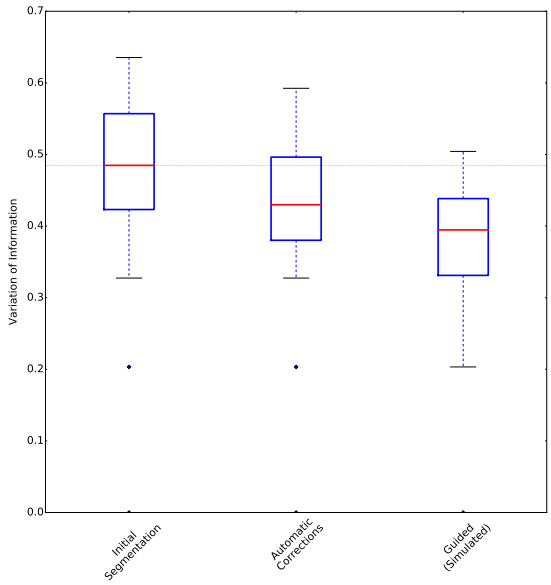
4. Evaluation

We evaluate our split and merge error detection and correction recommendation in the context of interactive proofreading tools: to direct users to regions with a high probability of error and to suggest corrections (Fig. 4). For comparison, we take publicly available mouse cortex data of the same kind as our training data. We perform two experiments: first, we compare our approach with previously reported proofreading results using real-world measurements; second, we evaluate guided proofreading in a simulated context with a much larger dataset.

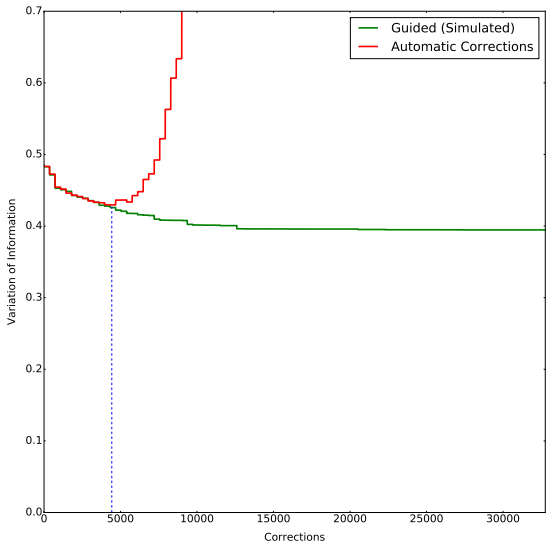
4.1. Comparison Study

For our first experiment, our dataset is part of the ISBI 2013 challenge training dataset ($1024 \times 1024 \times 100$ voxels) which was acquired using a serial section scanning electron microscope (ssSEM) with a resolution of $6 \times 6 \times 30\,nm$ per voxel. We use the available manually-labeled ground truth to score our approach using the variation of information (VI) metric, which is closely related to mutual information. VI is a measure of the distance between two clusterings, where lower VI numbers are better. Since our classifiers are trained on 2D image slices, we perform all evaluations on slices rather than 3D volumes.

Guided proofreading. Recently, Haehn et al. discussed requirements for interactive proofreading and evaluated three different connectomics tools in a study with naive users [13]. This study asked users to spend 30 minutes



(a) Guided Proofreading Results



(b) Variation of Information per Correction

Figure 5: Results of our simulated experiment on a larger dataset ($2048 \times 2048 \times 50$ voxels). Lower VI scores are better. (a) The initial segmentation is reduced by both fully-automatic correction (probability threshold $p_t = 0.95$) and our simulated user. (b) We compare the change in VI measure for each individual correction running automatically and user-driven. The dashed blue line indicates the moment p_t is reached. Guided proofreading is able to reduce VI beyond this threshold.

proofreading split and merge errors with the different tools to improve upon the automatic segmentation. The best performing tool in their evaluation was Dojo. We use their findings and their user-generated proofreading result data, which they kindly provided, as a baseline for the evaluation of our method. Haehn et al. performed their user study on the most representative sub-volume ($400 \times 400 \times 10$ voxels) in terms of object size distribution. For comparison, we use the same data and time constraints. We asked two experts to perform the proofreading task using our system (Fig. 6).

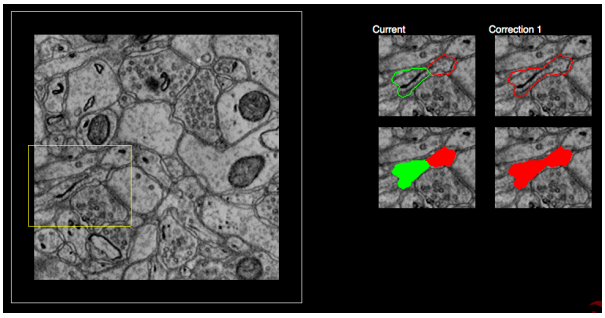
In addition, we simulate a user to provide insight into achievable performance. We assume that all classification has been computed ahead of time, and that the user is presented with a stream of error corrections to assess. The assessment is simulated by comparing the VI before and after each recommendation, and only accepting corrections when VI reduces. We test this across different user error rates (Fig. 4). In Haehn et al., the proofreading time was 30 minutes, in which human participants performed 59 corrections on average ($\approx$ 30 seconds per correction). In our scenario, and unlike in Haehn et al., users do not need to visually find errors and manually correct them. From real-world user performance, we observed an average decision time of 3.2 seconds. Thus, for our simulated user, we assume each correction assessment takes 5 seconds (360 assessments in 30 minutes). Split errors are likely to take less time than this; however, merge errors are harder to assess, as the user must select between the top 5 candidate boundaries. Since the performance of Haehn et al.'s human participants showed large variation (average VI improvement: -0.0582), we present the best performing Dojo user (VI improvement: 0.0102) as our baseline. The average VI improvement of two experts using our new system is 0.0528. For our simulated user, the VI improvement is 0.0768 (Fig. 4). In total, our classifier predicted 18 merge and 842 split errors.

Random recommendations. We test a classifier with random performance in comparison to our learned CNN. For split errors, the simulated user is presented with randomly picked boundaries, which they can accept or reject. For merge errors, the simulated user is presented with 5 randomly selected boundaries from the interior of the segmented region. Around 80% of all presented regions did not need to be corrected. Hence, the median VI did not decrease much (VI improvement: 0.0011). The significantly worse performance of this approach demonstrates that our network is informative to the user.

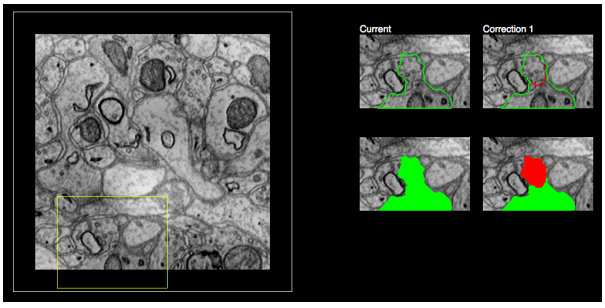
Automatic correction. As a comparison, we also perform automatic correction. During training, we define a probability threshold $p_t = 0.95$ for automatic split correction based on CNN probability from the test set. Then, for automatic correction, we apply both classifiers to produce lists of split and merge errors sorted by confidence. First, we correct merge errors with $\max(1 - p)$, followed by split error correction using p_t. The total time for correcting all errors was 17 minutes on a 3.2 GHz Quad-core Intel Xeon with an NVIDIA GeForce Titan (merge error correction 15min, split error correction 2min). The median VI improvement vs. ground truth was 0.02 (Fig. 4). This is not surprising, as the problem is very challenging, and this motivates the need for human-in-the-loop proofreading tools.

4.2. Simulated Experiment

For our second experiment, we proofread 50 slices of the blue 3-cylinder cortex volume of Kasthuri et al. [17]. The data was not seen by the network before and includes 2048 x 2048 x 50 voxels with a total number of 33,076 labeled objects. Since interactive proofreading of such a large dataset would consume a significant amount of time, we restrict our experiment to a simulated user and to automatic corrections. Similar to our comparison study (Sec. 4.1), the simulated user assesses a stream of errors by comparing VI before and after each performed correction. As before, corrections are only accepted when VI reduces. In contrast to our time limit in the comparison study, the simulated user proofreads until all objects in the volume were assessed. For automatic corrections, we use our defined probability threshold $p_t = 0.95$. Based on a time budget of 5 seconds per correction, the proofreading process for a real user would in theory take over 45 hours for this dataset. The total time for automatically correcting all errors was $\approx$ 6 hours on a 3.2 GHz Quad-core Intel Xeon with an NVIDIA Geforce Titan. The total time for the simulated user including the VI calculations after each assessment was $\approx$ 10 hours. Both approaches significantly reduce the VI in comparison to the initial segmentation by 0.0901 for the simulated user and by 0.0549 automatically (Fig. 5).



(a) Split error



(b) Merge error

Figure 6: Our web-based user interface includes a slice overview with the relevant area highlighted in yellow. The interface shows (a) a split error with a suggested correction as well as (b) a merge error with correction. The user selects whether to accept a correction or to discard it.

5. Application

In our experiments, we observed the best performance using a combination of user guidance and our trained network. In contrast to fully interactive proofreading tools like Dojo, Mojo, and Raveler, our system requires only minimal user input. We distinguish between merge and split errors and provide a very simple user interface to correct them (Fig. 6). The system shows only one potential error in the interface: either a potential false merge or split. In the case of merge errors, the user sees the five highest-scoring possible boundaries as overlays on the corresponding grayscale image, and can also draw a new boundary interactively. The user then chooses one of the suggestions, draws a boundary, or marks the cell as correct. For split errors, the system shows the grayscale image and a possible border, and the user marks whether the cell is correct. Our Dojo user study experiment baseline was limited to 30 minutes, and participants performed 59 corrections in average ($\approx$ 30 seconds per correction). Our experiments suggest that even non-experts can perform a correction using our system in < 5 seconds, resulting in increased proofreading throughput.

6. Discussion and Conclusion

Automatic cell boundary segmentation is difficult, and trying to improve such segmentations automatically as a post-process through split and error correction is, in principle, no different than trying to improve the underlying cell boundary segmentation. This is shown by the approximately equivalent VI distributions of the initial segmentation and our automatic segmentation correction (Fig. 4). Due to the task difficulty, manual proofreading of connectomics segmentations is necessary, but it is time consuming and error prone, as can be seen from the Dojo human trials: on average, participants made the segmentations worse. However, there is value in being able to recommend to users

possible regions for correction, as the time cost of proofreading is dominated by the visual search for errors.

We have addressed this problem through training a CNN to detect ambiguous regions from labeled data—in effect, (re-)learning a confidence measure on boundaries. This allows us to identify split and merge errors, and also to recommend their corrections, which is an improvement over existing systems which just provide semi-automatic merge error correction. Our experiments have shown that guided proofreading has the potential to reduce VI over existing interactive proofreading tools. This helps reduce the proofreading bottleneck in the analysis of large connectomics datasets. To encourage testing of our proposed architecture on more data, we provide the trained networks and classifier code as free and open source software at (link omitted for review).

References

[1] IEEE ISBI challenge: SNEMI3D - 3D segmentation of neurites in EM images. http://brainiac2.mit.edu/SNEMI3D, 2013. Accessed on 31/03/2016. 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] Eyewire. http://eyewire.org/, 2014. Accessed on 31/03/2014. 2

[4] A. Akselrod-Ballin, D. Bock, R. Reid, and S. Warfield. Improved registration for large electron microscopy images. In *IEEE Int. Symp. Biomedical Imaging (ISBI)*, 2009. 2

[5] 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. 3

[6] 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

[7] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9(142), 2015. 2

[8] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 5

[9] J. A. Bogovic, G. B. Huang, and V. Jain. Learned versus hand-designed feature representations for 3d agglomeration. *CoRR*, abs/1312.6159, 2013. 1, 2, 3

[10] 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

[11] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, 2012. 1

[12] 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

[13] 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, 3, 5

[14] 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

[15] Janelia Farm. Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2014. Accessed on 31/03/2016. 1, 2

[16] 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. 1, 3

[17] 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. 1, 4, 7

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] J. Masci, A. Giusti, D. C. Ciresan, G. Fricout, and J. Schmidhuber. A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *ICIP*, 2013. 1

[24] 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

[25] 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

[26] 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. 2

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

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

[29] 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

[30] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. *Bioinformatics*, 26(12):i57–i63, 2010. 2

[31] 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

[32] 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. 1, 3

[33] 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