Boundary Learning by Optimization with Topological Constraints
Supplementary Material

Viren Jain[4,*], Benjamin Bollmann[4,*], Mark Richardson[4], Daniel R. Berger[4,5], Moritz N. Helmstaedter[3], Kevin L. Briggman[3], Winfried Denk[3], Jared B. Bowden[2], John M. Mendenhall[2], Wickliffe C. Abraham[6], Kristen M. Harris[2,†], Narayanan Kasthuri[1], Ken J. Hayworth[1], Richard Schalek[1], Juan Carlos Tapia[1], Jeff W. Lichtman[1] and H. Sebastian Seung[4,5]

[1]Department of Molecular and Cellular Biology, Center for Brain Science, Harvard University, Cambridge, MA, USA

[2]Center for Learning and Memory, Department of Neurobiology, University of Texas at Austin, TX, USA

[3]Department of Biomedical Optics, Max Planck Institute for Medical Research, Heidelberg, Germany

[4]Brain & Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA

[5]Howard Hughes Medical Institute, Cambridge, MA, USA

[6]Dept. of Psychology and Brain Health and Repair Research Center, Univ. of Otago, Dunedin, New Zealand

1. Details of Experimental Procedures

In this section we provide additional details of the experimental comparisons that were performed in Section 4 of the main text. We also show an extended presentation of the warping error results shown in the main text. In particular Figure 1 shows the warping error on the test set of the convolutional network methods along with BEL and *gPb*-OWT-UCM. For this comparison, a threshold of *gPb*-OWT-UCM and BEL was chosen according to the threshold that achieved lowest Rand error also on the test set (shown in Figure 4 of the main text). These results are consistent with the relative ordering of algorithms that the Rand index produced, but the relative reduction in error between the methods is larger (for example, the *gPb*-OWT-UCM method has almost ten times as much warping error as the highest performer, BLOTC CN).

Figure 2 also shows a visual depiction of the segmentation and boundary maps of all methods that are discussed.

1.1. Multiscale Normalized Cut

Multiscale normalized cut was performed using publicly available code provided by the authors of [1]: `http://www.seas.upenn.edu/~timothee/software/ncut_multiscale/ncut_multiscale.html`

This technique requires that the number of objects in the image be specified by the user. We are interested in completely automated segmentation in which such information would not usually be available. Therefore we provided to the code the average number of objects in a training set image. However, we also tried providing the code with the true number of objects in each test set image. Then the results of "Multiscale Ncut" shown in Figure 4 of the main text improve slightly, but remain worse than all other techniques.

1.2. *gPb*-OWT-UCM

The *gPb*-OWT-UCM algorithm (global probability of boundary followed by the oriented watershed transform and a hierarchical region construction by ultrametric contour maps) was performed using publicly available code provided by the authors of [2, 3]: `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/gpb/grouping.zip`

The following is a summary of the algorithm implemented by the code, as described in [2]. First, the *gPb* contour detector was applied directly to the raw EM images, producing an 8-channel oriented localized probability of boundary map, as well as a single-channel thinned contour image which is the result shown in Figure 2 of the main text. The 8-channel boundary map was then converted to an oversegmentation using the oriented watershed transform (OWT), and then an ultrametric contour map (UCM) according to the dissimilarity between regions as determined by mean probability of boundary value. Finally, the ultrametric contour map was partitioned using connected components at various thresholds (where the x-axis in Figure 4 for this technique corresponds a segmentation of the UCM thresholded at a value of $255 * (1 - x)$, to compensate for the range of the UCM, and then the binary boundary map is converted to an in/out map prior to connected components by flipping the binary values).

1.3. Boosted Edge Learning

The Boosted Edge Learning algorithm was applied to our training set of EM images using publicly available code provided by the authors of [4]: `http://www.loni.ucla.edu/~ztu/Download.htm`

*These authors made comparable contributions to this paper.

†Supported by NIH/NINDS R37 NS021184 and NIH/NIBIB EB002170.

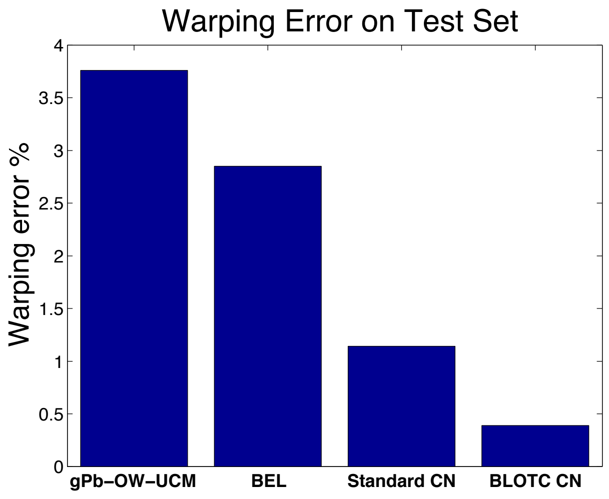

Figure 1. Warping error on test set.

A probabilistic boosting tree of depth 10 was used with 120 weak classifiers in each AdaBoost node. A patch size of 30 was used, which is a slightly larger than the field of view of the convolutional networks that were used. These were the default parameters provided in the code.

After training was complete, we proceeded to quantify segmentation performance on the test set of images. Here we had to choose our own method for generating segmentations from BEL output. The simple procedure of finding connected components of the thresholded BEL output (which was used for the CNs and *gPb*-OWT-UCM) produced a poor Rand error, as did the watershed transform on the negative of the BEL output. Therefore we applied the watershed transform only after using MATLAB's `imimposemin` command to damp local maxima of the BEL output, constraining them to occur where the BEL output was greater than a threshold. The watershed transform was performed with 8 connectivity (4 connectivity gave worse results).

1.4. Convolutional Networks

A convolutional network is an alternating sequence of linear filtering and nonlinear transformation operations. The input and output layers include one or more images, while intermediate layers contain "hidden" units with images called feature maps that are the internal computations of the algorithm. The activity of feature map a in layer k is given by

$$I_{k,a} = f\left(\sum_b w_{k,ab} \otimes I_{k-1,b} - \theta_{k,a}\right) \tag{1}$$

where $I_{k-1,b}$ are feature maps that provide input to $I_{k,a}$, and $\otimes$ denotes the convolution operation. The function f is the sigmoid $f(x) = 1/\left(1 + e^{-x}\right)$ and $\theta_{k,a}$ is a bias parameter. Gradient learning of this architecture can be performed with an version of the backpropagation algorithm [5, 6].

Our experiments were performed on the gray scale EM images and hence the networks contain a single image in the input layer. It is straightforward to extend this approach to color images by assuming an input layer with multiple images (e.g., RGB color channels). For numerical reasons, it is preferable to use input and target values in the range of 0 to 1, and hence the 8-bit integer intensity values of the dataset (values from 0 to 255) were normalized to lie between 0 and 1.

As our loss function during optimization of the convolutional network, we used the square-square loss

$$l(x, \hat{x}) = x \max(0, 1 - \hat{x} - m)^2 + (1 - x) \max(0, \hat{x} - m)^2$$

with $m = 0.2$, rather than the squared loss $(x - \hat{x})^2$. This loss function is a better approximation of the true binary classification error we seek to optimize. This is particularly important in the scenario in which the classifier has predicted the correct class of most examples (based on thresholding the analog output), and there are relatively few remaining incorrectly classified examples. In this case, the small amount of remaining error in the squared loss related to pushing values to 0 or 1 may dominate the error associated with incorrectly classified examples. The square-square loss allows the training to "give up" when the classifier output is correct by a sufficient margin. This gives the classifier more flexibility to achieve higher binary classification accuracy.

Batch learning is inefficient in this context, as the training set has millions of pixels and it is not practical to compute the gradient with respect to the entire training set for each update, particularly when many hundreds of thousands of updates may be required in order to reach convergence. Therefore we adapted stochastic online learning to this problem. We employed a minibatch implementation in which a randomly chosen 14×14 patch of the network output was used to compute each gradient update. A localized patch shares computation in a convolutional network and is therefore especially efficient to compute.

Segmentations were generated by thresholding the analog output of the convolutional network and then performing connected components with the same $\kappa = 4$ and $\bar{\kappa} = 8$ adjacency using in warping.

2. Classification of non-simple points

As discussed in Section 4 of the main text, in our experiments with BLOTC learning we allowed the warping to flip certain non-simple points in addition to any simple point. In particular, the warping was allowed to create holes within objects, therefore not penalizing the computer for detecting internal boundaries even if they were not traced by the human. The warping was also allowed to create new objects,

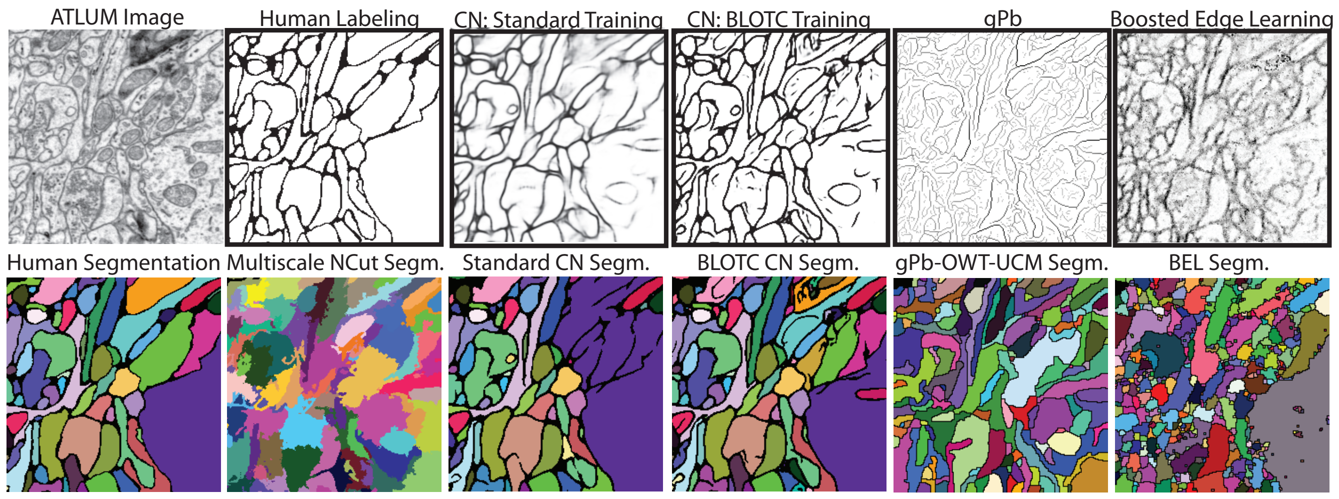

Figure 2. Comparison of boundary detector outputs and corresponding segmentations on an image from the test set. Both Standard and BLOTC CN segmentations were generated by connected components on the respective boundary detector outputs at threshold 0.75. For gPb and BEL, segmentations were generated at the optimal threshold according to the Rand error. Multiscale NCut directly generates a segmentation and thus no threshold was used (in the results shown above, the true number of objects in this specific test image was provided as input to to the multiscale normalized cut routine).

in case certain objects in the image were neglected in the human tracing (in practice this occured extremely rarely).

In order to allow only certain topological changes, it is necessary to be able to classify the type of topological change flipping a particular non-simple point will cause. By "change topology" we mean alter the number of components, or cavities. We would thus like to know which topological quantity is affected by flipping a particular non-simple point. In this supplementary section we discuss how non-simple points can be classified rigorously using additional concepts from the field of digital topology.

2.1. Identifying addition and deletion of objects using topological numbers

Certain cases can be identified based on topological number alone (see Section 2.1 of the main text for a review of topological numbers). Let $(\kappa, \overline{\kappa})$ be some complementary adjacency relation in either a 2d or 3d space.

Theorem 2.1. *Topological number characterization of object deletion. Suppose q is an element of the foreground L. Then flipping q will result in a κ-component deletion if and only if $T_\kappa(q, L) = 0$.*

Proof. The proof (given in [7]) is simple: suppose that q is flipped, then a connected component of L is removed if and only if q is an isolated point, in which case $T_\kappa(q, L) = 0$. Conversely suppose $T_\kappa(q, L) = 0$, then q must be an isolated point with respect to the foreground. Therefore flipping q results in an object deletion.

Note that object deletion is only one way to decrease the number of foreground connected components. The merging of two existing components is the other way. □

A similar statement can be offered for object addition:

Theorem 2.2. *Topological number characterization of object addition. Suppose q is an element of the background $\overline{L}$. Then flipping q will result in a κ-component addition if and only if $T_\kappa(q, L) = 0$.*

Proof. Suppose that q is flipped, then a connected component L is added if and only if q is an isolated point, in which case $T_\kappa(q, L) = 0$. Conversely suppose $T_\kappa(q, L) = 0$, then q must be an isolated point with respect to the foreground. Therefore flipping q results in an object addition. Again, we note that object addition is only one way to add to the number of components. Splitting an existing object into two is the other. □

Topological numbers can also be used to identify the creation and deletion of cavities, using a similar criteria based on the "background" topological number. Recall that the background is the unique infinite $\overline{\kappa}$-connected component of $\overline{L}$, and by definition any other $\overline{\kappa}$-connected component of $\overline{L}$ is called a cavity in L.

Topological number characterization of cavity deletion. Suppose q is an element of the background $\overline{L}$. Then flipping q will result in a $\overline{\kappa}$-component (cavity) deletion if and only if $T_{\overline{\kappa}}(q, \overline{L}) = 0$.

Proof. Suppose that q is flipped, then a connected component of $\overline{L}$ is removed if and only if q is an isolated background point, in which case $T_{\overline{\kappa}}(q, \overline{L})$=0. Conversely suppose $T_{\overline{\kappa}}(q, \overline{L}) = 0$, then q must be an isolated background point. Therefore flipping q results in deletion of a cavity. □

Theorem 2.3. *Topological number characterization of cavity addition. Suppose q is an element of the foreground L. Then flipping q will result in a $\overline{\kappa}$-component (cavity) addition if and only if $T_{\overline{\kappa}}(q, \overline{L}) = 0$.*

Proof. Suppose that q is flipped, then a connected component in $\overline{L}$ is added if and only if q is an isolated background point, in which case $T_{\overline{\kappa}}(q, \overline{L}) = 0$. Conversely suppose $T_{\overline{\kappa}}(q, \overline{L}) = 0$, then q must be an isolated point with respect to the background. Therefore flipping q results in the creation of cavity.

As in the case of foreground components, deletion/addition of cavities by flipping isolated background points is *not* the only way in which the number of background components can be changed. A cavity can be merged (with other cavities or the infinite background component) and split into two cavities by a single flip of some pixel q. However in such cases it will not be true that $T_{\overline{\kappa}}(q, \overline{L}) = 0$. □

We note that if $T_{\kappa}(q, L) = 0$ then it follows $T_{\overline{\kappa}}(q, \overline{L}) = 1$ and vice versa. This is because an isolated foreground (background) point is clearly surrounded by a background (foreground), which under normal adjacency relations for either κ or $\overline{\kappa}$ will be connected as a single component within such a neighborhood itself. Finally we recall that if $T_{\kappa}(q, L) = T_{\overline{\kappa}}(q, \overline{L}) = 1$ the point is simple and thus causes no topological change when altered. Therefore, we have a characterization for all points for which $T_{\kappa}(q, L) \in \{0, 1\}$ and $T_{\overline{\kappa}}(q, \overline{L}) \in \{0, 1\}$.

2.2. Identifying splits, mergers, and hole addition/deletion using extended topological numbers

Topological numbers as originally defined are insufficient to characterize the precise nature of topological changes caused by flipping non-simple points for which $T_{\overline{\kappa}}(q, \overline{L}) > 1$ or $T_{\kappa}(q, L) > 1$. For example, flipping a pixel q from background to foreground when $T_{\kappa}(q, L) > 1$ clearly implies a topological change since q is non-simple; however, the nature of this change depends on *non-local* properties that are not captured by the local neighborhood from which $T_{\kappa}(q, L)$ is computed. We need access to global connectivity information that would enable us to distinguish between, for example, merging two objects versus creating a hole. Global connectivity is well defined over the image, however the binary image itself does not represent such information directly. Hence, such classifications require additional measures beyond topological numbers that take into account this global information [5].

References

[1] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *CVPR*, 2005.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *Proc. CVPR*, 2009.

[3] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *CVPR*, 2008.

[4] P. Dollar, Z. Tu, and S. Belongie, "Supervised Learning of Edges and Object Boundaries," in *CVPR*, 2006.

[5] V. Jain, "Machine learning of image analysis with convolutional networks and topological constraints," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] G. Bertrand and G. Malandain, "A new characterization of three-dimensional simple points," *Pattern Recognition Letters*, 1994.