Supplementary Information: Flexible Learning-Free Segmentation and Reconstruction for Sparse Neuronal Circuit Tracing

Ali Shahbazi[1,+], **Jeffery Kinnison**[1,2,+], **Rafael Vescovi**[2,3], **Ming Du**[4], **Robert Hill**[5], **Maximilian Jösch**[6], **Marc Takeno**[7], **Hongkui Zeng**[7], **Nuno Maçarico da Costa**[7], **Jaime Grutzendler**[5], **Narayanan Kasthuri**[2,3], **and Walter J. Scheirer**[1,*]

[1]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA
[2]Center for Nanoscale Materials, Argonne National Laboratory, Lemont, IL, USA
[3]Department of Neurobiology, University of Chicago, Chicago, IL, USA
[4]Department of Materials Science and Engineering, Northwestern University, Evanston, IL, USA
[5]Department of Neurology, Yale University, New Haven, CT, USA
[6]Neuroethology Group, IST Austria, Klosterneuburg, Austria
[7]Allen Institute for Brain Science, Seattle, WA, USA
[*]walter.scheirer@nd.edu
[+]These authors contributed equally to this work.

ABSTRACT

This document contains all supplemental figures and methods descriptions.

Supplementary Methods: Evaluation Metrics

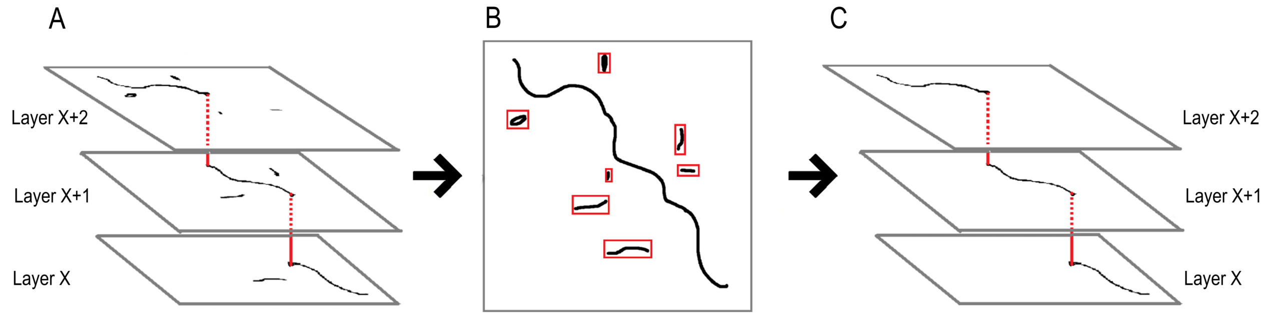
In our experiments we compared our EM reconstructions and one of our μCT X-ray stacks against expert annotated ground-truth. Since 3D reconstruction is the goal of the work, we chose Hausdorff distance as an evaluation tool. It is a well-known evaluation method for 3D meshes and reconstructions[1]. Instead of one-by-one matching as is done when calculating variation of information, which checks the location of each point on the first model with a similar location on second model, Hausdorff distance considers many-to-many correspondence. Consider a 3D reconstruction A and corresponding 3D ground-truth B: $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_n\}$. The Hausdorff distance from A to B will be $\tilde{\delta_H}(A,B) = max_{a \in A} min_{b \in B} \|a - b\|$. Throughout this article, we use the convention A is in X/Y of the distance of B. Thus when we say A is in 1/1000 of the distance of B, this means the average distance between A and B is less than 0.001.

Supplementary Methods: 3D U-Net Training

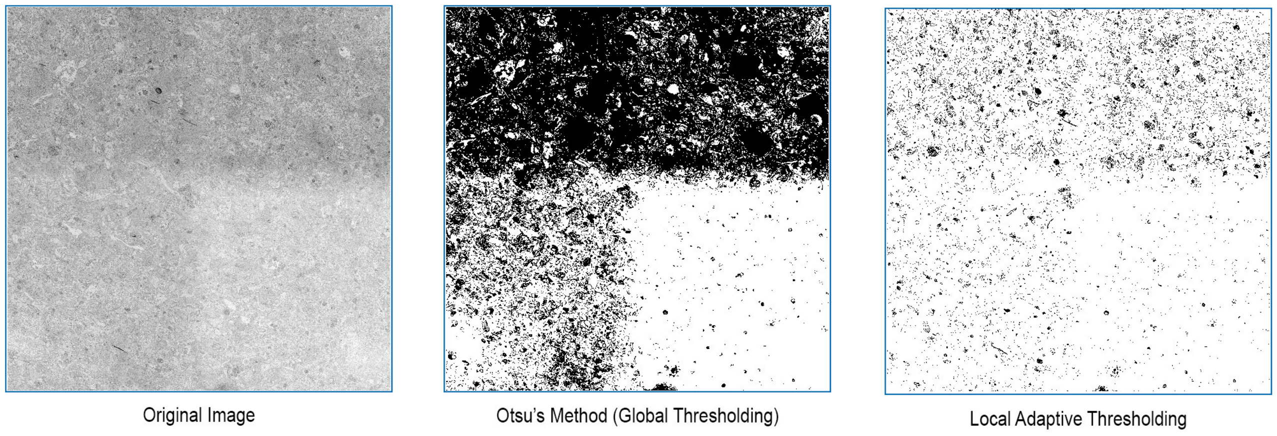
For μCT X-ray segmentation, a standard 3D U-Net model[2] was trained on the manual segmentation of the SRB dataset[3], which consists of a $300 \times 300 \times 100$ voxel volume. Due to memory constraints, each training batch consisted of a single $256 \times 256 \times 19$ block augmented by randomized rotation, transpose, and grayscale intensity scaling. The model was trained for 60 epochs using weighted mean-squared error loss and the Adam optimizer[4] with a learning rate of 1×10^{-4}. Training for sSEM APEX2-positive SAC segmentation proceeded using the same protocol, however the model was trained on manual segmentations created by Jösch *et al.*[5].

Supplementary Table S 1. Running time comparison to the previous version of the pipeline[5].

Method	Running time
Jösch et al. [5]	136 min
2D FLoRIN	24.82 min
3D U-Net	3559.3 min



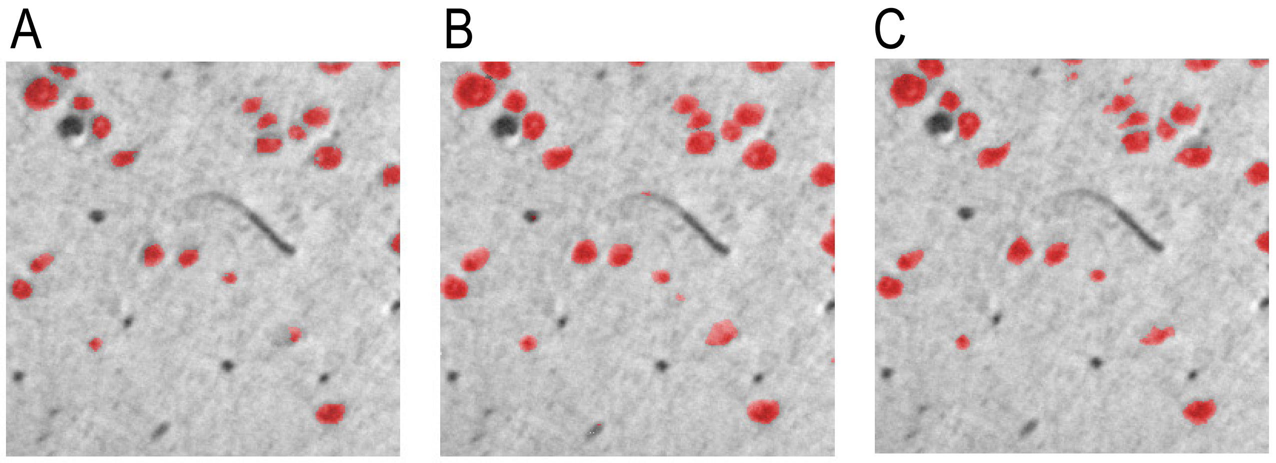
Supplementary Figure S 1. The 2D Identification process in FLoRIN first registers all connected components discovered in the Identification stage into a single plane. Connections are preserved by connecting components that overlap with one another in 2D, and unconnected components are discarded. The remaining components are then placed into a 3D volume.


Original Image
Otsu's Method (Global Thresholding)
Local Adaptive Thresholding

Supplementary Figure S 2. A comparison of the performance of global versus local thresholding methods on a noisy image. (Left) Neural images are often subject to large shifts in the grayscale distribution that can hamper reconstruction efforts. (Center) Otsu's method[6] binarizes images using a global threshold value, however due to grayscale shifts the binarization includes large portions of the image background. (Right) Our LAT algorithm operates by observing a local neighborhood around each voxel to reduce the impact of distant noise. In this case, LAT captures more of the features of interest despite large grayscale shifts.

References

1. Aspert, N., Santa-Cruz, D. & Ebrahimi, T. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, vol. 1, 705–708 (IEEE, 2002).
2. Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. & Ronneberger, O. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 424–432 (Springer, 2016).
3. Dyer, E. L. *et al.* Quantifying mesoscale neuroanatomy using x-ray microtomography. *eNeuro* **4**, ENEURO–0195 (2017).
4. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
5. Joesch, M. *et al.* Reconstruction of genetically identified neurons imaged by serial-section electron microscopy. *eLife* **5**, e15015 (2016).
6. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Syst. Man, Cybern.* **9**, 62–66 (1979).



Supplementary Figure S 3. A comparison of SRB (A) manual annotations, (B) 3D U-Net cell segmentation, and (C) 3D FLoRIN cell segmentation. In general, 3D U-Net tends to over-segment the cells, incurring merge errors with nearby cells and on correctly identifying a vasculature segment as a cell. 3D FLoRIN, on the other hand, clearly separates grouped cells and does not misclassify vasculature.

```
Data: img: an n-dimensional image to threshold
Data: n: the dimensionality of img
Data: d: an n-tuple containing the size of each dimension of img
Data: s: an n-tuple containing the dimensions of the box around each pixel
Data: t: the threshold value to use, number in range $[0, 1]$
Result: binarization of img
let out be an array the same size as img;
intImg = img;
for i in 1..n do
    // Compute the cumulative summation over dimension i
    intImg = cumulativeSummation(intImg, i);
end
let indices be the set of all binary strings length n;
let low, hi, vertex be length n arrays filled with zeros;
parity = n mod 2;
foreach element e in intImg do
    x = index(intImg, e);
    for i in 1..n do
        low[i] = x[i] - s[i] / 2;
        hi[i] = x[i] - s[i] / 2;
        if low[i] $<$ 1 then
            low[i] = 0;
        end
        if hi[i] $>$ d[i] then
            hi[i] = d[i];
        end
    end
    count = $\prod_{i=1}^{n} hi[i] - low[i]$;
    sum = 0;
    foreach idx in indices do
        p = 0;
        for i in 1..n do
            p = p + idx[i];
            if idx[i] = 1 then
                vertex[i] = hi[i];
            else
                vertex[i] = low[i];
            end
        end
        p = p mod 2;
        if p = parity then
            sum = sum + intImg[vertex];
        else
            sum = sum - intImg[vertex];
        end
    end
    if img[x] $\times$ count $\leq$ sum $\times$ (1.0 - t) then
        out[x] = 0;
    else
        out[x] = 1;
    end
end
return out
```

Algorithm 1: Local Adaptive Thresholding