Biologically-Aware Algorithms for Connectomics

A DISSERTATION PRESENTED
BY
BRIAN P. MATEJEK
TO THE JOHN A. PAULSON SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
APRIL 2021

Dissertation advisor: Prof. Hanspeter Pfister Brian P. Matejek

Biologically-Aware Algorithms for Connectomics

Abstract

With billions of neurons and trillions of synapses in humans, the inner workings of the brain remain one of the great open mysteries of the universe. A complete understanding of the brains of even tiny organisms such as *Caenorhabditis elegans* remains elusive. The field of connectomics seeks to unravel the mysteries of brains by analyzing their circuitry at a synaptic level. In pursuit of this goal, neuroscientists extract and image brain tissue from various species, trace the neurons through the images, identify all synaptic connections, and construct a wiring diagram. A team of researchers spent a dozen years manually extracting the first nearly complete connectome from an image stack. This early success contained 302 neurons and 5,000 synapses. In the succeeding decades, automated algorithms have complemented and, at times, replaced manual human reconstruction efforts. More recently produced wiring diagrams, or connectomes, contain thousands of neurons with millions of synaptic connections.

Recent advancements in image acquisition using multi-beam electron microscopes enable neuroscientists to capture one terabyte of raw image data every hour. Although these engineering achievements open the door for imaging larger brain volumes from more evolved species, human labor, particularly that which requires expert knowledge, has become the bottleneck. Therefore, researchers have increasingly relied on automated solutions to reconstruct neurons from the raw image data and extract the wiring diagrams. The previous decade has seen a deluge of new algorithms that confront the challenges of converting the raw image data into connectomes for further

analysis. Despite the incredible success of these algorithms, there is still room for improvement in accuracy and efficiency.

By guiding algorithm design with existing biological knowledge, we can improve accuracy, reduce complexity, and generate results more faithful to the neuronal data. We demonstrate these principles with four biologically-aware algorithms that confront various problems across the connectomics pipeline. We explicitly tailor our solutions for connectomic data and outperform existing methods that are generally agnostic to the underlying biology. This dissertation considers just a tiny fraction of the numerous challenges when working with connectomics data. However, we believe that biologically-aware algorithms like those presented here can make inroads into tackling a wide range of problems in connectomics.

Contents

1 Introduction 1
- 1.1 The Connectomics Pipeline . . . 5
- 1.2 Thesis Statement . . . 9
- 1.3 Contributions and Outline . . . 9

2 Compression 13
- 2.1 Motivation . . . 14
- 2.2 Related Work . . . 16
- 2.3 The Compresso Scheme . . . 18
- 2.4 Evaluation and Results . . . 22
- 2.5 Conclusions . . . 24

3 Error Correction 26
- 3.1 Motivation . . . 27
- 3.2 Related Work . . . 30
- 3.3 Biologically-Constrained Graphs . . . 31
- 3.4 Experimental Set Up . . . 38
- 3.5 Results . . . 41
- 3.6 Conclusions . . . 47

4 Subgraph Enumeration 48
- 4.1 Motivation . . . 50
- 4.2 Related Work . . . 55
- 4.3 Subgraph Enumeration on the Connectome . . . 56
- 4.4 Experimental Set Up . . . 62
- 4.5 Results . . . 64
- 4.6 Conclusions . . . 70

5 Skeleton Generation 73
- 5.1 Motivation . . . 74
- 5.2 Related Works . . . 78

5.3 Biologically-Aware Skeleton Generation 79
5.4 Experimental Set Up 90
5.5 Results 92
5.6 Conclusions 96

6 Conclusions and Future Work **97**
6.1 Future Work 99
6.2 Concluding Remarks 100

Appendix A Error Correction **101**
A.1 Node Generation and Reduction 101
A.2 Node Convolutional Neural Network 103
A.3 Skeleton Benchmark and Generation 105
A.4 Edge Generation 107
A.5 Edge Weights 109

Appendix B Skeleton Generation **113**

Bibliography **119**

Listing of figures

1.1 Connectomics as an Interdisciplinary Science 2
1.2 The *Caenorhabditis elegans* Connectome 3
1.3 Mapping Behavior to Function Through Structure 4
1.4 Conflicting Scales of Connectomics 5
1.5 The Connectomics Pipeline 6
1.6 Label Volumes 7
1.7 Segmentation Errors 8
1.8 Biologically-Aware Algorithms in the Connectomics Pipeline 10

2.1 Compression in the Connectomics Pipeline 14
2.2 64-bit Label Volumes 15
2.3 Compresso Window Encoding 18
2.4 Label Volume Window Redundancy 19
2.5 Compresso Quantitative Results 23

3.1 Error Correction in the Connectomics Pipeline 27
3.2 Biologically-Constrained Graphs 28
3.3 Node Generation 32
3.4 Singleton Segments 33
3.5 Common Split Errors 35
3.6 Generating Skeletons for Error Correction 36
3.7 Machine-Learned Morphologies 37
3.8 Qualitative Successes and Failures of Error Correction 42
3.9 Edge Generation Results 44
3.10 Edge Classification Results 45

4.1 Subgraph Enumeration in the Connectomics Pipeline 49
4.2 Motifs in the Wiring Diagram 50
4.3 Canonical Labeling of Subgraphs 52
4.4 Subgraph Enumeration for Large-Scale Connectomes 53
4.5 Enumeration Times and Neighborhood Sizes 58

4.6 Problems with Advanced Heuristics 60
4.7 Subgraph Distributions of Size Three 65
4.8 Most Frequent Motifs 66

5.1 Skeleton Generation in the Connectomics Pipeline 74
5.2 Block-Based Skeletonization 75
5.3 Biologically-Aware Topological Thinning 76
5.4 Skeleton Generation Overview 80
5.5 Bubble Filling 81
5.6 Anchor Point Computation 84
5.7 Topological Thinning 85
5.8 Neurite Width Estimation 88
5.9 Skeleton Refinement 89
5.10 Qualitative Skeleton Generation Results 94
5.11 Computational Complexity 96

A.1 Singleton Slices 102
A.2 Distribution of Non-Trivial Skeletons 103
A.3 CNN Architecture for Region Merging 105
A.4 Skeleton Benchmark Dataset 106
A.5 Determining the t_{edge} Parameter 108
A.6 Additional Edge Generation Qualitative Results 109
A.7 Multicut β Analysis 111
A.8 Edge Weight Conversion Functions 112

B.1 Dendrite Comparison 114
B.2 Axon Comparison 115
B.3 JWR Additional Visual Results 116
B.4 FIB-25 Additional Visual Results 117
B.5 Zebrafinch Additional Visual Results 118

To my family.

Acknowledgments

Thank you to my advisor, Hanspeter Pfister, for an enlightening five years of academic studies and constant support as I pursued my varied interests. I am incredibly grateful to have had the opportunity to work with you for these last five years.

I also would like to thank my numerous mentors over the years both at Harvard and Princeton, especially my dissertation committee Todd Zickler and Michael Mitzenmacher, as well as the other academics I was fortunate enough to work with: Thomas Funkhouser, Christopher Moretti, Babis Tsourakakis, and Kálmán Palágyi. I would like to thank Donglai Wei and Toufiq Parag for their guidance as leaders of the "Connectomics Subgroup" within the Visual Computing Group.

This dissertation would not be possible without the thoughtful and engaging conversations I had at Harvard with my fellow Ph.D. labmates: Daniel Haehn, Fritz Lekschas, Spandan Madan, Salma Abdel Magid, Nam Wook Kim, Felix Gonda, Zudi Lin, Tica Lin, and Gaurav Bharaj. I also greatly appreciated the conversations during the weekly pizza happy hours with Sai Srivatsa Ravindranath, Adi Suissa-Peleg, William Gilpin, and Alex Shapson-Coe.

Graduate studies can be lonely affairs. Thankfully, I had great friends in Boston in Chris Yee and Natalie Murillo, who provided me with some much-needed escape from the academic grind.

I want to thank my siblings, Megan, Michael, Jimmy, and Robert, for their unwavering support during my studies. I particularly appreciated the home-cooked meals from Jimmy and Scott (and Rudy, of course!) before the pandemic and the weekend `Civilization` marathon sessions with Robert during the lockdown.

Most importantly, I would like to thank my parents for their continual encouragement, especially when I was most discouraged.

Contributing Authors

The following authors contributed to this dissertation:

- Chapter 2: Daniel Haehn, Fritz Lekschas, Michael Mitzenmacher, and Hanspeter Pfister
- Chapter 3: Daniel Haehn, Haidong Zhu, Donglai Wei, Toufiq Parag, and Hanspeter Pfister
- Chapter 4: Donglai Wei, Tianyi Chen, Charalampos E. Tsourakakis, Michael Mitzenmacher, and Hanspeter Pfister
- Chapter 5: Tim Franzmeyer, Donglai Wei, Xueying Wang, Jinglin Zhao, Kálmán Palágyi, Jeff W. Lichtman, and Hanspter Pfister

1

Introduction

The inner workings of the brain, with billions of neurons and trillions of synapses in humans, remain one of the great open mysteries of the universe [20, 130]. Although over a century of research has provided significant insights into the interplay between individual neurons through synaptic connections and other biological mechanisms, a large amount remains unknown about the brain's internal mechanisms as the number of involved neurons scales. However, neuroscientists can now image large swaths of brain tissue with electron microscopes to extract wiring

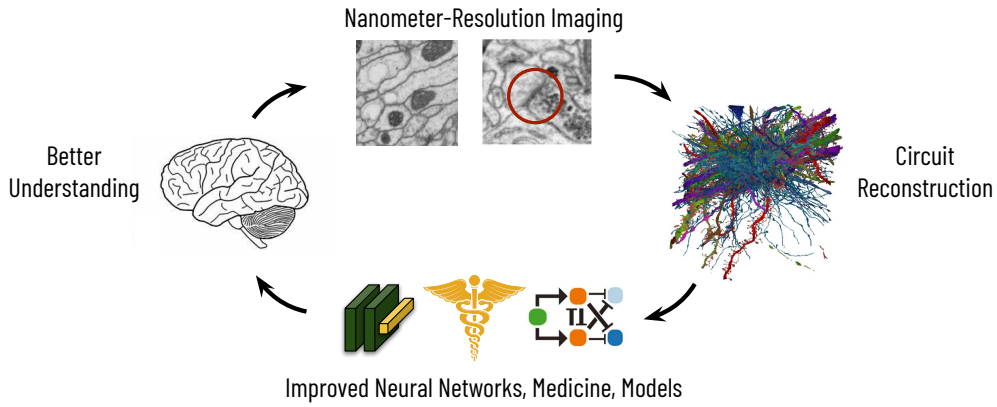


Figure 1.1: The connectomics discipline requires close collaboration between neuroscientists and computer scientists. Neuroscientists extract a region of brain tissue and image it at nanometer resolution. Automatic reconstruction and detection algorithms extract a wiring diagram from these images. By analyzing this circuitry, we hope to improve neural networks, medicine, and computational models of the brain. These advancements will further our understanding of the brain, which will then guide the imaging process.

diagrams, or connectomes, at a synaptic level. The field of connectomics seeks to unravel the brain's mysteries by analyzing the underlying circuitry observed in these images.

The connectomics discipline requires close collaboration between neuroscientists and computer scientists. Figure 1.1 provides a brief overview of a typical connectomics workflow. First, neuroscientists extract a relevant section of brain tissue and use electron microscopes to image the data at nanometer resolution. Next, computer vision algorithms transform the images into a wiring diagram by segmenting the neurons and identifying synapses and other organelles such as mitochondria. By analyzing the reconstructed circuitry, researchers hope to advance artificial neural networks [43], improve understanding of certain neurological diseases [30], and create more faithful computational models of the brain [70]. Neuroscientists can better understand the brain with these improvements, with the cycle continuing as these insights further guide the imaging process.

In 1986, White *et al.* achieved a significant milestone in connectomics by reconstructing the

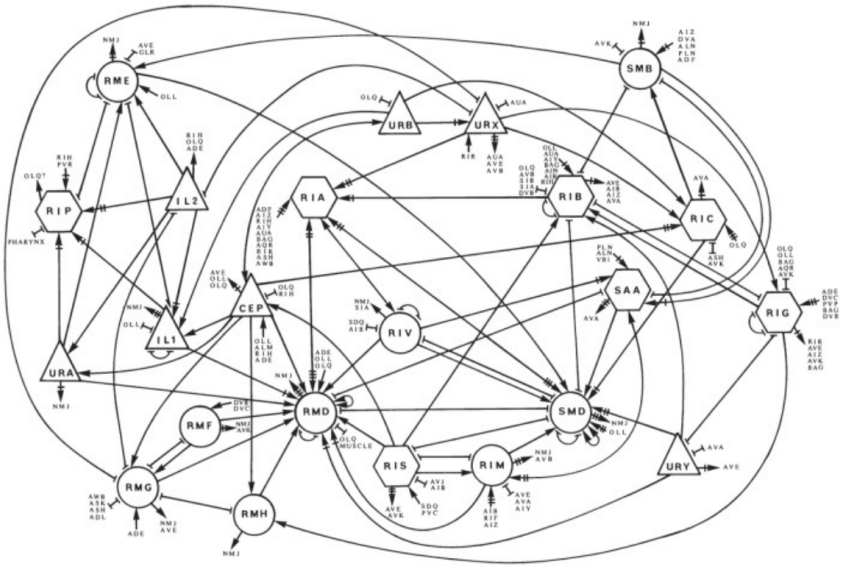


Figure 1.2: One of the earliest success stories for connectomics came in 1986 when White *et al.*reconstructed a nearly complete connectome for *C. elegans*. This early wiring diagram contains 302 neurons and over 5,000 synaptic connections. Image from Emmons [26].

first nearly complete wiring diagram, a feat requiring over a dozen years of tedious manual labor (Figure 1.2) [136]. This first connectome of *Caenorhabditis elegans* (*C. elegans*) contained exactly 302 neurons and approximately 5,000 chemical synapses and 600 gap junctions. This monumental accomplishment quickly proved valuable as researchers identified a handful of motifs—frequently occurring clusters of neurons with important biological function [121]. In the three and a half decades since, advancements in image acquisition [25, 140, 142], automatic reconstruction [49, 67], and synaptic detection [45, 72] have enabled the extraction of larger partial wiring diagrams from more sophisticated species such as fruit flies [124, 141], mice [25, 123], and zebra finch [62], among others.

Research by Jovanic *et al.*showed the potential power of connectomics in going from observed behavior to a working computational model [52]. The authors observed two distinct behaviors to crawling *Drosophila* larvae in response to mechanosensory stimuli, i.e., an air puff (Figure 1.3,

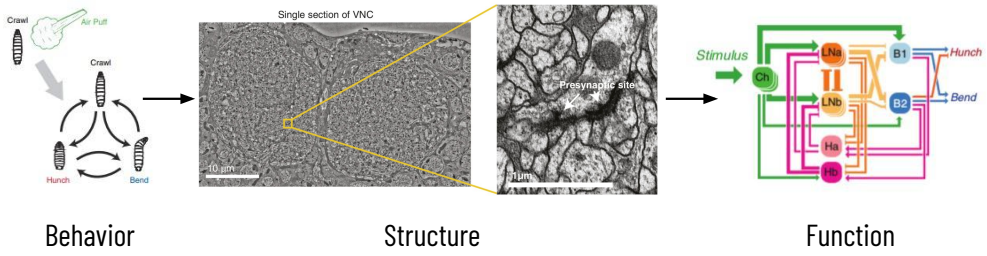


Figure 1.3: One goal of connectomics is to map behavior to neuronal structure to create better computational models of the brain. A study by Jovanic *et al.*explored the behavior exhibited by *Drosophila* larvae when confronted with a puff of air while crawling. After noticing that the specimens either hunched or bent away from the puff, they extracted the relevant section of brain tissue and reconstructed the neurons and synaptic connections. From the extracted wiring diagram, they created a computational model that matched the observed behavior. Images from Jovanic *et al.* [52].

left). The larvae either hunched to avoid the air puff or bent their head away from the puff to explore the local environment. After viewing this behavior, the researchers extracted the relevant region of the brain, imaged it at nanometer resolution (Figure 1.3, center), and reconstructed the neurons and synaptic connections to extract a wiring diagram (Figure 1.3, right). The authors identified specific motifs in the circuitry that would result in a hunch or a bend with correct inputs into the sensory neurons. This mapping from observed behavior to neuronal structure to computational function provides a brief glimpse into the potential future insights that connectomics might offer.

Although much of the research in connectomics has previously focused on small species such as worms and fruit flies, more recent endeavors consider more sophisticated animals, including birds and small mammals. Neurons in these new specimens can easily span more than 100 μm; however, we still require image resolutions as fine as 10 nm to identify the synaptic connections (Figure 1.4). This four order-of-magnitude difference between resolution and image size compounds in each dimension, and even extracting a small wiring diagram from a mammalian brain requires teravoxel image volumes. These teravoxel volumes represent the minimum size needed

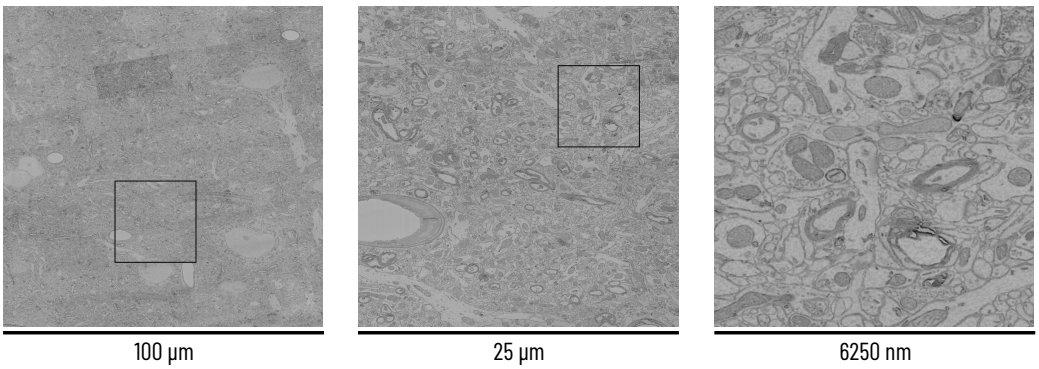


Figure 1.4: Neurons can easily span over $100\ \mu\text{m}$ in length in mammalian brains. Thus, our image volumes often exceed this width, height, and depth. However, we need resolutions as fine as $10\ \text{nm}$ to detect synaptic connections and other organelles, such as mitochondria. This $10,000\times$ differential per dimension means extracting even small circuits require teravoxel image volumes.

to extract a small wiring diagram from a mammalian brain; increasingly, neuroscientists extract brain regions that exceed a cubic millimeter (petavoxel of data) [115, 142]. In the extreme case for mice, a typical brain has a volume of $500\ \text{mm}^3$, and would require one exabyte (1000 petabytes) of image dat to completely reconstruct [71]. Although these numbers may seem outlandish now, an increasing number of labs worldwide view such goals as a significant future milestone.

1.1 The Connectomics Pipeline

We have briefly discussed the overarching goal of connectomics and provided a crude outline of the process: we extract a wiring diagram from brain tissue to improve machine learning, medicine, and understanding of the underlying neuronal circuitry. However, the leap between identifying a region of interest in the brain and producing an accurate connectome with analysis consists of numerous steps that each require close collaboration between neuroscientists and computer scientists (Figure 1.5).

First, neuroscientists extract either a section of or an entire brain for reconstruction and future

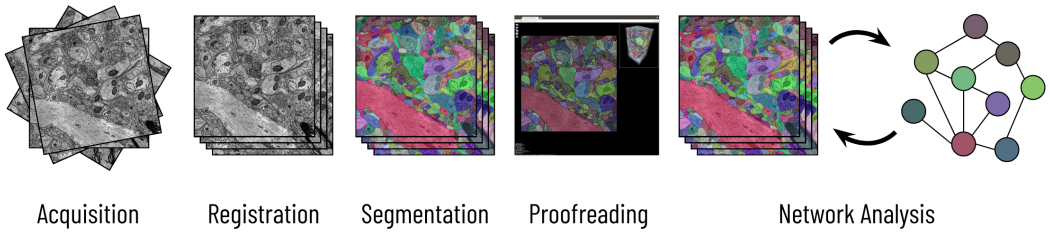


Figure 1.5: There are five main stages in the connectomics pipeline to transform a brain tissue section into a wiring diagram: acquisition, registration, segmentation, proofreading, and network analysis. Each step produces unique challenges.

analysis. Often, they target brain regions based on existing knowledge of their function, such as the visual cortex of a mouse or the mushroom body of a fly. They then stain the brains with various chemicals such as formaldehyde or osmium tetroxide. These chemicals bind to membranes and other organelles to provide stark contrast when imaging [12]. The brain tissue is then typically cut into thin slices between 20–40 nanometers thick. An electron microscope images each slice of brain tissue. A multi-beam electron microscope can collect one terabyte of raw image data every hour or one petabyte every six months [56, 123]. The throughput of these multi-beam electron microscopes will only continue to improve as engineering breakthroughs enable even more parallel beams.

After imaging the data, the registration and alignment step transforms the image stack into a 3D volume. The physical slicing of the brain tissue and then its transport to the microscope can cause deformations such as stretching or scrunching in each slice. We correct these abnormalities during this step. First, a feature detection algorithm such as SIFT identifies salient points across the image stack [112]. Next, we pair points in neighboring image slices representing the same physical structure, e.g., the boundary of a mitochondrion or the edge of a blood vessel. Various optimization techniques reduce a cost function on the distances between matched points to create an affine transformation for each image slice [58]. After registration and alignment, we consider

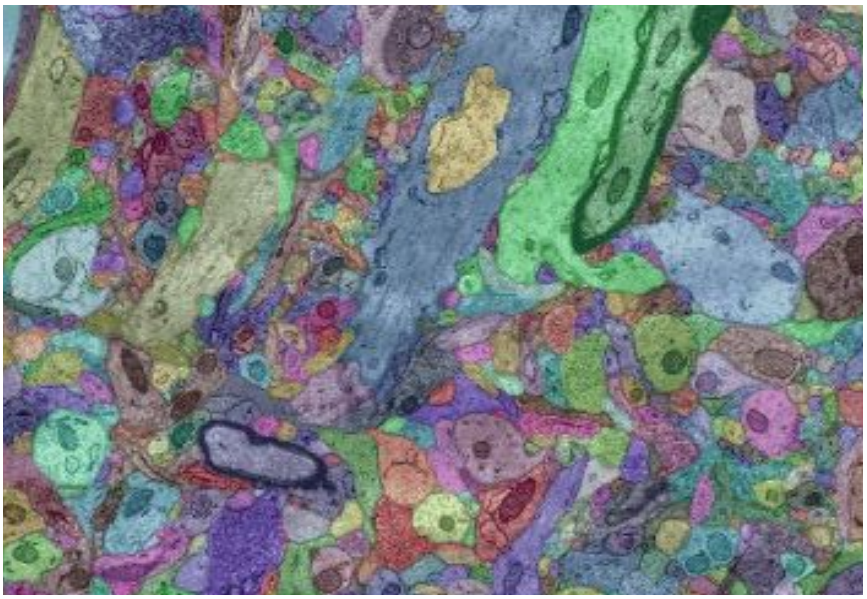


Figure 1.6: Automatic segmentation algorithms transform the image stacks into label volumes where two voxels receive the same label if they belong to the same neuronal process. Previous manual and semi-automatic methods cannot scale to the ever-increasing size of the acquired data.

the array of image slices as an image volume with resolutions typically between $3 - 6$ nanometers in x and y and $20 - 40$ nanometers in z. Figure 1.4 shows one slice of an image volume after acquisition, registration, and alignment.

Once the image stack is aligned correctly, we can begin segmenting the images into individual neurons and identifying synaptic connections. During the segmentation step, we create label volumes from the image volumes. In these label volumes, two voxels receive the same label if and only if they belong to the same neuron in the image (Figure 1.6). In the early years of connectomics, a team of biologists manually traced neurons through the image stack [136]. However, this manual process cannot scale to the ever-increasing size of the connectomic image volumes. More recently, machine learning algorithms have replaced humans in this step [88, 97], with deep learning methods outperforming existing strategies [15, 32, 110]. These automated solu-

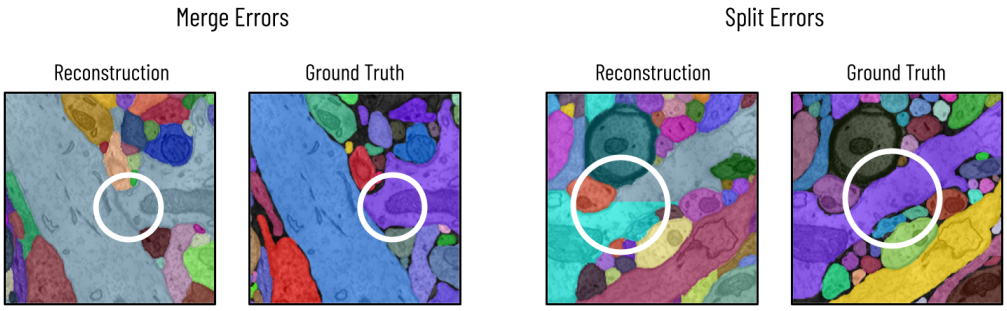


Figure 1.7: Despite the phenomenal advancements in automatic segmentation techniques within the last ten years alone, they still produce errors, particularly at large scales. Here, we see the two types of errors. In *merge errors*, left, two neuronal processes receive the same label. In *split errors*, right, one neuronal process receives two or more labels.

tions have improved significantly in the last decade, with some methods even surpassing human accuracy on small challenge datasets [34, 49, 66, 81, 82]. Synapse detection has followed a similar evolution, with automated deep learning methods supplanting human efforts [45, 99]. Beyond the neurons and synapses themselves, researchers now consider the tasks of segmenting smaller organelles such as mitochondria, which they believe can act as essential cues for differentiating neuron types [11, 133].

Despite the phenomenal accuracy of the automated reconstruction algorithms, they still produce some errors, particularly at larger scales. For the label volumes, there are two types of observed errors (Figure 1.7). In *merge errors*, two or more neuronal processes receive the same label. Conversely, in *split errors*, one neuronal process receives two or more labels. We need to correct these errors to produce accurate wiring diagrams. A diverse set of tools allow for manual [38], semi-automatic [40], or automatic [23, 76, 150] error correction.

After segmenting the neurons and detecting the synapses, we can extract the wiring diagram from the image volume. Typically, we represent the connectome as a graph (Figure 1.2) where vertices correspond to neurons and weighted edges to the synaptic connections [77, 116, 141].

After graph construction, we can quantify numerous properties of the connectomes like edge density, graph diameter, and the clustering coefficient. Beyond these global attributes, we also want to identify subgraphs, or motifs, that correspond to biologically important functionality. Current analyses typically concern large-scale motifs between different types of neurons [116] or small-scale motifs of at most a few neurons [18, 103, 129]. As the pipeline has become more automated, the analyses have increased in sophistication, with longitudinal studies now differentiating between different sexes [18] and ages [137].

1.2 Thesis Statement

As connectomic data increases in both size and complexity, end-to-end deep learning models can become cumbersome, and standard off-the-shelf approaches often fail to capture the nuance of the underlying data. However, by guiding algorithm design with existing biological knowledge, we can improve accuracy, reduce complexity, and generate results more faithful to the neuronal data. These biologically-aware algorithms can incorporate domain knowledge in several ways, such as guiding the initial principles, refining the solution search space, and constraining (or even augmenting) the outputs. We can design these explicitly tailored algorithms for the numerous problems along the entire pipeline as each step from segmentation to network analysis introduces new challenges.

1.3 Contributions and Outline

In this dissertation, we propose four biologically-aware algorithms for problems in the segmentation-to-analysis section of the connectomics pipeline (Figure 1.8). In Chapter 2, we discuss the compression of the label volumes produced during the reconstruction process. We represent the label volumes as 64-bit integers for large-scale reconstructions given the many potential neurons

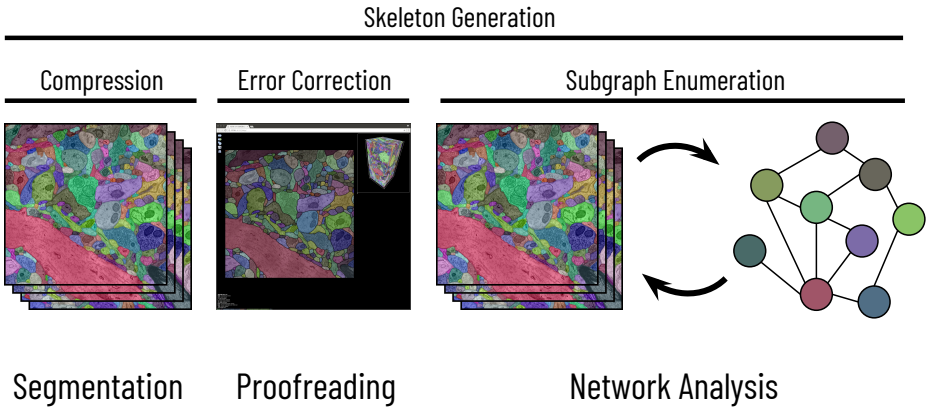


Figure 1.8: This dissertation discusses four challenges along the segmentation-to-analysis component of the connectomics pipeline: compression (Chapter 2), error correction (Chapter 3), subgraph enumeration (Chapter 4), and skeleton generation (Chapter 5). We propose biologically-aware algorithms that address each challenge with an emphasis on adhering to specific properties of the neuronal processes.

and neuron fragments. In contrast, we only need one byte per voxel in the image volume since the electron microscope assigns each pixel an integer value in the range $[0, 256)$. As discussed previously, the multi-beam electron microscopes can now image ten terabytes of raw image data per day [123]. After reconstruction, the corresponding label volumes for one day of imaging require 80 terabytes of disk storage when uncompressed. Furthermore, this storage disparity deepens as we improve and refine the segmentation, often generating multiple snapshots of the same image volume. We propose a window-based compression scheme for the label volumes that exploits the highly redundant boundary shapes between adjacent neurons. We compress the label volumes by over $700\times$, a 6-fold improvement over the general-purpose methods currently used in the community.

Although the automatic reconstruction techniques provide excellent results, even surpassing human accuracy on certain challenge datasets [34, 49, 67], these algorithms still make mistakes, particularly at large scales. In Chapter 3, we consider the problem of refining an oversegmentation

(i.e., a volume with many split but few merge errors) to produce a more accurate reconstruction. We reformulate the error correction process as a multicut graph partitioning optimization problem to leverage global context. However, since graph optimization is a computationally expensive task, we employ biological constraints when constructing our graph to reduce vertices and edges. Our hand-designed geometric constraints and machine-learned morphologies substantially prune the graph to allow for high throughput error correction. We improve on the segmentation results from two state-of-the-art algorithms on four publicly available datasets on average by 21.3%.

After neuronal reconstruction and synapse detection, neuroscientists typically transform the volumes into a graph, with vertices corresponding to individual neurons and weighted edges indicating synaptic connectivity. With this representation, neuroscientists hope to find motifs—frequently occurring subgraphs within the wiring diagram that correspond to specific biological functions. To date, most analyses on these graphs have focused only on small motifs with two or three verticies [18, 137], or on large motifs with specific constraints on the type of neuron [116]. In Chapter 4, we describe a network-centric subgraph enumeration strategy to identify frequently occuring subgraphs in the connectomes. In contrast to previous methods, we augment our input graphs with edge labels to differentiate between types of synaptic connections (e.g., excitatory/inhibitory or chemical/electrical) and better adhere to the underlying biological properties of the wiring diagram. We demonstrate our results on eleven connectomes and provide extensive summaries of the over 26 trillion subgraphs enumerated.

Skeletons play an important role across the entire connectomics pipeline with applications in analysis [125, 141], segmentation evaluation [49], visualization [85], and error correction [23, 76]. Most of these applications use off-the-shelf skeleton generation techniques that are agnostic to the underlying biology. In chapter 5, we propose a biologically-aware, topological thinning strategy to create more faithful representations of neuronal structures. We ensure connectivity between synapses and anchor the skeletons onto the cell bodies. Inspired by cable theory, we esti-

mate the cross-sectional widths of each neurite during skeletonization and calculate the geodesic distance between synapses to the cell body. We demonstrate results on 1,255 neuron and neuron fragments and achieve throughput of over one million voxels per second.

The field of connectomics has substantially evolved since White *et al.* published the first complete connectome in the mid-1980s. An undertaking that once required a dozen years of tedious manual labor from domain experts, deep learning models can now reconstruct teravoxel image volumes in a matter of weeks. The steady improvement of image acquisition pushes the field to consider reconstructing larger brain tissue volumes from more evolved species. This dissertation discusses four biologically-aware algorithms for compression, error correction, subgraph enumeration, and skeleton generation as a small step towards achieving this future goal of extracting a diverse set of wiring diagrams.

2

Compression

Segmented label volumes often require 64 bits per pixel, especially for teravoxel image volumes where the number of segments after automatic reconstruction can exceed 2^{32}. Uncompressed, these label volumes require 8× more storage than the raw image data. Compounding this disparity, multiple segmentations of the same image volume often exist as different algorithms are evaluated and potential errors flagged and corrected. Current pipelines use general-purpose compression schemes not tailored explicitly for connectomics label volumes. In this chapter, we present

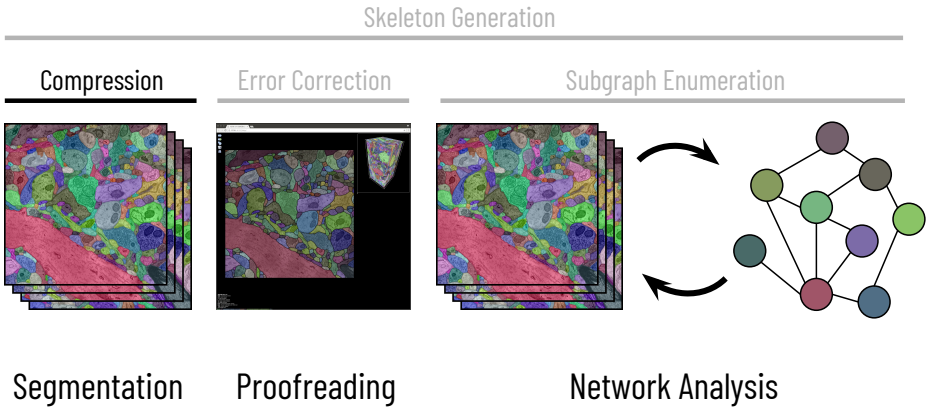


Figure 2.1: Multi-beam electron microscopes image one terabyte of image data every hour. Within six months, neuroscientists can extract a cubic millimeter of brain tissue which requires 10^{15} voxels. The reconstructed label volumes require 8 bytes per voxel. Uncompressed, storing these label volumes becomes infeasible for even relatively small brain samples. In this chapter, we discuss Compresso—a novel compression scheme tailored explicitly for these label volumes. Our method exploits the underlying properties of these label volumes, including the generally smooth borders between neurons.

Compresso, a compression scheme explicitly designed for connectomic label volumes. Our method exploits the underlying properties of the label volumes, including the generally smooth borders between neurons. We divide the entire label volume into a series of non-overlapping congruent 3D windows and perform a one-to-one mapping between the windows and a small set of integers. The high level of redundancy between windows enables us to compress the data by over 700× on most connectomic label volumes. Our methods extend to label volumes derived from other image modalities such as MRIs and natural images.

2.1 Motivation

Connectomics—reconstructing the wiring diagram of a brain at nanometer resolution—results in datasets at the scale of petabytes [38, 123, 141]. Machine learning methods find cell membranes and create cell body labelings for every neuron [32, 49, 89, 110] (Figure 2.2). These segmentations

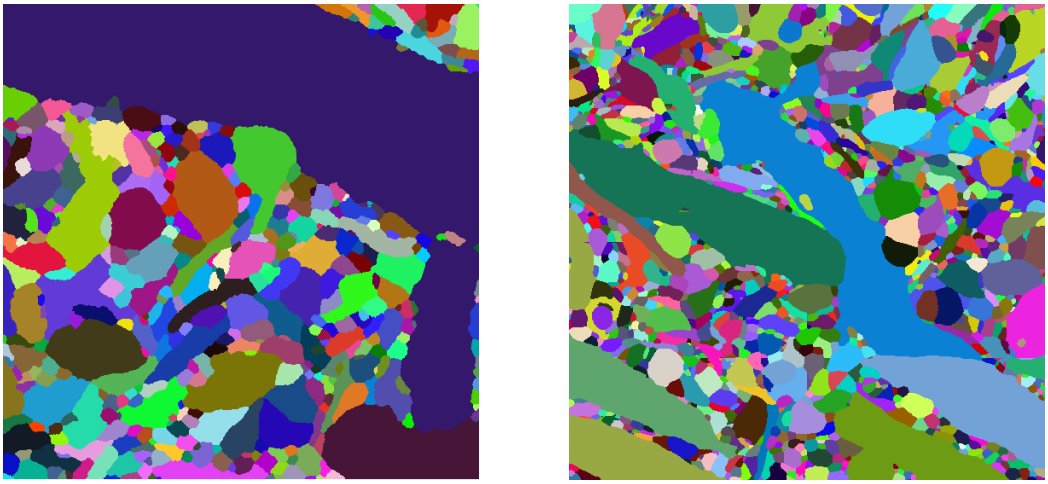


Figure 2.2: Here we see two examples of connectomics segmentation data with a different color per cell. These datasets now exceed terabytes, and even petabytes, in size.

are stored as label volumes that are typically encoded in 32 bits or 64 bits per voxel to support the labeling of millions of different neurons. Storing such data is expensive and transferring the data is slow. To cut costs and delays, we need compression methods to reduce data sizes.

The literature currently lacks efficient compression of label volumes. General-purpose compression schemes [4, 17, 22, 69, 90, 101, 118, 134, 148] are not optimized for this data. In this chapter, we exploit the typical characteristics of label volumes such as large invariant regions without natural relationship between label values. These properties render 2D image compression schemes inadequate since they rely on frequency reduction (using, e.g., wavelet or discrete cosine transform) and value prediction of pixels based on local context (differential pulse-code modulation) [108, 120]. Color space optimization strategies in video codecs [83] also do not affect label volumes, even though the spatial properties of a segmentation stack (z-axis) are similar to the temporal properties of video data (time-axis). A compression scheme explicitly designed for label volumes is part of the visualization software Neuroglancer [36]. This method exploits segmentation homogeneity by creating small blocks with N labels and reducing local entropy to $\log_2 N$

per pixel. Lookup tables then decode the values $[0, N)$ to the original 64-bit labels. We compare the Neuroglancer scheme with our method.

We explore the lossless compression of gigavoxel neuron segmentation volumes with high-bit encodings. We study and evaluate the performance of existing lossless compression methods, and their combinations, on multiple connectomics, magnetic resonance imaging (MRI), and general segmentation datasets. As our main contribution, we present Compresso—a novel compression method designed for label volumes using windowed feature extraction. Compresso yields compression ratios on label volumes 80% higher than the current best tools (Section 2.4). We release an open-source C++ implementation of our method with a Python wrapper.*

2.2 Related Work

General-purpose Compression. Compression tools are well studied and compared in regards to their compression rate, compression speed, and decompression speed [3, 65]. Classic encoders are zlib [22] (which utilizes LZ77 [147] and Huffmann codes [46]), LZ78 [148], LZF [69], LZO [90], LZW [134], bzip2 [3], and LZMA; LZMA is usually reported to provide very high compression rates. Recent efforts to optimize these approaches include Zopfli [128], Brotli [4], and Zstandard [17].

We evaluate several general-purpose methods on our datasets. However, these methods target any input data, resulting in compression that does not consider the typical characteristics of label volumes. Furthermore, some of these methods cannot handle the size of connectomics data (Section 2.4).

Image Compression. Many compression schemes for 2D images exist. The two most well-known lossless compression methods are PNG and JPEG2000 — optimized for photographs.

*https://github.com/VCG/compresso

These techniques rely on frequency reduction (using, e.g., wavelet or discrete cosine transform) and value prediction of images based on local context (differential pulse-code modulation). In contrast to photographs, segmentation data consists of large invariant regions with no natural relationship between their identifiers. Thus, we cannot apply these frequency reductions and value predictions to segmentation data. Therefore, off-the-shelf image compression techniques are not optimized and provide lower compression ratios. We still include PNG and JPEG2000 for evaluation.

Video Compression. We can map the z-axis of a segmentation stack to the time axis of video data to relate video compression to label volume compression. In practice, the change between slices in connectomics segmentation data is limited given natural constraints on neuronal shape and restrictions in between-slice resolution. The currently most widespread video compression is x264 which provides lossless encoding when quantization is disabled. Intuitively, this does not yield the best performance on segmentation data since any possible color space optimization is not applicable. Nevertheless, we include x264 for evaluation in Section 2.4.

Neuroglancer. Neuroglancer [36] is the closest current compression tool—a web-based 3D segmentation viewer that provides a compression scheme for segmentation data. The assumption behind the compression scheme is that, on average, small local areas will contain very few distinct segment identifiers with high probability. Based on this assumption, Neuroglancer divides the segmentation data into several small blocks. Consider an arbitrary block b with N distinct segment identifiers. Neuroglancer maps the identifier for each pixel into $[0, N)$ and stores a lookup table for $[0, N)$ to the original value. This mapping reduces the number of bits needed to encode a pixel in a given block to $\log_2 N$. The lookup table requires $64 * N$ bits of information to reconstruct the original data. To allow random access, Neuroglancer creates a header that stores the byte offset to the lookup table and encoded values for each block. We compare the performance

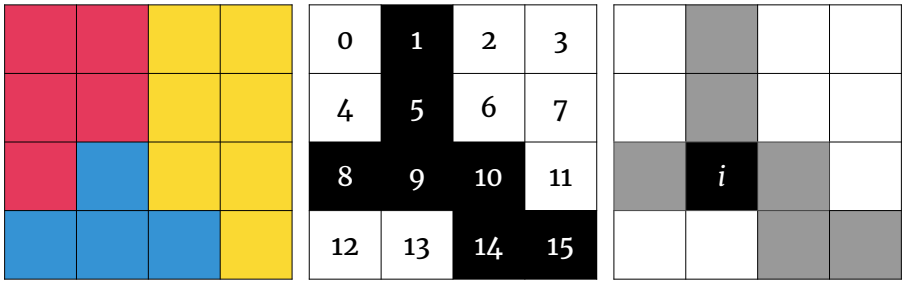


Figure 2.3: A $4 \times 4 \times 1$ pixel window where three unique labels meet (left). The boundary map for the same window, where dark pixels represent the boundary (center). This window has an encoded value of 50,978 $(2^1 + 2^5 + 2^8 + 2^9 + 2^{10} + 2^{14} + 2^{15})$. A boundary pixel i that is indeterminate and requires additional decoding information (right).

of Neuroglancer with our scheme in Section 2.4.

2.3 The Compresso Scheme

2.3.1 Encoding

Overview. Segmentation datasets contain two essential pieces of information across the image stack: per-segment shape and per-pixel label. Decoupling these two components allows for better compression on each.

Boundary Encoding. To encode the segment shapes, we consider the boundary pixels between two segments. Removing the per-pixel labels, we produce a boundary map for each slice where a pixel (x, y, z) is 1 if either pixel at $(x + 1, y, z)$ or $(x, y + 1, z)$ belongs to a different segment. The boundary map is divided into non-overlapping congruent 3D windows. If there are n pixels per window, each window w is assigned an integer $V_w \in [0, 2^n)$ where V_w is defined as:

$$V_w = \sum_{i=0}^{n-1} \mathbb{I}(i) 2^i, \tag{2.1}$$

and $\mathbb{I}(i)$ is 1 if pixel i is on a boundary and 0 otherwise. Figure 2.3 shows an example segmentation

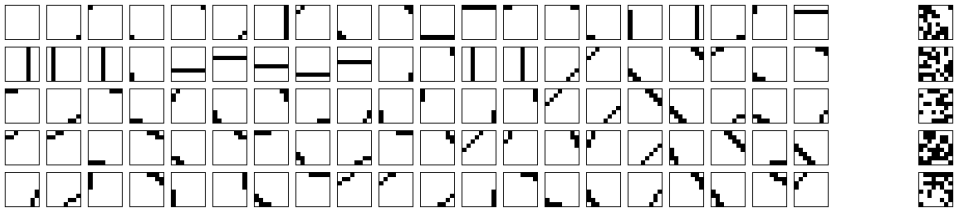


Figure 2.4: The 100 most frequent windows accounting for approximately 82% of the over 1.2 million V_w values on a representative connectomics dataset contrasted with 5 randomly generated windows. Each box represents an $8 \times 8 \times 1$ window where black pixels are boundary, and white pixels are non-boundary.

with a window size of $4 \times 4 \times 1$.

A priori, each window could take any of 2^n distinct values and therefore require n bits to encode without further manipulation. However, boundaries in segmentation images are not random, and many of these values never appear. Indeed, we find that a small subset of high-frequency V_w values accounts for most windows, allowing for significant compression. Figure 2.4 shows the 100 most common windows for a representative connectomics dataset. These 100 frequently occurring windows account for approximately 82% of the over 1.2 million V_w values in a representative dataset. Nearly all of these windows correspond to simple lines traversing through the window. In contrast, we also display five randomly generated windows that never occur in the dataset.

We define N as the number of distinct V_w representing all of the windows in an image stack. We construct an invertible function $f(V_w) \rightarrow [0, N)$ to transform the window values into a smaller set of integers. For all real-world segmentations $N \ll 2^n$; however, we assume no constraint on N in order to guarantee lossless compression. With this function, each V_w requires $\log_2 N$ bits of information to encode. So long as $N \leq 2^{n-1}$, we require fewer bits than the original binary boundary map. We create two arrays that store the per-segment shape encoding: `WindowValues[]` contains the value $f(V_w)$ for every window w and `ValueMapping[]` contains the reverse mapping from $[0, N) \rightarrow [0, 2^n)$ based on the function f. Long sequences of 0s in `WindowValues[]` are

reduced using run-length encoding.

Per-Pixel Label Compression. So far, we have focused exclusively on transforming the boundary map of a label volume. However, the per-pixel labels themselves are equally important. The boundary map divides each image slice into different segments. By design, all pixels in the same segment have the same label, so we store only one label per segment for each slice. We use a connected-component labeling algorithm to store one label per segment [42]. The algorithm labels all pixels clustered within a component m from M section labels. We store the original label for a segment m in slice z in `Labels_z[m]`. We concatenate these arrays for every image slice to create a variable `Labels[]`.

Exceptions. Thus far, we have assumed the boundaries described in Section 2.3.1 provide enough information to reconstruct the entire segmentation. We easily relabel pixels not on a segment boundary using the `Labels[]` array. However, we require more care for pixels on the segment boundaries. Consider Figure 2.3, which depicts a difficult boundary to decode. If a boundary pixel has a non-boundary neighbor to the left or above, then that pixel merely takes on that neighbor's value. However, the pixel i requires more care since its relevant neighbors are both boundary pixels. If a non-boundary neighbor pixel shares a label with the undetermined pixel, we add the offset to that neighbor to an array `IndeterminateValues[]`. Otherwise, we add that per-pixel label.

Metadata. We construct a data structure containing the two per-segment shape and two per-pixel label arrays. The last component of the data structure is the `Header`, which contains the dimensions of the original data, the window size, and the arrays' size. Compresso could be improved by further compressing the individual components of the encoding (e.g., Huffman encoding the V_w values). We achieve strong overall compression using a second-stage general compression scheme such as LZMA (Section 2.4).

2.3.2 Decoding

The first step in decoding the data is to reconstruct the boundary map. We iterate over every pixel, determine the corresponding window w, and retrieve the encoded window value $f(V_w)$ from the `WindowValues[]` array. These values range from 0 to $N-1$ and correspond to an index in `ValueMapping[]` that contains the original V_w value. After decoding V_w, the value of pixel i in window w equals $V_w \wedge 2^i$.

After reproducing the boundary map, we execute the same deterministic connected-components algorithm per slice as when encoding. Each component in the boundary map receives a label between 0 and $M-1$. Using the `Labels[]` array, we can easily translate these component labels into the original per-pixel labels for every slice. We iterate over the entire dataset in raster order to determine the per-pixel labels for every boundary pixel. Any boundary pixel (x, y, z) with a non-boundary neighbor at $(x-1, y, z)$ or $(x, y-1, z)$ shares the same per-pixel label. If both relevant neighbors are boundaries, we consider the next unused value in the `IndeterminateValues[]` array and update this pixel's label.

2.3.3 Complexity

In what follows, P is the number of input pixels; N is the number of distinct window values; X, Y and Z are the sizes of the x, y, and z dimensions of the input data; and α is the inverse Ackermann function [31].

Encoding. Extracting the boundaries from the segmentation, generating the V_w values, and populating the `IndeterminateValues[]` array are all linear work in P. The N unique window values are sorted to create the `ValueMapping` variable. Generating the `Labels[]` array requires running a connected-component labeling algorithm over each z slice; we use a union-find data structure with union by rank and path compression optimizations. The overall complexity of the compres-

Table 2.1: For evaluation, we use the following publicly available datasets. Segmentations were obtained using a combination of U-net [110] and watershed, semi-automatic, or manually. Compresso paired with LZMA yields the best compression ratio on all datasets indicated by an asterisk (*). Neuroglancer paired with LZMA achieved the best compression ratio only for the SPL Brain Atlas (724x).

Dataset	Size	Segmentation	Compression + LZMA	
			Speed (Com./Dec.) (↑ / ↑)	Compression Ratio (↑)
AC3 Subvolume mouse cortex, EM	$1024 \times 1024 \times 150$ vx ($6 \times 6 \times 30$ nm^3/vx)	U-net	100 / 209 MB/s	814× *
AC4 Subvolume mouse cortex, EM	$1024 \times 1024 \times 100$ vx ($6 \times 6 \times 30$ nm^3/vx)	U-net	105 / 218 MB/s	701× *
L. Cylinder [55] mouse cortex, EM	$2048 \times 2048 \times 300$ vx ($3 \times 3 \times 30$ nm^3/vx)	U-net	103 / 180 MB/s	952× *
CREMI A, B, C drosophila brain, EM	$1250 \times 1250 \times 125$ vx ($4 \times 4 \times 40$ nm^3/vx)	U-net	110 / 218, 118 / 243, 110 / 219 MB/s	857× *, 1239× * 960× *
SPL Brain Atlas T1/T2-weighted MRIs	$256 \times 256 \times 256$ vx ($1 \times 1 \times 1$ mm^3/vx)	Semi-autom.	85 / 254 MB/s	636×
SPL Knee Atlas MRI	$512 \times 512 \times 119$ vx ($0.28 \times 0.28 \times 1$ mm^3/vx)	Semi-autom.	136 / 244 MB/s	1553× *
SPL Abdominal Atlas CT	$256 \times 256 \times 113$ vx ($0.94 \times 0.94 \times 1.5$ mm^3/vx)	Semi-autom.	91 / 254 MB/s	480× *
BSD500 Segmentation Challenge	321×481, 2696 images	Manual	110 / 187 MB/s	1188× *
PASCAL VOC 2012 Challenge	Varying, 2913 images	Manual	146 / 222 MB/s	2217× *

sion scheme is therefore $O\left(P(1+\alpha(XY)) + N\log N\right)$.

Decoding. Decoding the window values, reconstructing the boundary map, and applying the correct per-pixel labels for all boundary pixels using the array `IndeterminateValues[]` are all linear work in P. Reconstructing the per-pixel labels requires running the connected-component labeling algorithm over every image slice. The overall complexity of the decompression scheme is therefore $O\left(P(1+\alpha(XY))\right)$.

2.4 Evaluation and Results

We consider the following compression schemes: Compresso, Neuroglancer, BZip2, Zlib, LZ78, LZF, LZMA, LZO, LZW, Zstandard, PNG, JPEG2000, and X.264. In addition to these standalone compression schemes, we consider all pairs with a first stage encoding using either Compresso or Neuroglancer and a second stage using one of the general-purpose algorithms. Both

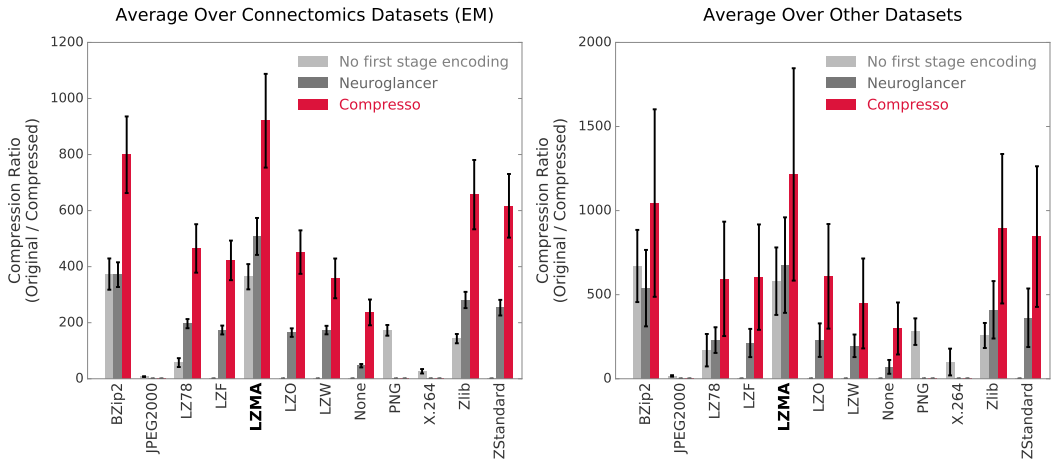


Figure 2.5: Compression ratios of general-purpose compression methods combined with Compresso and Neuroglancer. Compresso paired with LZMA yields the best compression ratios for all connectomics datasets and on average (four out of five) for the others.

Compresso and Neuroglancer leave some redundancies that a general-purpose compressor can optimize; such multi-stage schemes are common in image compression. Table 2.1 presents six connectomics, three MRI, and two image segmentation datasets used for evaluation. Compresso works for any arbitrary 2D and 3D window dimensions. We achieve the results in this section using an 8x8x1 window.

The combination of Compresso and LZMA provides superior compression on all connectomics datasets (Table 2.1). Figure 2.5 shows the compression ratios for every compressor on segmentation data. For example, Compresso achieves a compression ratio of over 950x on *L.Cylinder* reducing the 10-gigabyte volume to 10.5 megabytes. LZMA performs very well by itself and pairs with any encoding strategy. X.264 performs surprisingly poorly on these datasets, in part because of our requirement of lossless compression. It performs better when information loss is tolerated; however, it still does not surpass the more specialized encoding schemes. These observations also hold for JPEG2000 and PNG. Compresso with LZMA outperforms all other existing methods on connectomics datasets by 80%.

The fundamental principles guiding Compresso are valid for a diverse set of segmentation datasets (Figure 2.5, right). We evaluate our compression scheme's performance on three MRI and two image segmentation datasets to demonstrate additional potential use cases. Compresso followed by LZMA compresses the MRI datasets reasonably well, particularly on the SPL Knee Atlas, which contains highly redundant boundary segments. The Berkeley Segmentation and PASCAL Visual Object Class datasets are two very common benchmarks in image segmentation [27, 75]. These datasets currently use GZIP and PNG compression, but Compresso with LZMA can improve the storage required by a factor of over 10x and 5x, respectively.

To demonstrate usability for even larger label volumes, we compress a 100 μm^3 label volume. Uncompressed, the 2.08 trillion voxel label volume requires 19.25 terabytes of storage. After using Compresso with LZMA, we reduced the size of the data by 742× to 25.94 gigabytes. At the time, this reduction in disk space would reduce the potential storage costs on Amazon Web Services from $442.75 to $0.60 per month.

In terms of speed, Compresso is on par with Neuroglancer across all datasets and achieves a throughput of 112.16 megabytes per second ($SD = 18.62$ MB/s) for compression and 222.85 megabytes per second ($SD = 32.14$ MB/s) for decompression. All experiments ran on a single core of an Intel Xeon 2.3GHz CPU.

2.5 Conclusions

We have introduced Compresso, an efficient compression tool for segmentation data that outperforms existing solutions on connectomics, MRI, and other segmentation data. We use existing knowledge of the shape of neuronal membranes, namely that they are generally smooth, as a guiding principle for Compresso. Although this insight stemmed from our biological priors, we believe that this generally holds for a wide variety of segmentation data sets. Thus, we expect

Compresso to generalize well to other label volumes.

We plan to improve random access to lower memory requirements for online viewers and enhance the compression of the metadata in the future. Also, we will integrate Compresso into our analysis pipeline and various end-user applications. To encourage testing of our tool, replication of our experiments, and adoption in the community, we release Compresso and our results as free and open research at github.com/VCG/compresso.

3

Error Correction

Early endeavors in connectomics relied on the tedious manual reconstruction of the image stacks to segment neurons. More recently, automatic methods segment the volumes with increasing accuracy, surpassing estimated human accuracy on small challenge datasets. However, these techniques still produce errors that we must correct before we can extract an accurate wiring diagram. Most segmentation methods follow a two-stage framework where a convolutional neural network (CNN) predicts affinities and a watershed transform clusters pixels into basins. Next, an agglom-

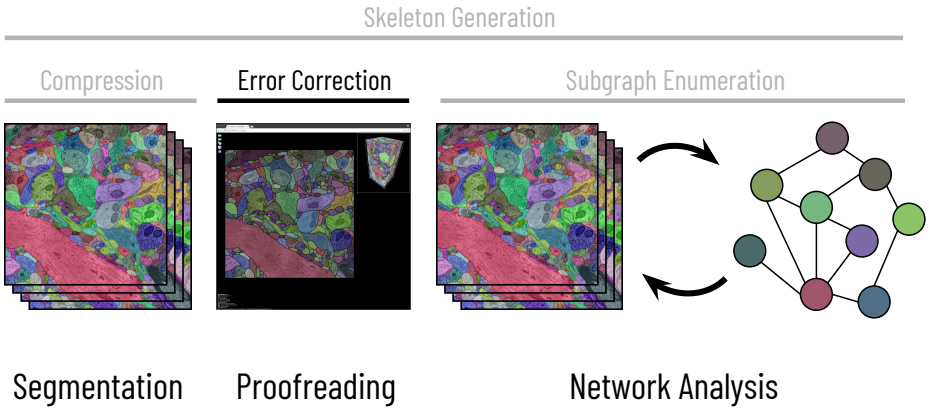


Figure 3.1: Automatic reconstruction techniques have significantly improved in the last decade, surpassing human accuracy on some challenge datasets. Despite these successes, these methods still produce errors at scale. In this chapter, we explore one such error, the *split error*. We reformulate the error correction of these split errors as a multicut graph partitioning problem. Since graph partitioning is a computationally expensive process, we use biological constraints to simplify our graph. Our hand-designed geometric constraints and machine-learned morphologies improve computational performance as we improve accuracy over existing state-of-the-art methods by 21.3%.

eration step merges the superpixels into more significant segments. We propose a third step in this framework that takes an oversegmentation and reformulates the problem as a graph partitioning one. When constructing our graph, we use biological priors through hand-designed geometric constraints and machine-learned morphologies to improve accuracy and increase throughput. The reformulation allows us to use both local, when constructing the graph, and global, when partitioning the graph, context to improve on existing segmentations. We evaluate our methods on four connectomic datasets and achieve an average improvement on existing state-of-the-art methods by 21.3%.

3.1 Motivation

By studying connectomes–wiring diagrams extracted from the brain containing every neuron and the synapses between them–neuroscientists hope to understand better certain neurological

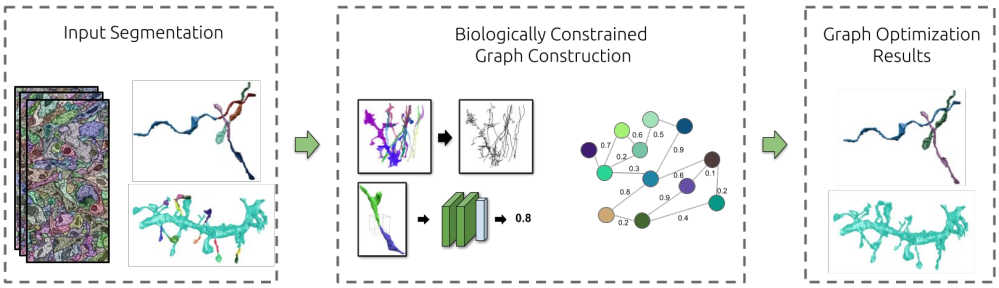


Figure 3.2: Most current state-of-the-art segmentation pipelines consist of affinity generation with watershed transform and region merging (left). We follow these existing methods by constructing a graph derived from their segmentation by enforcing geometric constraints inspired by the underlying biology and learning typical neuronal morphologies (center). Our graph formulation allows us to partition the graph with a global optimization strategy to improve the segmentation (right).

diseases, generate more faithful models of the brain, and advance artificial intelligence [39, 43]. To this end, neuroscientists produce high-resolution images of brain tissue with electron microscopes where every synapse, mitochondrion, and cell boundary is visible [55]. Since these datasets now exceed a petabyte in size, manual tracing of neurons is infeasible, and automatic segmentation techniques are required.

Current state-of-the-art automatic 3D reconstruction approaches often use pixel-based CNNs and watershed transforms to generate an initial oversegmentation [66, 110, 149], followed by region merging steps [32, 60, 67, 88, 98]. Flood-filling networks combine these two steps into one by gradually expanding segments from a seed voxel [49]. However, all of these above strategies make decisions using only the local context and do not consider the global ramifications to individual merges. Therefore, a small number of compounding merge errors can create an undersegmentation with several neuronal processes labeled as one neuron. Since correcting such *merge errors* is computationally challenging, current methods typically favor oversegmentation, where a neuronal process is segmented into multiple labels. Unfortunately proofreading these *split errors*, while easier, still remains onerous [96].

We propose a third step for connectomics reconstruction workflows to refine these oversegmentations and close the gap between automatic and manual segmentation. We reformulate the region merging problem as a graph partitioning one to leverage global context during the agglomeration process. Thus far, the computational burden associated with global optimization strategies remains their biggest drawback despite some research into parallelizing the computation [7]. Performing the graph partitioning step after an existing agglomeration technique allows us to capture larger shape context when making decisions. Furthermore, the amount of computation significantly decreases as the input method correctly segments many supervoxels. The remaining *split errors* typically occur in places where a neuronal process becomes relatively thin or the corresponding image data noisy—difficult locations to reconstruct using only the local context from images and affinities.

When constructing our graph, we employ geometric constraints guided by the underlying biological morphology to reduce the number of nodes and edges. Due to their biological nature, oversegmented regions should be connected with specific geometric and topological properties in mind. For example, among other biological considerations, L-shaped junctions and arrow-shaped junctions are rare in neuronal structures. We can both use and learn these shape priors to produce a more accurate region merging strategy.

Our region merging framework consists of several steps to construct a graph from an input segmentation and then partition the graph using a global optimization strategy (Figure 3.2). We first identify segments that are oversegmented based on our knowledge of the span of neuronal processes and use a trained CNN to merge these segments with larger ones nearby. The remaining segments receive a node in our graph. We then generate skeletons for each segment to produce a simple yet expressive representation of a given segment's underlying shape (Figure 3.2, center). We identify potential segments to merge from these skeletons, which will receive a corresponding edge in the graph. Another CNN classifier learns the local structural shapes of neurons and pro-

duces probabilities that two segments belong to the same neuron. Finally, we employ a graph optimization algorithm to partition the graph into an improved reconstruction (Figure 3.2, right). Our graph formulation creates a formal description of the problem enabling a diverse range of optimization strategies in the future.

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

3.2 Related Work

Initial Pixel-based Segmentation Methods. There are two main approaches to segmenting electron microscopy images at the voxel level. In the first, 2D or 3D convolutional neural networks are trained to produce an intermediate representation such as boundary [16, 47, 60, 110] or affinity maps [66, 127]. Advancements in architecture designs (e.g., 3D U-Net [15]), model averaging techniques [143], segmentation-specific loss functions (e.g., MALIS [10]), and data augmentation strategies [67] have greatly improved the results for these intermediate representations. Afterwards, clustering techniques such as watershed [19, 28, 149] or graph partition [5] transform these intermediate representations into a segmentation. In the second approach, neural networks [49, 81] are trained recursively to grow the current estimate of a binary segmentation mask, which is further extended to handle multiple neurons [82]. Despite impressive segmentation accuracy, this approach's computational burden remains a limitation as the network needs to infer each segment separately.

Agglomeration Strategies. Agglomeration methods are parameterized by the similarity metric

between adjacent segments and merging strategy. For the similarity metric, Lee *et al.* [67] and Funke *et al.* [32] rely solely on the predicted affinities and define the metric as the mean affinity between segments. Classification-based methods generate the probability to merge two segments from handcrafted [48, 60, 88, 97, 149] or learned features [9]. Niko *et al.* [64] use the information about post- and pre-synaptic connections to refine the multicut algorithm and prevent axons and dendrites from merging. Most methods use variants of hierarchical agglomeration for the merging strategy [60, 88, 97, 98, 149] to merge a pair of regions at a time greedily. Other methods formulate agglomeration as reinforcement learning [48] and superpixel partitioning problems [7]. More recently, flood-filling networks [49] use different seeding strategies with the same network from the initial segmentation step to agglomerate regions.

Error-correction Methods. Although significant advancements in the above methods produce impressive results, there are still errors in the segmentation. These errors are corrected either manually with human proofreading [38, 59] or automatically [150]. Since correcting errors is a computationally expensive task, various research explores how to use machine learning to improve human efficiency [40], automatic detection of error regions [109, 150], or reduce the search space via skeletonization [23]. However, these methods rely only on local context for decision-making and do not enforce biological constraints on their corrections.

3.3 Biologically-Constrained Graphs

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

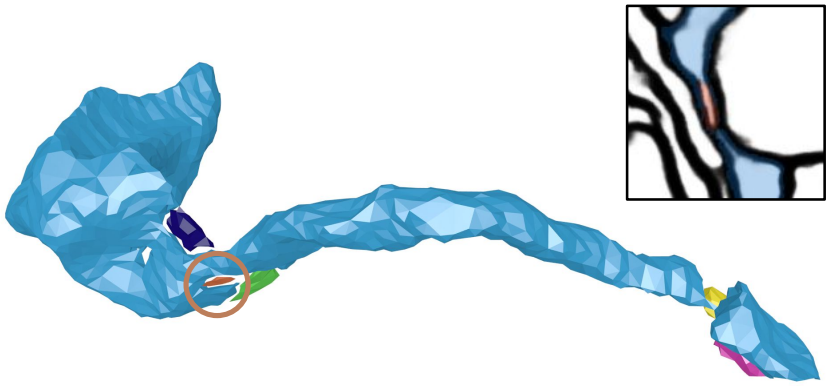


Figure 3.3: The above neuronal process is incorrectly segmented into several labels. Five of the segments are very small, indicating that they must merge with a nearby larger segment. Frequently these tiny segments are artifacts of noisy affinities around locations where a process becomes quite thin.

aid in the graph generation process.

3.3.1 Node Generation

Current pipelines that agglomerate regions based on the affinity predictions alone produce a large number of tiny segments (e.g., 86.8% of the segments produced by the waterz algorithm on a representative dataset contain fewer than 9,600 voxels corresponding to a volume of approximately $0.01\,\mu m^3$). Since these strategies use only the mean affinity between two supervoxels, noise in the affinity generation process produces these minor artifacts. In particular, these segments frequently occur in regions where a neuronal process becomes relatively thin, leading to low affinities between voxels (Figure 3.3). We can leverage additional information about the underlying biology to identify and correct these segments: neurons are expansive and should not contain few voxels when segmented. Figure 3.3 shows an example neuronal process oversegmented into six distinct components, five of which are relatively small. Each of these segments had sufficiently low mean affinities with its neighbors.

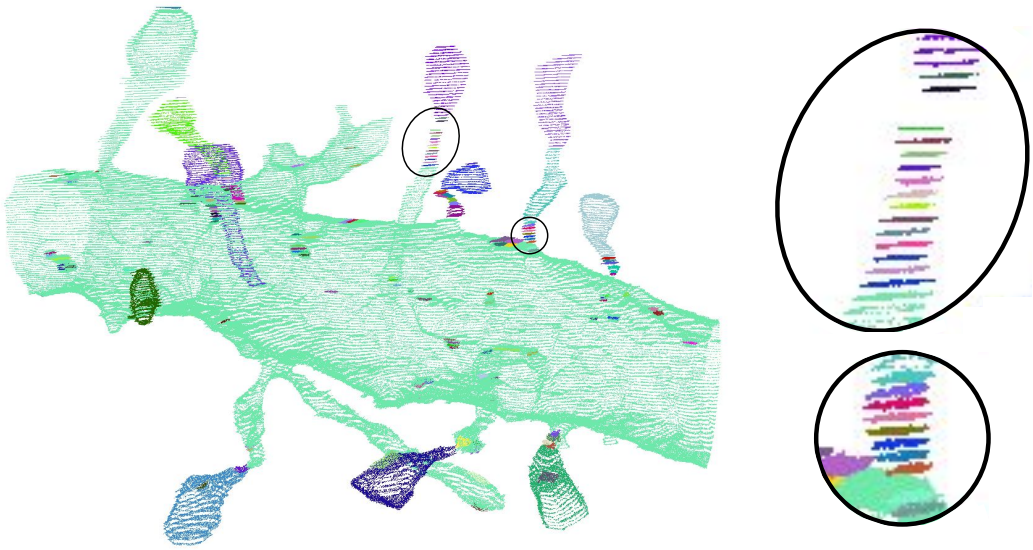


Figure 3.4: Some automatic segmentation methods tend to produce a series of singleton segments fully contained within a z slice. These slices are most common along dendritic spines. Correcting these singletons is critical to ensure a connection between the end of a spine, where synapses frequently occur, and the primary neuronal process.

We identify these small segments and merge them before graph construction to reduce the number of nodes (and consequently edges). We flag any segment whose volume is less than t_{vol} cubic microns as *small* and create a list of nearby *large* segments as potential merge candidates. The simplest method to absorb these segments is to agglomerate them with a non-flagged neighbor with the highest mean affinity. However, these segments arise because of inaccuracies in the affinities. We employ two methods to merge these nodes based on the geometry of the small segments themselves. Some agglomeration strategies produce several "singleton" segments that are entirely contained within one image slice (Figure 3.4). We link these singletons together across several slices by considering the Intersection over Union when superimposing two adjacent slices. Second, we train a neural network to learn if two segments, one small and the other large, belong to the same neuron.

Looking at the local shape around two segments can provide important information over raw

image data or affinities alone. Often split errors occur at regions with either image artifacts or noisy affinities; however, the segment shapes provide additional information. We extract a small cube with diameter d_{node} nanometers around each *small–large* segment pair. We train a feed-forward 3D CNN to learn the neuron morphology and predict which pairs belong to the same neuron. The CNN takes as input three channels corresponding to if the voxel belongs to the small segment, the large segment, or either segment (Figure 3.7). Our network contains three *VGG-style* convolution blocks [13] and two fully connected layers before a final sigmoid activation. The network parameters are further discussed in Section 3.4.2. Each small segment is merged with exactly one nearby large segment to prevent a merge error from connecting two distinct neurons via a shared small segment neighbor.

3.3.2 Edge Generation

Each remaining segment in the volume has a large number of adjacent neighbors (28 per segment averaged over three gigavoxel datasets). We use a geometric prior on the split errors to reduce the number of considered errors greatly. Most split errors follow one of two modalities: either a neuronal process is split into two or more parts across its primary direction (Figure 3.5, top) or several spines are broken off a dendrite (Figure 3.5, bottom).

We generate skeletons for each segment to create a simple yet expressive representation of a volume's underlying shape. For example, this approach allows us to quickly identify all of the dendritic spines in a segment with minimal computation (Figure 3.6). Some previous research focuses on the development and use of skeletons in the biomedical and connectomics domains for quicker analysis [114, 144] and error correction [23]. Topological thinning and medial axis transforms receive a significant amount of attention in the computer graphics and volume processing communities [68, 94].

We first downsample each segment using a max-pooling procedure to a resolution of X_{res}, Y_{res},

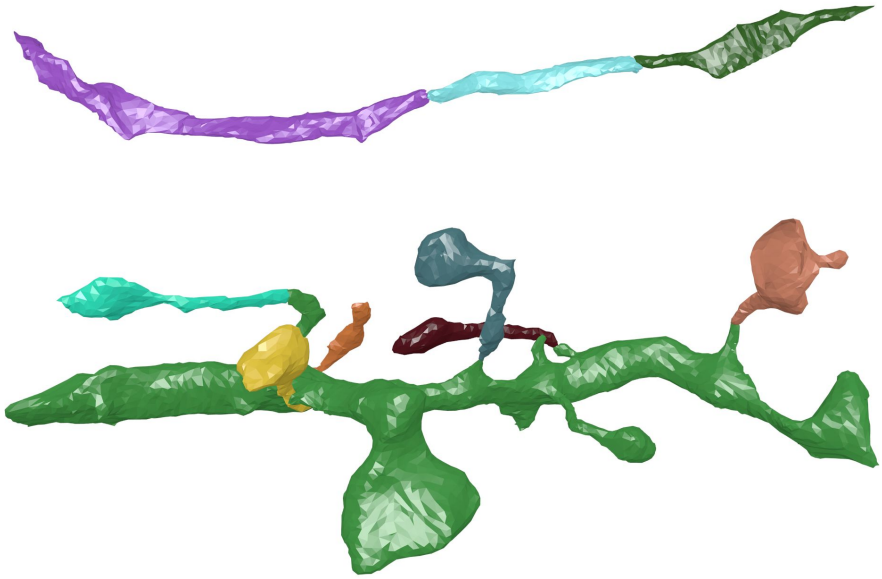


Figure 3.5: Here we see two typical instances of split errors in connectomics segmentations. In the top image, the neuronal process is split multiple times at some of its thinnest locations. On the bottom, multiple spines are split from the dendrite.

and Z_{res} nanometers before generating the skeletons. This process does not cause significant detail loss since the finest morphological features of neurons are on the order of 100 nm [109]. Instead, the produced skeletons follow the underlying geometry more closely when downsampling since these segments' boundaries can be exceptionally noisy. We use a sequential topological thinning algorithm [92] to gradually erode the boundary voxels for each segment until only a skeleton remains. Figure 3.6 shows two example segments with their corresponding skeletons. The larger spheres in the skeleton correspond to endpoints. We generate a vector at each endpoint to indicate the direction of our skeleton before endpoint termination.

When generating the edges for our graph, we exploit the aforementioned split error modalities that follow from neurons' underlying biological structure. To identify these potential *split error* locations, we use the directional vectors at each skeleton endpoint. For each endpoint v_e in a given segment S_e we consider all voxels v_n within a defined radius of t_{edge} nanometers. If that

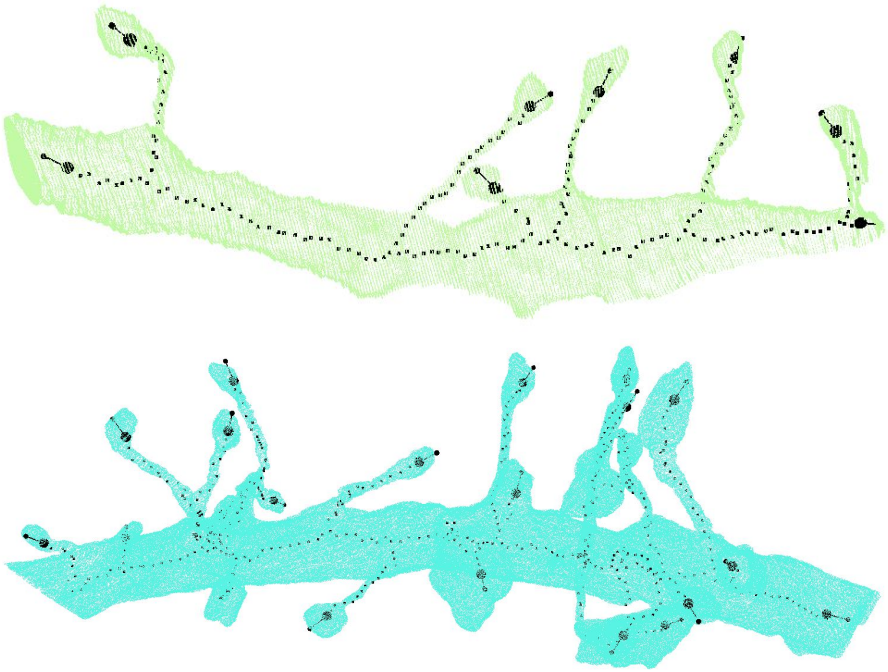


Figure 3.6: Here are two example skeletons produced by a topological thinning algorithm [92]. The larger spheres represent endpoints, and the vectors protruding from them show the skeleton's direction at endpoint termination.

voxel belongs to another segment S_n that is locally adjacent to S_e and the vector between v_e and v_n is within θ_{max} degrees of the directional vector leaving the skeleton endpoint, nodes S_e and S_n receive an edge in the graph. θ_{max} is set to approximately 18.5°; this value follows from the imprecision of the endpoint vector generation strategy.

3.3.3 Edge Weights

To generate the merge probabilities between two segments, we use a CNN similar to the one discussed in Section 3.3.1. We extract a small cube of diameter d_{edge} nanometers around each potential merge location found in the edge generation step. Again, we train a new feed-forward 3D CNN with three channels encoding whether a voxel belongs to each segment or either (Figure 3.7). The network follows the same general architecture with three *VGG-style* convolution

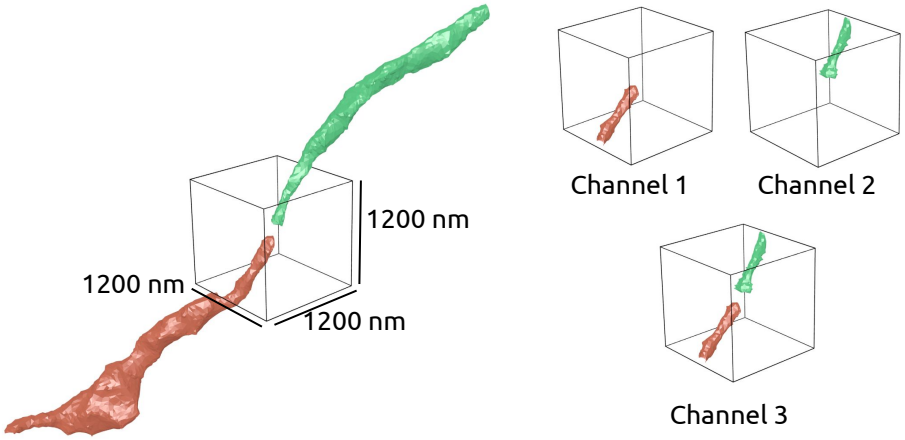


Figure 3.7: Both the node and edge networks take three channels as input corresponding to if a particular voxel belongs to segment one, segment two, or either segment. This particular example is input to the edge CNN to determine if two segments belong to the same neuronal process.

layers followed by two fully connected layers and a final sigmoid activation.

We next convert these probabilities into edge weights with the following scheme [57]:

$$w_e = \log \frac{p_e}{1 - p_e} + \log \frac{1 - \beta}{\beta} \tag{3.1}$$

where p_e is the corresponding merge probability and β is a tunable parameter that encourages over- or undersegmentation. Note that high probabilities transform into positive weights. This requirement follows from our optimization strategy (discussed below), which minimizes an objective function and should collapse all positively weighted edges.

3.3.4 Graph Optimization

Our graph formulation enables us to apply a diverse range of graph-based global optimization strategies. Here, we reformulate the partitioning problem as a multicut one. There are two pri-

mary benefits to this minimization strategy: first, the final number of segments depends on the input and is not predetermined; second, the solution is globally consistent (i.e., a boundary remains only if the two corresponding nodes belong to different segments) [57].

We use the greedy-additive edge contraction method to produce a feasible solution to the multicut problem [57]. Following their example, we use the more general lifted multicut formulation where all non-adjacent pairs of nodes receive a "lifted" edge and a corresponding edge weight indicating the long-range probability that two nodes belong to the same neuron. Ideally, these weights perfectly reflect the probability that two nodes belong to the same neuron by considering all possible paths between the nodes in the graph. Unfortunately, such computation is expensive, so we create a lower estimate of the probability by finding the shortest path on the negative log-likelihood graph (i.e., each original edge weight w_e is now $-\log w_e$) and setting the probability equal to e raised to the distance [57].

3.4 Experimental Set Up

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

3.4.1 Datasets

We evaluate our methods using four datasets with different resolutions, acquisition techniques, and input segmentation strategies (Table 3.1). The PNI volumes were given to us by the authors of [150] and contain nine separate volumes imaged by a serial section transmission electron microscope. We use four of these volumes to train our networks and tune parameters, three for validation, and the last two for testing. These image volumes have an initial segmentation produced by a variant of a 3D U-Net followed by zwatershed and mean agglomeration [67].

Table 3.1: We show results on four testing datasets, two from the PNI volumes, one from the Kasthuri volume, and one on the SNEMI3D challenge dataset. We use four PNI volumes for training and three for validation. We further finetune our neural networks on separate training data for both the Kasthuri and SNEMI3D volumes.

Dataset	Brain Region	Sample Resolution	Dimensions	Segmentation
PNI	Primary Visual Cortex	$3.6 \times 3.6 \times 40\,\text{nm}^3$	$2048 \times 2048 \times 256$	Zwatershed and Mean Agg [67]
Kasthuri	Somatosensory Cortex	$6 \times 6 \times 30\,\text{nm}^3$	$1335 \times 1809 \times 338$	Waterz [32]
SNEMI3D	Somatosensory Cortex	$3 \times 3 \times 30\,\text{nm}^3$	$1024 \times 1024 \times 100$	Waterz [32]

The Kasthuri dataset is freely available online* and represents a region of the neocortex imaged by a scanning electron microscope (SEM). We divide this volume into training and testing blocks. We initially use a 3D U-Net to produce affinities and agglomerate with the waterz algorithm [32].

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

3.4.2 Parameter Configuration

Here we provide the parameters and CNN architectures discussed in Section 3.3. Appendix A provides additional experiments that explore each of these parameters and network architectures in further detail.

Node Generation. To determine a suitable value for t_{vol}—the threshold to receive a node in the graph—we consider the edge generation step, which requires expressive skeletons. Skeletons generated through gradual boundary erosion [92] tend to reduce small segments to a singular point removing all relevant shape information. After exploring various threshold values on four training datasets we set $t_{vol} = 0.010\,36\,\mu\text{m}^3$.

Skeletonization Method. To evaluate various skeleton generation approaches, we create and

*https://neurodata.io/data/kasthuri15/
†http://brainiac2.mit.edu/SNEMI3D/home

publish a skeleton benchmark dataset.[‡] We evaluate three different skeleton approaches with varying parameters on this benchmark dataset [68, 92, 114]. Downsampling the data to 80 nanometers in each dimension followed by a topological thinning algorithm [92] produces the best results.

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

CNN Training. Of the nine PNI datasets, we use four for training and three for validation. We experimented with various network architectures and input cube sizes. Our node network receives a cube with $d_{node} = 800$ nm which is then sampled into a voxel grid of size $(60, 60, 20)$. Our edge network receives a cube with $d_{edge} = 1200$ nm which is similarly sampled into a voxel grid of size $(52, 52, 18)$

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

3.4.3 Error Metrics

We evaluate the performance of the segmentations using the variation of information (VI) [80]. The split and merge variation of information scores quantify over- and undersegmentation, respectively, using conditional entropy. The sum of the two entropies gives the total variation of information. For our CNNs, a true positive indicates a corrected split error and a false positive a

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

Table 3.2: Our proposed method reduces the total variation of information by 20.9%, 28.7%, 15.6%, and 19.8% on four testing datasets. The variation of information split decreases significantly, achieving a maximum reduction of 45.5% on the second PNI testing dataset.

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

merge error introduction.

3.5 Results

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

3.5.1 Benchmark Comparison

Table 3.2 shows the total VI improvement of our method over our input segmentations on four test datasets. We reduce the total VI on the two PNI, Kasthuri, and SNEMI3D datasets by 20.9%, 28.7%, 15.6%, and 19.8% respectively. Our VI split scores decrease by 34.5%, 45.5%, 23.8%, and 23.3% on the four datasets. Our proposed method only merges segments and does not divide any into multiple components, and thus our VI merge scores can only increase. However, our input segmentations are very oversegmented and have a small VI merge score at the start. Our algorithm increases the VI merges (i.e., it makes some wrong merge decisions), but the overall decrease in VI split overcomes the slight increases in VI merge. On the SNEMI3D dataset, we generate multiple baselines and proposed segmentations by varying the merging threshold in the waterz algorithm. We show the results on the best baseline compared to the best-corrected segmentation, and thus the VI merge can decrease for this dataset.

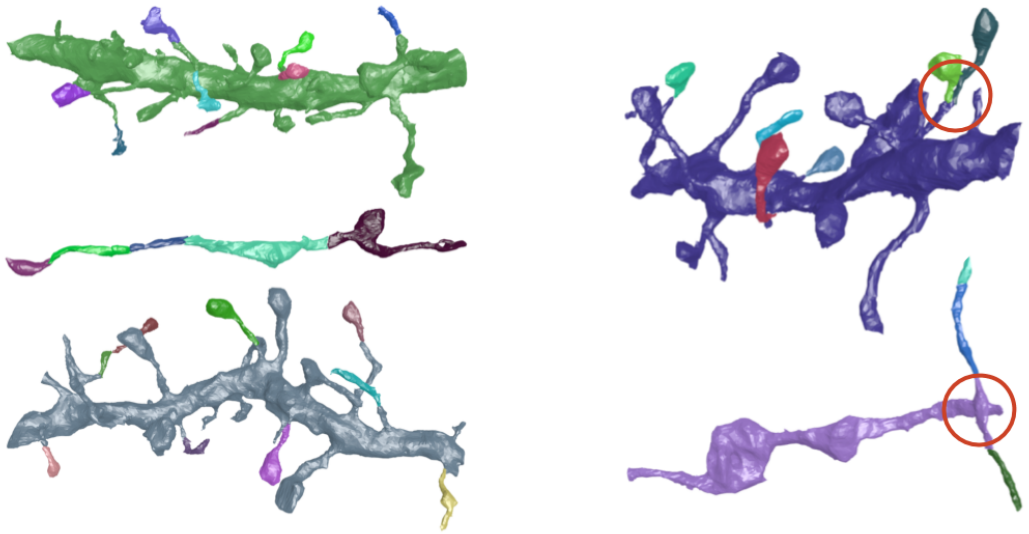


Figure 3.8: Here we show three success (left) and two failure (right) cases for our proposed methods. On the left, we see two dendrites with eight spines, each correctly merged. Correcting these splits errors is particularly essential for extracting the wiring diagram since synaptic connections occur on the spines. In between these examples, we show a typical neuronal process initially split at numerous thin locations. Circled on the top right is an incorrectly merged spine to the dendrite. We correctly connect five spines, but we accidentally merge two spines to the same location once. Below is an example where a merge error in the input segmentation causes an error.

Figure 3.8 shows five examples from our proposed method, three correct (left) and two failures (right). Here, we see two example dendrites, each with eight spines, correctly reconnected to the neuronal process. Fixing these types of split errors is crucial for extracting the brain's wiring diagram: electrical signal from neighboring cells is propagated onwards through post-synaptic densities located on these spines. We show a typical neuronal process between these two dendrites split into multiple segments at locations where the process becomes relatively thin. Our edge generation step quickly identifies these locations as potential split errors, and our CNN predicts that the neuronal process is continuing and not terminating. On the top right, we show an example dendrite where we correctly merge five spines. However, in one location (circled), we accidentally merge one additional spine causing a merge error. Below that, we show an error caused by a merge error in the input segmentation. The purple neuronal process is incorrectly merged

Table 3.3: Our proposed node generation strategy that merges small segments into nearby larger ones outperforms the baseline strategy. In our best instance, we correctly merge 444 small segments while only incorrectly merging 75.

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

at one location with a perpendicular traversing process (circled). We merge other segments with the perpendicular process causing an increase in VI merge.

3.5.2 Empirical Ablation Studies

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

Node Generation. Table 3.3 summarizes the success of our node generation strategy in terms of correctly merging small segments to larger ones from the same process. We compare our results against the following simple baseline: how many small labels are correctly merged if they receive the same label as the adjacent large segment with which it shares the highest mean affinity. Our method significantly outperforms the baseline on the PNI datasets. The baseline performs poorly as expected since the input segmentation agglomeration strategy initially opted not to merge these small segments based on the affinities alone. In each case, we correctly merge between 76 and 85% of small segments. The waterz agglomeration strategy produces many more small segments than the mean agglomeration method. Interestingly, the baseline is much higher for this strategy, indicating that a simple post-processing method of merging small segments based on a thresholded affinity might be justified.

Edge Generation. There are two main components to edge generation: skeletonization and location of potential split errors. We created a skeleton benchmark dataset for connectomics segmen-

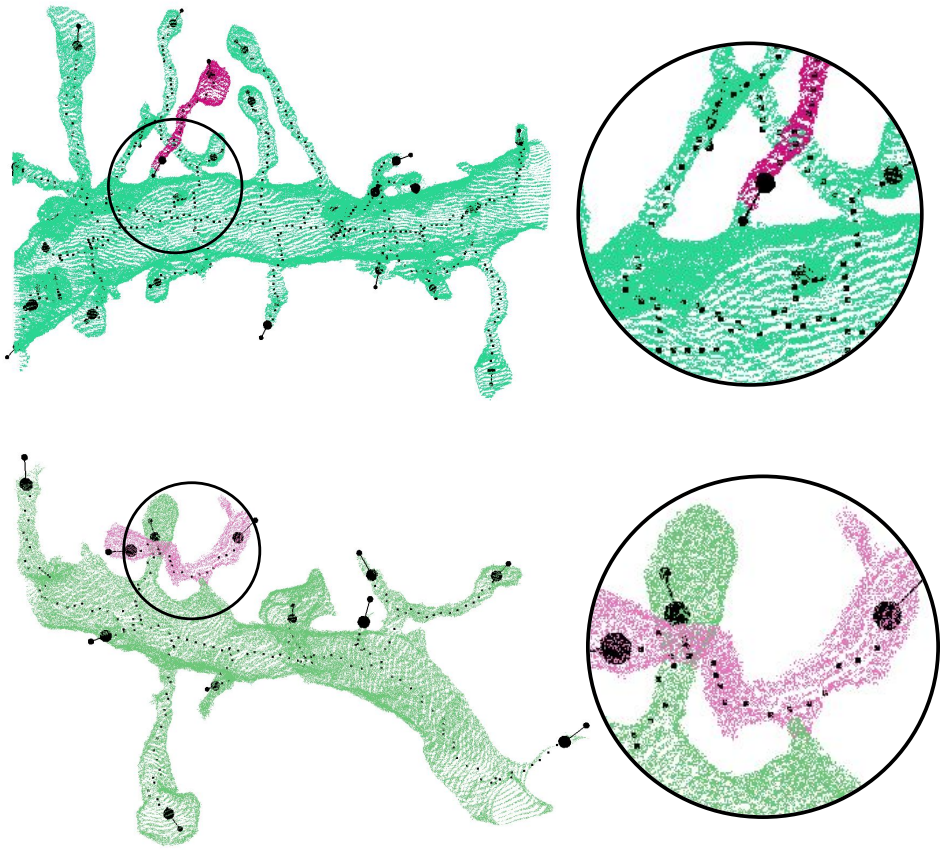


Figure 3.9: One success (top) and one failure (bottom) of our proposed biologically-constrained edge generation strategy. In the top instance, the broken spine has a skeleton endpoint with a vector directed at the main process. In the bottom example, two spines are split from the dendrite but merged in the input segmentation. The skeleton traverses near the broken location without producing an endpoint.

tations and labeled the endpoints for 500 ground truth segments. The utilized skeletonization approach has a precision of 94.7% and a recall of 86.7% for an overall F-score of 90.5% on the benchmark dataset.

Figure 3.9 shows some qualitative examples of where our method succeeds (top) and fails (bottom). Our method correctly establishes edges whenever one of the neuronal processes has a skeleton endpoint and directional vector in the vicinity of the error (top). In this particular example, the broken spine has an endpoint vector pointing directly at the corresponding dendrite. On the

Table 3.4: Our edge generation strategy reduces the number of edges in the graph by around 60% on each of the three datasets. Impressively 80% of the actual split errors remain after the edge pruning operations.

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

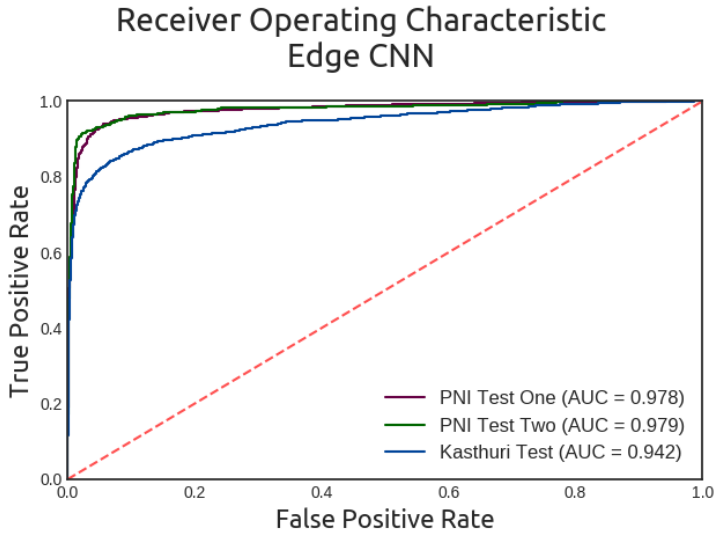


Figure 3.10: The receiver operating characteristic (ROC) curve for our learned edge features for three test datasets.

bottom, we see a failure where two spines are connected, causing the skeleton to have no endpoints at the break.

Table 3.4 provides the quantitative results for our edge generation method. The simple baseline strategy is to use the adjacency graph from the segmentation. That is, two nodes receive an edge if the corresponding segments have a pair of neighboring voxels. We notice that the adjacency graph creates many edges between neuronal processes that should not merge. In contrast, our proposed method reduces the graph size by around 60% on each of the three datasets. Similarly, our recall of true split errors is around 80% on each dataset.

We provide the results of our edge CNN in Figure 3.10. Overall our network performs well

Table 3.5: Using a global graph optimization strategy prevents segments from merging incorrectly over a traditional greedy approach. Our average decrease in VI merge over the baseline is 15.1%, with a maximum decrease of 23.6%.

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

on each dataset with accuracies of 96.4%, 97.2%, and 93.4% on the PNI and Kasthuri datasets, respectively.

Graph Partitioning. Lastly, we quantify the benefits of using a global graph partitioning strategy over a standard agglomeration technique. As a baseline, we merge regions using only the local context from our CNN classifier. To create a fair comparison with our proposed method, we merge all segments whose predicted merge scores exceed 95% (a corollary to the chosen β value). Table 3.5 shows the improvement in variation of information merge over a greedy agglomeration approach. The VI merge score decreases by 15.1% on average when using a global optimization strategy.

3.5.3 Computational Performance

All performance experiments ran on an Intel Core i7-6800K CPU 3.40 GHz with a Titan X Pascal GPU. All code is written in Python and is freely available.[§] We use the Keras deep learning library for our neural networks with Theano backend and cuDNN 7 acceleration for CUDA 8.0.

Table 3.6 shows the running time for each step of our proposed method on the PNI Test Two dataset ($2048 \times 2048 \times 256$). Our method achieves a throughput of 1.66 megavoxels per second.

[§]http://rhoana.org/biologicalgraphs

Table 3.6: Running times on a gigavoxel dataset. In total, our algorithm requires less than 11 minutes end-to-end, achieving a throughput of 1.66 megavoxels per second.

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

3.6 Conclusions

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

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

4

Subgraph Enumeration

After reconstruction of the image volumes into individual neurons with synaptic connections, we extract the wiring diagram. We typically represent the wiring diagrams as a graph with nodes corresponding to neurons and weighted edges indicating synaptic strength between neurons. From these graphs, we hope to find motifs—small, frequently occurring subgraphs of biological importance. Although we suspect or have confirmed the importance of certain subgraphs, we want to identify essential motifs not yet known to be biologically significant. Thus, we need to enumerate

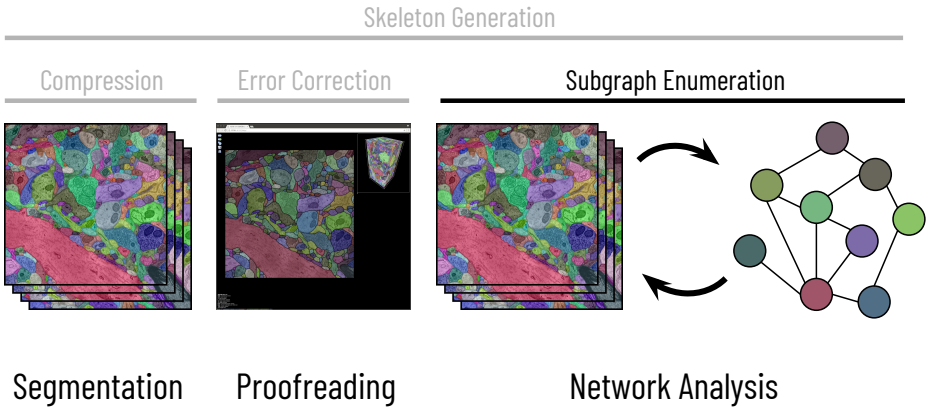


Figure 4.1: After reconstructing the image volume and identifying all synapses, we extract the wiring diagram as a graph. Under this paradigm, each vertex corresponds to a neuron with edges indicating synaptic connections between neurons. We augment the graphs to allow for edge labels corresponding to synapse strength or type (excitatory/inhibitory or chemical/electrical). Once we construct a graph from the input volume, we identify frequently occurring subgraphs of biological importance. In this chapter, we discuss the process of enumerating subgraphs in a connectome.

subgraphs in the connectome. However, Subgraph enumeration is an incredibly computationally intensive task. Furthermore, the connectome has more sophisticated properties than most general graphs. For example, edge labels can indicate either excitatory or inhibitory connections, or electrical or chemical synapses. We propose a subgraph enumeration strategy for connectomics that considers the various biological properties implicit in the connectome. By augmenting the graph with biologically relevant details, we can more accurately identify motifs and their functionality. Furthermore, we propose a divide-and-conquer strategy, which in the future will enable intraspecimen longitudinal studies that compare motifs in different brain regions. We demonstrate our results on eleven connectomes and publish extensive summaries of the 26 trillion enumerated subgraphs.

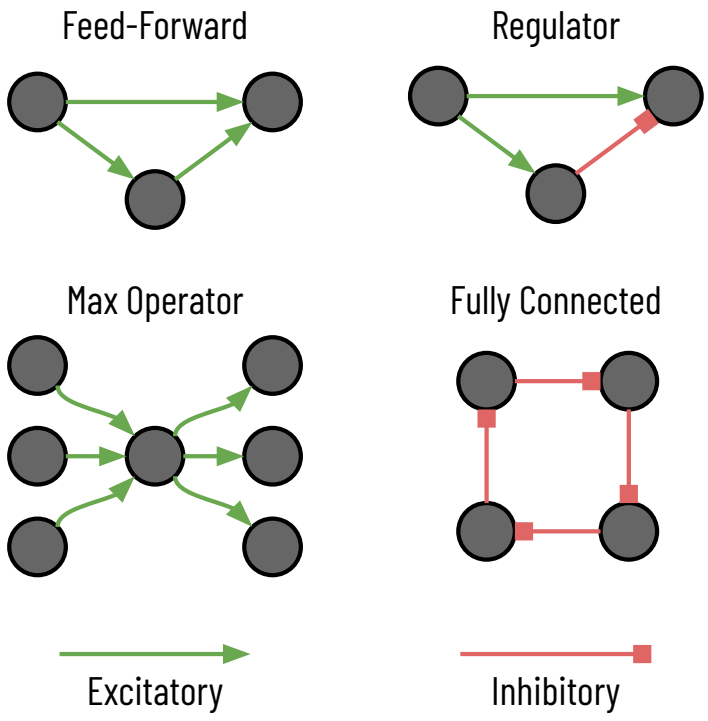


Figure 4.2: These four subgraphs that appear in the connectome have specific biological functions. Although the biological significance of these motifs was previously hypothesized, enumerating all subgraphs can identify additional essential motifs.

4.1 Motivation

After more than a dozen years of tedious image acquisition and manual reconstruction, *Caenorhabditis elegans* (*C. elegans*) became the first species to have a nearly complete mapping of its neuronal wiring diagram in 1986 [136]. This first connectome contained 302 neurons and approximately 5,000 chemical synapses. For over twenty more years [129], and continuing still [18, 137], a significant amount of research has dissected the connectome of *C. elegans*, improving its accuracy and gleaning additional insights. Building on these successes, recent rapid advancements in image acquisition techniques [25, 140, 142] paired with automatic neurite segmentation [49, 67]

and synapse prediction [45, 72] methods has enabled the extraction of more complex partial connectomes from more sophisticated species with nearly two and three orders of magnitude more neurons and synapses, respectively [141]. As these automated processes further improve, requiring less human correction and verification, we can expect more diverse animal connectomes at an even larger scale [1, 62, 123, 133].

Two significant goals for analyzing connectome structures are to generate more faithful models of the brain and improve artificial intelligence [39, 43, 70]. To this end, researchers often represent these connectomes as graphs where vertices represent neurons, and neurons that share a synaptic connection receive a directed weighted edge between the corresponding vertices [141]. Furthermore, the graph's edges can have labels, which we will refer to as colors. Edge colors indicate a specific connection type, e.g., excitatory or inhibitory. After graph construction, one of the primary goals is to identify motifs, small reoccurring subgraphs that correspond to specific biological function, hidden within the graph (Figure 4.2). A significant amount of existing literature on motif discovery for these connectomes has focused on finding specific motifs of known biological importance within the wiring diagram [116, 126, 141]. However, an open goal for exploring the connectome is to identify motifs whose biological significance was previously unknown [126].

One approach to finding these motifs is to enumerate all subgraphs in the wiring diagram to identify those frequently occurring subgraphs. Subgraph enumeration is a computationally expensive task for two reasons. First, the number of subgraphs rapidly expands as the subgraph size k increases and the possible set of connected vertices expands. Second, once enumerating a given subgraph, we need to determine its canonical labeling since two subgraphs can have different adjacency matrices and yet belong to the same equivalence class. For example, Figure 4.3 shows six subgraphs that have different adjacency matrices and yet belong to the same equivalence class; in the figure, the six subgraphs have an edge-preserving bijection that transforms the vertices from one instance into the other. The node colors indicate the bijections between the six graphs.

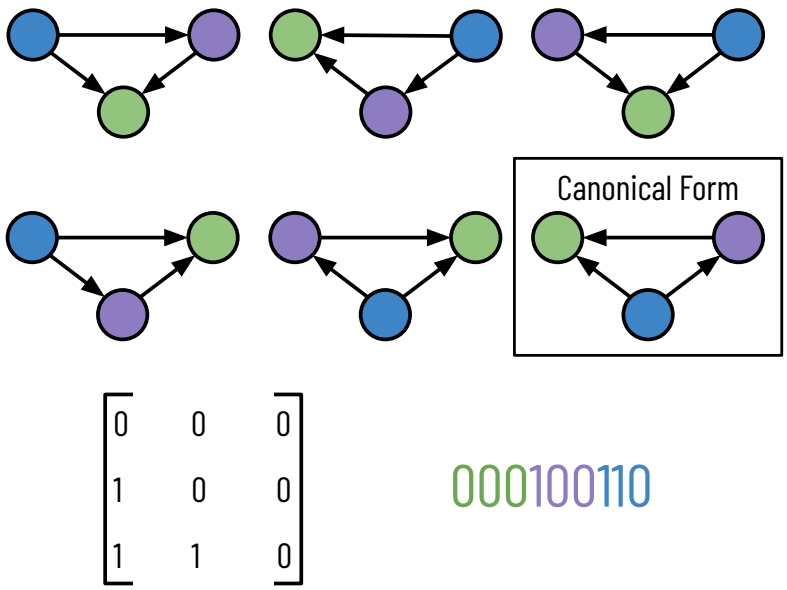


Figure 4.3: Here, we see six isomorphic subgraphs of size 3. The node colors do not represent labels but rather correspond to the bijection that preserves edges. The boxed subgraph is the canonical form where the string (bottom right) corresponding to the adjacency matrix (bottom left) is minimized.

Therefore, during subgraph enumeration, we determine the canonical labeling of every subgraph to guarantee that each subgraph uniquely maps to its equivalence class to ensure proper counting. The canonical labeling of a graph is the adjacency matrix of all subgraphs in the equivalence class that has the smallest value when written as a string (Figure 4.3, bottom right). There are no known polynomial-time algorithms to retrieve the canonical labeling from a subgraph [78]. Unfortunately, it is not feasible to count all unique adjacency matrices and determine equivalence classes as a post-processing step. There are 2^{k^2} total possible adjacency matrices for a directed graph (noting that self-loops are possible) for a subgraph of size k. Compounding these two issues, we must get the canonical labeling for each enumerated subgraph. We propose a parallel subgraph enumeration technique that builds on the existing `Kavosh` algorithm [54] (Figure 4.4, left). Our method divides the input graph using simple features about the vertices into a set of batch jobs

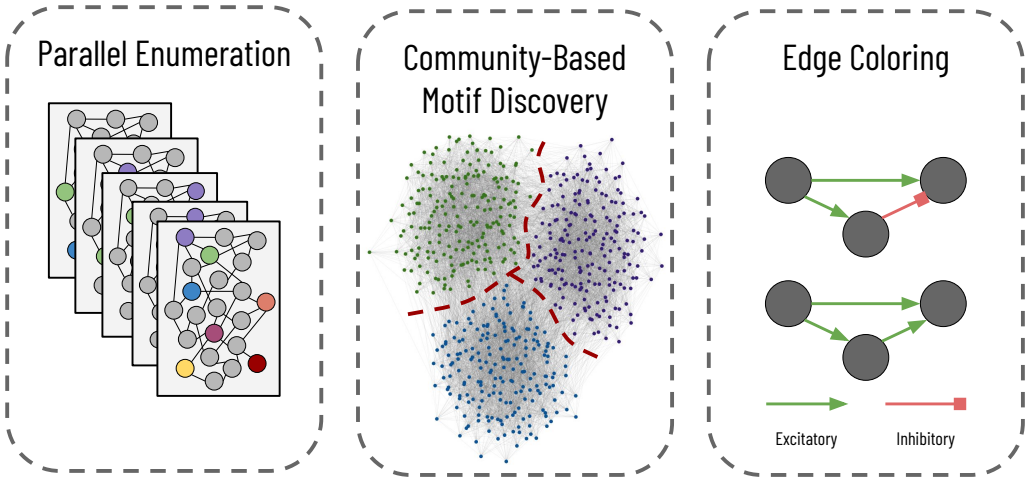


Figure 4.4: We propose three improvements on existing subgraph enumeration strategies to improve throughput and better capture the underlying biology. First, we parallelize an existing subgraph enumeration strategy to work on a distributed cluster. For each job, we enumerate a subset of vertices, indicated here by color. Second, we first cluster the vertices into different communities for larger connectomes and perform subgraph enumeration within each community. Lastly, we add edge colors to the graph to match the diversity of the synaptic connections.

with relatively equal computation times. We reduce the "wall time" over sequential computation needed for one representative dataset by 10×, as we explain further in Section (4.3.3).

Subgraph enumeration becomes infeasible even with parallelization for dense graphs with high average degrees. The size of the available extracted wiring diagrams has dramatically increased in the last decade, and future connectomes will only further increase the number of neurons and synapses [123]. For enumerating larger subgraph sizes, we first cluster the graph into communities and then perform subgraph enumeration within each community (Figure 4.4, center). This two-step divide-and-conquer approach allows us to increase the size of explored subgraphs by significantly reducing the search space. We use an automatic graph clustering algorithm that produces communities of approximately equal size. However, future users could designate communities using existing biological knowledge about the different regions of the brain. Although this approach will miss subgraphs spanning two or more communities, we will still find potentially

significant motifs within clusters and later can differentiate motifs between brain regions.

Some existing subgraph enumeration strategies allow one to distinguish vertices by "color" or "label". Some analyses on the connectome have used these algorithms to distinguish between different types of neurons [2, 103, 116]. However, few enumeration strategies immediately allow for edge colors without some manipulation. In the wiring diagram, however, edges can correspond to different types of connections with opposite functionalities (e.g., excitatory synapses stimulate and inhibitory synapses suppress) (Figure 4.4, right). Identical motif topologies with different connections can produce wildly different neural behavior (Figure 4.2, feed-forward versus regulator). Thus, we need to differentiate between edge types during subgraph enumeration to better match the input graph's actual biological realities. Although our algorithm allows for the differentiation of either vertex or edge types, we do not discuss vertex coloring since it is already prevalent in the existing literature.

We present four novel contributions for large-scale subgraph enumeration for connectomic graphs. First, we parallelize an existing subgraph enumeration strategy to significantly reduce the wall time needed to enumerate all subgraphs by balancing the computation time more fairly on a distributed computing cluster. Second, we implement a two-step subgraph enumeration strategy that first clusters the input graph into communities and then enumerates subgraphs within each community. We automatically determine these clusters in this implementation; however, this method easily extends to other clustering techniques, including manual labelings based on the underlying biology (e.g., by brain region). Third, we extend the subgraph enumeration task to include edge colors to represent the biologically diverse set of possible connections better. We demonstrate our results on three datasets containing eleven connectomes from two different animal species: *C. elegans* and *Drosophilia*. As our fourth contribution, we provide extensive analysis of the subgraphs found across both species; to facilitate further analyses, we make all data and code publicly available, including the summaries of all unique subgraphs found and their correspond-

ing counts, totaling over 26 trillion subgraphs requiring 9.25 years of computation time.

4.2 Related Work

Subgraph enumeration tasks typically fall into two categories: subgraph-centric, also referred to as motif-centric, or network-centric [107]. Subgraph-centric algorithms take as input a query subgraph and identify all occurrences of that particular subgraph [21, 37, 63]. These algorithms can efficiently find all occurrences of a given subgraph and use symmetry-breaking conditions to reduce complexity significantly. However, these algorithms require query subgraphs and do not efficiently enumerate over all subgraphs, which may be more beneficial to determine the biological importance of all subgraphs. Network-centric algorithms exhaustively enumerate all subgraphs in a given network for a given subgraph size [54, 84, 135]. These methods enumerate each subgraph once and determine its canonical labeling using `Nauty` or a similar tool [78]. In this way, these algorithms incrementally determine the number of occurrences for each unique subgraph. Analytic methods count the total number of occurrences of each subgraph type without enumerating all subgraphs by constructing a matrix of linear equations that encode connectivity information [74, 91]. However, computational restrictions limit these methods to $k \leq 6$ [107]. G-Tries represent a middle approach to network- and motif-centric methods [105, 106]. This suite of methods constructs a tree of subgraphs, where the tree leaves represent the complete set of subgraphs to enumerate.

As noted above, complete subgraph enumeration is a computationally expensive task as the number of subgraphs grows quickly as k increases. Furthermore, there are no polynomial-time algorithms for producing the canonical labeling of a subgraph [78]. Approximate counting algorithms reduce the search space by traversing to a new vertex with a fixed probability [100] or searching a compressed network [131], among other solutions. However, these strategies can miss

subgraphs, which, although very rare, would seldom appear in a random network, and therefore could indicate a biologically important motif (e.g., a fully connected subgraph of a large size). There is a significant amount of research into distributing computation for exact subgraph enumeration. These methods typically look for asymmetrical search spaces induced at each vertex [132] or edge [119].

Early research into motifs in neural wiring diagrams typically focused only on small motifs of two or three neurons [121]. Furthermore, these early discoveries typically concerned a few neuron types in specific regions of the brain. Research into motifs in the wiring diagram of *C. elegans* has focused primarily on small motifs between four or fewer neurons [103, 129]. Varshney *et al.* found similar motifs in the *C. elegans* worm as to those found in the mammalian neocortex [121, 129]. Cook *et al.* compare the relative frequency of all possible subgraphs of sizes two and three between the two specific sexes of the *C. elegans* worm [18]. Scheffer *et al.* consider both small and large motifs of the fruit fly *Drosophila melanogaster* [116]. The authors confirm some of the traditional results from previous works for small motifs, such as over-representation of reciprocal connections [44, 116]. Others have also observed these frequently occurring motifs in the connectomes of mammals and worms. Scheffer *et al.* also search for families of large motifs, such as large cliques where nearly all possible edge connections are present. However, in both the small and large cases, the authors focus primarily on motifs theorized previously theorized to be critical instead of exploring all possible subgraphs to find new motifs.

4.3 Subgraph Enumeration on the Connectome

4.3.1 Connectomes as Graphs

We construct a graph $G(V, E)$ from each connectome. Each neuron or cell represented in the connectome corresponds to a single vertex in the graph. Each vertex receives a unique index, although

these indices do not contain biological significance. Edges in the graph indicate a synaptic connection between two cells. The edges can receive labels (or colors) that can correspond to synaptic strength (moderate or strong) or connection type (excitatory/inhibitory or chemical/electrical). We use k to refer to the size (i.e., the number of vertices) of a subgraph throughout the remaining sections.

4.3.2 Kavosh Subgraph Enumeration

We extend the Kavosh algorithm to enumerate all subgraphs in our connectomes [54]. This algorithm is extremely fast and easily parallelizable (Section 4.3.3). The Kavosh algorithm begins with the vertex v with the smallest index and enumerates all subgraphs of size k for which v belongs. The algorithms then updates v to be the vertex of the next smallest index, enumerates all subgraphs similarly, and so on. At any point, the algorithm ignores any neighbors of v that have a smaller index than v to avoid counting the same subgraph multiple times since each subgraph has a unique lowest-indexed vertex. This pruning does not miss subgraphs; every subgraph has a lowest value vertex v_0, and the subgraph will be found only during the enumeration rooted at v_0. The number of enumerated subgraphs grow quickly (Table 4.6) as k increases. Therefore, we cannot store each subgraph as a tuple of vertices as the disk storage becomes too onerous quickly.

4.3.3 Kavosh Parallelization

Since the Kavosh algorithm enumerates all subgraphs rooted at a given vertex, we can divide subgraph enumeration tasks by vertex. We do not need to worry that this division will overcount individual subgraphs since a subgraph is only enumerated if the root vertex has the smallest index. Therefore, we can spawn off as many enumeration threads as vertices and guarantee complete enumeration with no subgraph duplication. However, given the large number of vertices,

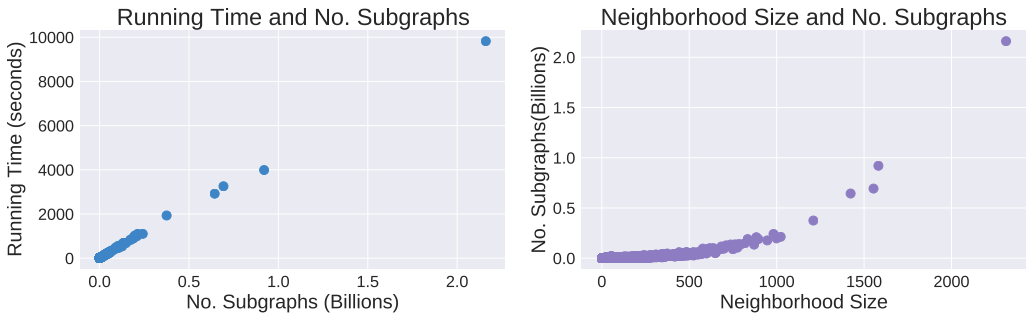


Figure 4.5: The running time for enumerating from a single vertex is linear with the number of subgraphs rooted at that vertex. We can estimate the number of subgraphs from a single vertex by looking at its immediate neighborhood. We use a heuristic to reduce the maximum neighborhood size around a given vertex to reduce the difference between the maximum and minimum running times per vertex.

it is more practical to group together vertices into batch jobs.

The running time to enumerate all subgraphs from a given vertex varies greatly (Figure 4.5, left). In one large-scale connectome with over 20,000 neurons, enumeration of subgraphs of size 4 for some vertices took three or more hours. However, enumeration concluded for 95% of neurons in under 30 seconds. Although such disparities do not significantly influence the total CPU time required, they can drastically alter the "wall time" required when running on a distributed cluster. In most large-scale wiring diagrams, the neuron indices are typically random—an artifact of the automatic reconstruction methods that gradually agglomerate an oversegmentation [49, 67]. Even with species with highly stereotyped connectomes, like *C. elegans*, we can merely assign a new "enumeration index" to each vertex that substitutes the given neuron index. Therefore, we selectively choose the "enumeration indices" for each vertex to reduce computation time variance.

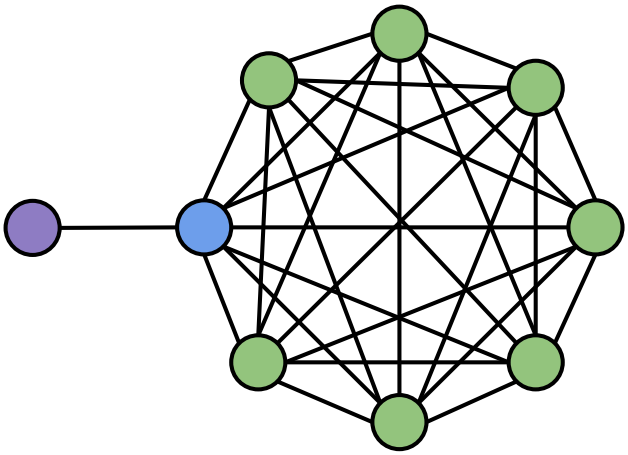
For each vertex v, the number of neighbors of v with a larger *enumeration index* strongly indicates the number of subgraphs rooted at that vertex, and consequently, the running time required to enumerate from v (Figure 4.5). Intuitively, we know that if there are n neighbors of v with a higher index, at a minimum, there are $\binom{n}{k-1}$ subgraphs contained within the immediate neighborhood alone. We only consider the neighborhood of vertices with larger enumeration indices

Algorithm 1 CalculateEnumerationIndex

```
Input G(V, E) ← graph with V vertices and E edges
for v in V do
    v.degree ← len(v.neighbors)
end for
order ← [ ]                                   ▷ Final enumeration order
while len(order) ≠ ||V|| do
    v ← arg min_{v∈V} v.degree
    order.append(v)
    for u ∈ v.neighbors do
        u.degree−−
    end for
end while
```

because Kavosh does not enumerate subgraphs rooted at v that include vertices with lower-valued indices. When we use the pre-existing neuron indices as the enumeration index, a few unlucky neurons have a low index and an exceptionally high degree. These corresponding vertices dominate the computation time. Therefore, we greedily generate the vertex indices using Algorithm 1. In summary, we assign an enumeration index of 0 to the vertex v_0 that has the smallest edge degree. We then decrement v_0's neighbors' degrees; enumeration from those vertices will not consider subgraphs that include v_0. Next, we choose the vertex, v_1, with the lowest remaining edge degree, and similarly update the remaining vertices' neighborhoods. We continue this process until each vertex has an enumeration index.

One might expect that we could gather more information about the possible running times by considering second-order neighborhoods that include neighbors of neighbors. However, we found little correlation between the number of subgraphs to enumerate and second-order neighborhoods. Intuitively, vertices on the periphery of a large clique will have a small first-order neighborhood and a substantial second-order neighborhood (Figure 4.6). However, the number of subgraphs rooted at the peripheral vertex will be significantly less than its neighbor since every



Vertex		
Immediate Neighborhood Size	1	8
Extended Neighborhood Size	8	8
Size 3 Subgraphs	8	29
Size 4 Subgraphs	21	56

Figure 4.6: Although immediate neighborhood size is a good indicator of the number of enumerated subgraphs, there is little correlation between the extended neighborhood size and the number of subgraphs.

subgraph from the peripheral vertex must also include its neighbor.

4.3.4 Community-Based Enumeration

Even with smarter parallelization strategies, enumerating all subgraphs for even $k = 5$ becomes quickly infeasible for large-scale connectome graphs. For a representative connectome with approximately 22,000 neurons and 820,000 edges, subgraph enumeration started at 9.44 minutes, and 2.15 days for motifs of size three and four, respectively, but 2.77 years for motifs of size five (Table 4.6). Considering that identifying motifs often requires subgraph enumeration on ran-

domized graphs of a similar size, this computational cost becomes overbearing even across a large distributed compute cluster. Therefore, we allow users to cluster the vertices in the graph into communities and perform subgraph enumeration within each community. A downside of using clustering is that we miss enumerating subgraphs that span more than one community. However, we can still identify the frequently occurring motifs within clusters and contrast subgraph counts between different clusters.

For graph clustering, we use the METIS algorithm, which divides an input graph into a predetermined number of clusters [53]. This algorithm has the desirable property of producing relatively evenly sized communities. Other such algorithms provide unequal clusters, which significantly reduces the effectiveness of the divide-and-conquer strategy. Although we use an automatic clustering technique, future analyses could segment the connectomes based on biological priors such as brain regions. With a divide-and-conquer clustering approach, future research could contrast subgraph distributions based on the brain regions themselves.

4.3.5 Edge Coloring

In the innermost loop of subgraph enumeration, we must identify each subgraph's canonical form to classify a given collection of connected vertices correctly. Although it is not known if the graph isomorphism problem is NP-Complete, there are currently no polynomial time algorithms, and the current best provable complexity is $\exp\left(\mathcal{O}(\sqrt{n \log n})\right)$ [6]. Many motif discovery algorithms use the `nauty` library for graph isomorphism. This highly optimized library returns the canonical labeling of a vertex-colored graph [79]. However, `nauty` does not currently support edge colors without some manipulation of the input, requiring an expansion of the graph size to $\mathcal{O}(n \log d)$, where d is the number of possible edge colors [78]. Thus, although many motif discovery algorithms allow for vertex colors, they typically do not consider different colored edges. However, different edge types are essential for brain networks and differentiate between

Table 4.1: We enumerate subgraphs on eleven connectomes from two different species. Two of the adult *C. elegans* connectomes also contain end-organs and muscles.

Species	Age	Sex	Neurons	Edges	Edge Types
Caenorhabditis elegans	Adult	Hermaphrodite	473	6,897	Chemical/Electrical/Both
Caenorhabditis elegans	Adult	Male	598	7,725	Chemical/Electrical/Both
Caenorhabditis elegans	0 h	Hermaphrodite	225	775	N/A
Caenorhabditis elegans	5 h	Hermaphrodite	225	986	N/A
Caenorhabditis elegans	8 h	Hermaphrodite	225	1,006	N/A
Caenorhabditis elegans	16 h	Hermaphrodite	225	1,101	N/A
Caenorhabditis elegans	23 h	Hermaphrodite	225	1,504	N/A
Caenorhabditis elegans	27 h	Hermaphrodite	225	1,524	N/A
Caenorhabditis elegans	Adult (50 h)	Hermaphrodite	225	2,193	N/A
Caenorhabditis elegans	Adult (50 h)	Hermaphrodite	225	2,189	N/A
Drosophila melanogaster	Adult	Female	21,739	841,720	Moderate/Strong

excitatory and inhibitory connections and chemical and electrical (gap junction) synapses. Additionally, some pairs of neurons will have multiple pathways between them (e.g., chemical synapses and gap junctions). We can assign a new edge color to indicate neurons that have multiple types of connections. Although the `Kavosh` algorithm does not natively support edge colors, the `nauty` documentation briefly discusses how to add edge colors [78]. We modify the enumerated subgraphs before generating the canonical labeling based on the `nauty` documentation.

4.4 Experimental Set Up

4.4.1 Datasets

We perform motif discovery on three classes of datasets containing eleven connectomes, ten from *C. elegans*, and one from *Drosophila melanogaster*. Table 4.1 summarizes each of the eleven connectomes, which are all publicly available.

Our first two datasets contain one complete connectome from each of the two biological sexes of *C. elegans*: hermaphrodite and male. The two sexes have 302 and 385 neurons for the hermaphrodite and male, respectively. These two datasets also include muscles, end-organs, and glial cells

leading to 473 and 598 nodes in the graph for the two sexes.* Cook *et al.* produced the first male connectome and provided an analysis of the differences between the two sexes [18]. In their analyses, they focus on subgraphs of sizes two and three only. We extend these analyses by exploring larger subgraphs than initially considered. We also enumerate all subgraphs with three possible edge colors: chemical synaptic connections, gap junction (electrical), or both. We refer to the male and hermaphrodite connectomes from this dataset as *C. elegans AM* and *C. elegans AH* for *adult male* and *adult hermaphrodite*, respectively.

Our next eight datasets come from a longitudinal study of developmental growth of *C. elegans* published by Witvliet *et al.* [137]. These eight partial connectomes contain 225 neurons, each with increasing numbers of synapses based on the specimen's age. Six of these connectomes come from adolescent worms, and the oldest two are adults. These datasets only contain chemical synapse information, so there are no unique edge types. Likewise, we refer to these datasets as *C. elegans D[1-8]* with the number indicating the specimen's relative age.

The *Drosophila* dataset is the largest publicly available connectome with over 21,000 neurons and four million synaptic connections [141]. We define three synaptic connectivity levels between two neurons: weakly, moderately, and strongly connected. Weakly connected neurons share three or fewer synapses, moderately connected neurons share more than three but fewer than ten synapses, and strongly connected neurons share ten or more synapses. Based on the authors' discussion on the precision of synapse detection [141], we prune our graph to contain only synaptic connections that are moderately or strongly connected. Removing these edges reduces the number of edges in our graph from 3,550,404 to 841,720. Compared to mammalian brain samples, the EM imagery does not provide enough detail to differentiate excitatory and inhibitory

*The number of nodes and edges differ from the original publication of these datasets [18]. Our connectomes come directly from the adjacency matrices in Supplementary Information 5, corrected version July 2020. These datasets contain extrapolated information from other *C. elegans* specimens for cells missing from the reconstruction of Cook *et al.*

connections [141]. Therefore, our edges do not reflect that distinction.

4.4.2 Implementation Details

We implement our subgraph enumeration algorithm in `C++` and provide a `Python` wrapper. We use the `nauty` [78] library to generate canonical labelings for each enumerated subgraph. We ran timing analysis on a distributed cluster with Intel E5-2695 v2 processors at 2.40 GHz 12 core, with 90 gigabytes of RAM. All code and enumerated subgraph results are publicly available.[†]

4.5 Results

This section provides summaries and insights on the subgraphs in these connectomes and quantitative statistics on the number of subgraphs and the computation time. To encourage further analyses by others in the field, we provide complete summaries of all subgraph counts for all connectomes. These summaries cover over 26 trillion subgraphs over the eleven connectomes and required over 9.25 years of computation time.

4.5.1 Motifs

Figure 4.7 shows the relative proportion of size three motifs for both the developmental growth connectomes and the two sexes of *C. elegans*. Amazingly, the distribution of subgraphs between the specimens in each grouping of datasets is remarkably similar for both subgraphs of size three. These similarities cannot be explained entirely by the multiple connectomes sharing an abundance of joint edges. For *C. elegans AH* and *AM*, the two connectomes share 449 neurons, muscles, end-organs, and glial cells, leaving 24 and 149 unique cells for the hermaphrodite and male, respectively. The two specimens have 3,572 equivalent edges among the shared nodes, leaving

[†] github.com/rhoana/subgraph_enumeration

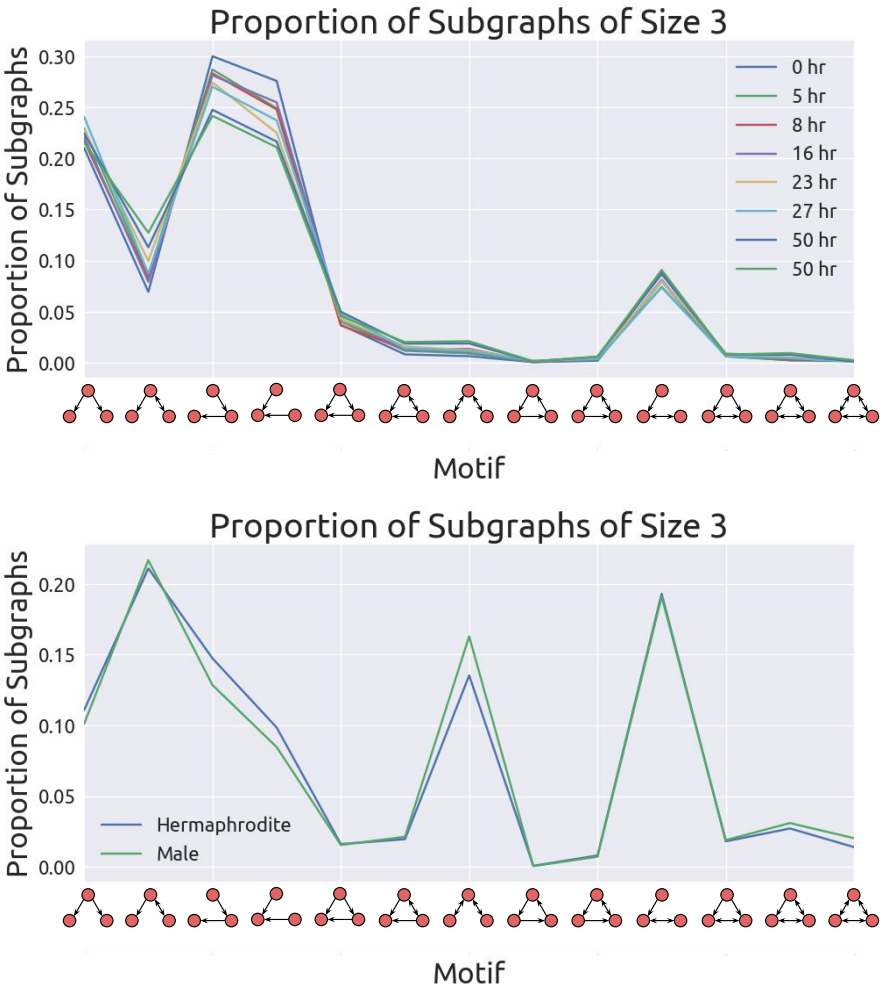


Figure 4.7: The distributions of subgraphs of size three are amazingly similar in the *C. elegans* developmental and the *C. elegans* sexes datasets. Note that these datasets are not comparable to one another since the datasets differentiating the two sexes include muscles, non-muscle end organs, and glial cells.

3,325 and 4,153 unique edges for each sex. We cannot compare the developmental series with the two other adult worms for two reasons. First, the developmental growth series are only partially connectomes with only 225 neurons each. Second, *C. elegans AH* and *AM* contain muscles and end-organs in addition to all of the neurons in the brain. We provide a more thorough summary of the five most frequent subgraphs of sizes four and five in Figure 4.8. We note that the same

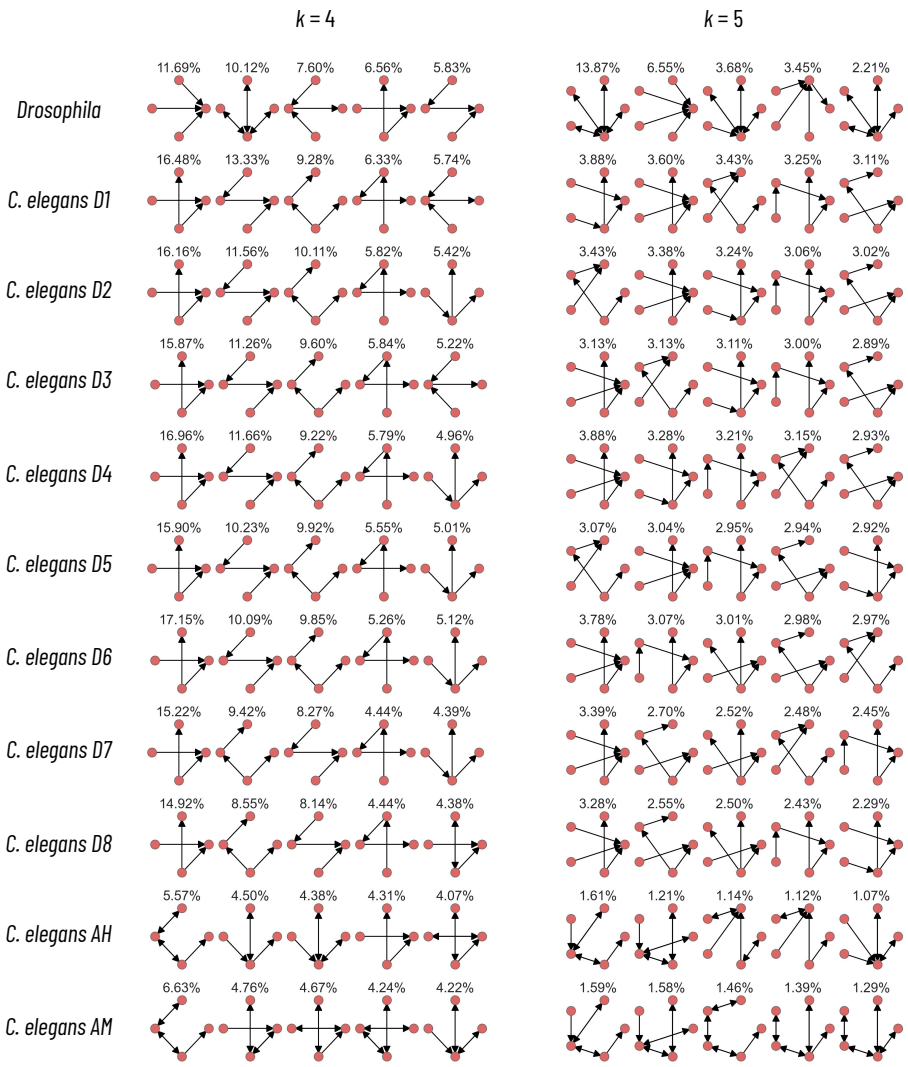


Figure 4.8: Here, we summarize the five most common motifs of sizes four and five for the eleven connectomes. We notice a large number of similarities between connectomes in the two longitudinal studies for *C. elegans*.

motifs frequently appear between specimens in the two longitudinal studies.

Table 4.2: We can significantly reduce the wall time required when enumerating subgraphs over a compute cluster by relabeling the vertex indices. We reduced the maximum time for enumerating from a root vertex from just under three hours to slightly more than one minute on the *Drosophila* dataset. The new enumeration order reduces the idle CPU time by over 665 hours.

	Naïve	Algorithm 1
Mean Time	8.21 sec	8.54 sec
Median Time	0.92 sec	5.80 sec
Maximum Time	9820.40 sec	74.12 sec
Wall Time	175.20 min	15.78 min
Idle CPU Time	680.44 h	14.18 h

4.5.2 Ablation Studies

Kavosh Parallelization. Subgraph enumeration for even moderate k on the larger *Drosophila* dataset is simply infeasible without parallel processing. We enumerate all subgraphs of size five on this dataset in under 7 days when running on 145 CPUs. Without parallelization, that enumeration would take 2.77 years (Table 4.6). Even on the much smaller *C. elegans* datasets, enumerating subgraphs of size 7 on *C. elegans AH* takes 17.95 days of CPU time.

Although the amount of total CPU time is similar using a naïve enumeration ordering as to the ordering described in Algorithm 1, the maximum time spent for a single vertex decreases significantly on the *Drosophila* dataset from 9820.40 sec to 74.12 sec (13.23×). When running on 250 CPUs, our algorithm reduces the wall time by 11.10× and the idle CPU time by 47.99×.

Community-Based Enumeration. A divide-and-conquer approach to subgraph enumeration can significantly reduce the total computation time required on a given dataset. The number of subgraphs found decreases, particularly for larger values of k. These decreases correspond to subgraphs that span more than one community. In going from one to five communities, the number of subgraphs decreases by 55%, with a similar reduction in total computation time. Although this is a significant decrease in enumerated subgraphs, the relative proportions of the mo-

Table 4.3: A divide-and-conquer approach can significantly decrease the total amount of computation time required when enumerating very large connectomes. The number of enumerated subgraphs quickly drops as the number of communities increases.

No. Communities	No. Subgraphs	Total Time	Cosine Similarity
1	36,041,949,778	51.54 h	1.0000
5	16,111,511,700	23.49 h	0.9577
10	10,660,308,898	15.66 h	0.9207
15	8,103,662,469	11.37 h	0.8693
20	6,340,790,895	8.85 h	0.8244
25	4,563,678,610	6.41 h	0.7946
30	3,819,932,809	5.39 h	0.7915

tifs found remain relatively stable, as described by the cosine similarity metric, which compares the distances between the two distributions of motif counts. Despite this reduction in the number of subgraphs, we can still identify significant motifs per community. We believe this trade-off between exhaustive enumeration and quicker processing is a necessity moving forward. Although the *Drosophila* dataset is the largest connectome to date, another most likely will surpass it within the next half-decade. A divide-and-conquer approach can avoid enumerating subgraphs that span multiple communities and focus only on those tightly connected regions of the brain. Furthermore, we believe this methodology extends well to connectomes with predefined communities such as different brain regions. In these instances, one could perform intraspecimen comparisons between the motifs found in different communities.

Edge Coloring. Adding edge colors increases the amount of computation time required for subgraph enumeration. However, differentiating edges by color is critical for these biological graphs where two edges can have markedly different properties. Table 4.4 gives a brief overview of the increase in running time when adding edge colors to the connectome. Typically, enumerating subgraphs with edge colors increases execution time by 2–2.5×.

Table 4.4: Adding edge colors dramatically increases the running time for subgraph enumeration.

Dataset	No Color/Edge Color				
	3	4	5	6	7
Drosophila	566.63/1053.48 s	2.15/4.39 d	2.77/6.12 yr	N/A	N/A
C. elegans AH	0.37/0.77 s	14.75/31.78 s	727.96/1607.75 s	9.53/21.76 hr	17.95/42.48 d
C. elegans AM	0.38/0.72 s	13.18/29.61 s	593.57/1296.49 s	7.13/16.38 hr	12.54/30.11 d

Table 4.5: We publish extensive summaries of the subgraphs found for the eleven connectomes we analyzed. For three of the connectomes (*), we enumerate the graph both with and without edge colors.

Dataset	3	4	5	6	7
*Drosophila**	126,610,248	36,041,949,778	12,522,283,314,604	N/A	N/A
C. elegans D1	6,327	71,720	896,426	11,515,005	147,005,821
C. elegans D2	9,612	128,983	1,888,538	28,275,169	418,971,019
C. elegans D3	9,665	129,049	1,878,288	27,946,970	412,000,679
C. elegans D4	11,276	159,734	2,464,032	38,761,285	602,393,981
C. elegans D5	18,644	318,317	5,870,163	110,031,311	2,034,438,610
C. elegans D6	19,676	342,951	6,425,487	122,105,708	2,288,712,164
C. elegans D7	35,361	782,947	18,348,047	429,395,327	9,761,169,981
C. elegans D8	34,061	740,874	17,091,235	394,556,031	8,863,956,956
*C. elegans AH**	126,977	4,284,966	156,792,085	5,758,405,667	205,778,553,473
*C. elegans AM**	125,601	3,809,067	126,545,565	4,286,896,477	143,807,877,796

4.5.3 Enumerated Subgraph Dataset

We publish exhaustive summaries of the subgraphs found over the eleven connectomes to encourage further analyses. For *C. elegans AH* and *AM* datasets, we found all subgraphs $3 \leq k \leq 7$ with and without edge colors. For the *C. elegans D[1-8]* datasets, we found all subgraphs $3 \leq k \leq 7$ (no edge colors available). Finally, for the *Drosophila* dataset, we enumerated subgraphs of $3 \leq k \leq 5$ with and without edge colors. These datasets contain summaries of over 26 trillion subgraphs across all of the connectomes, which required over 9.25 years of total computation time. We hope that this readily available dataset will encourage additional longitudinal studies across species, sexes, and developmental stages in the future.

Table 4.6: The number of subgraphs, and the corresponding amount of time needed for enumeration, greatly increases as k increases on large connectomes.

Motif Size	No. Subgraphs	CPU Time
	Drosophila	
3	126,610,248	9.44 min
4	36,041,949,778	2.15 d
5	12,522,283,314,604	2.77 yr

4.5.4 Computational Complexity

Time. Subgraph enumeration is a computationally expensive task, as we see the number of subgraphs quickly explode in the Drosphila dataset (Table 4.6). Although enumerating subgraphs of size three takes less than ten minutes, the total CPU time required balloons to 2.77 years. Comparatively, we enumerate subgraphs for all of the *C. elegans* connectomes considerably quicker. The most time-intensive enumeration for these specimens was for subgraphs of size seven for the adult hermaphrodite (473 vertices, 6,897 edges), requiring 17.95 days computation time.

Memory. One of the most significant benefits of the Kavosh algorithm is its memory efficiency [54]. Empirically we found that our parallel implementation requires less than 800 MB of RAM for each generated process for subgraphs with six or fewer vertices. Our methods require more RAM for larger subgraphs, although all processes required less than 3 GB.

4.6 Conclusions

Twelve years of onerous work yielded the first connectome in 1986 with 302 neurons and over 5,000 synapses. Since then, advancements in image acquisition, neural reconstruction, and synapse detection have significantly reduced the amount of time needed to extract these wiring diagrams. A recent dataset of a fruit fly contains over 21, 000 neurons and 800, 000 moderate and strong synaptic connections. We expect similar growth in the future as neuroscientists recon-

struct brain tissue from even more evolved species such as bumblebees, shrews, and even humans. One goal of extracting these wiring diagrams is to create more faithful models of the brain and advance artificial intelligence. Thus, we will need to identify the motifs in these wiring diagrams that correspond to biologically essential functions.

Subgraph enumeration on these dense connectomes is a computationally expensive process that becomes infeasible even for small wiring diagrams with < 1000 nodes. We present a novel subgraph enumeration strategy for large connectomes that enables us to find frequent motifs. We parallelize our method to work across a distributed cluster and, when needed, use a two-step enumeration method that first divides the wiring diagram into communities. Building on previous enumeration strategies, we include methods for differentiating edge types to resemble the underlying biology better. We evaluate our methods on eleven connectomes from two species and provide summaries to the over 26 trillion enumerated subgraphs. Ten of these connectomes come from two existing longitudinal studies on *C. elegans* [18, 137]. We hope that our published motifs will encourage further analysis into these eleven connectomes and longitudinal studies across brain regions within a specimen and even across different species. Some motifs are known to occur in high frequency across brain regions and animal species, such as the feed-forward loop and the reciprocal connection [18, 116, 121]. We hope that publishing enumerated subgraphs in variable wiring diagrams will enable the discovery of other biologically essential motifs that span species and offer additional insights into the brain's inner workings.

As connectomes continue to increase in the number of neurons and synaptic connections, we will need to explore new ways to enumerate subgraphs efficiently. By dividing the brain into individual regions with shared functionality, we can perform intraspecimen longitudinal studies while significantly reducing the computation time for enumeration. Eventually, however, we may need to consider approximate counting methods that sample subgraphs randomly to produce estimates for the motif counts. Although these methods may miss important yet rare motifs, such

as fully connected cliques, they will reduce computational costs while still producing extensive summaries of the frequently occurring subpatterns in the brain.

5

Skeleton Generation

Skeletons have become an increasingly important component along the connectomics pipeline with applications in analysis, segmentation evaluation, visualization, and error correction. As the connectome volumes have increased in size, skeletons provide a simplified yet expressive representation for working with the data. Most of these applications use off-the-shelf skeleton generation strategies that are not explicitly designed for connectomics. We propose a biologically-aware topological thinning algorithm to produce accurate centerlines that remain faithful to the

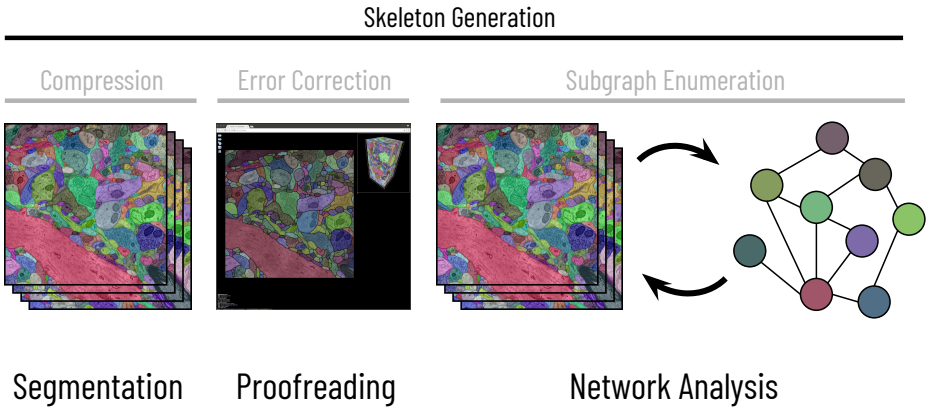


Figure 5.1: Skeletonized representations of the reconstructed label volumes have many uses across the connectomics pipeline with application in analysis, segmentation evaluation, visualization, and error correction. The most frequently used off-the-shelf approaches, such as isthmus thinning or the TEASER algorithm, are agnostic to the underlying biology. In this chapter, we discuss a biologically-aware skeleton generation technique. We guarantee synapse connectivity and anchor neurite skeletons to the cell body. Furthermore, during skeleton generation, we calculate geometric attributes such as the geodesic distance from synapse to soma and the width of the neurites. Cable theory, the mathematical model that estimates the electrical signal received from a particular synapse, relies on these two values.

underlying biology. We anchor our skeletons to the cell body and guarantee that all synapses remain connected. We simultaneously estimate neurite widths and geodesic distances between the synapses and the cell body while generating the skeletons. We need these vital geometric statistics to estimate the perceived synaptic strength between two neurons more accurately. We divide our method into a series of data-intensive parallel tasks and cheap global recombination steps to achieve throughputs of over one million voxels per second. We evaluate our methods on 1,255 neurons and neuron fragments from three datasets with hundreds of billions of voxels.

5.1 Motivation

Increased throughput of electron microscopy imaging techniques [123] has enabled nanometer resolution image volumes of brain tissue exceeding terabytes [123, 146] and even petabytes [25,

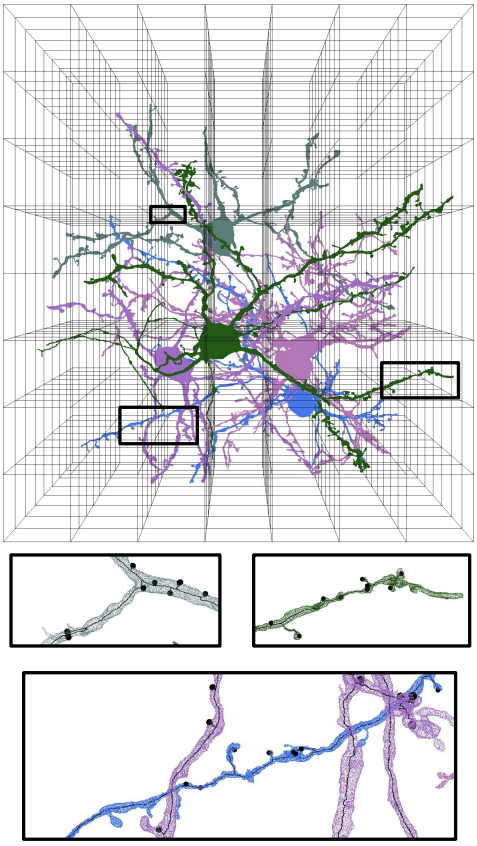


Figure 5.2: Neural reconstructions of electron microscopy image stacks easily exceed billions of voxels in size. Here, we show five different neurons that span 270 billion voxels. Our block-based synapse-aware skeleton generation strategy produces expressive skeletons efficiently. We enlarge three regions to show our generated skeletons that connect all synapses, represented by black spheres, to the somata.

142] in size. Since manual reconstruction and synapse annotation is infeasible at this scale, researchers employ automatic techniques to segment the volumes into individual neurons [49, 141] and identify synapses [24]. One of the primary goals of connectomics is to better understand the brain's computational workings by analyzing the wiring diagrams extracted from these large image volumes [70, 122]. In turn, researchers hope to improve artificial neural networks [43], expand knowledge of neurological diseases [30], and better understand the brain's underlying computational mechanisms [70].

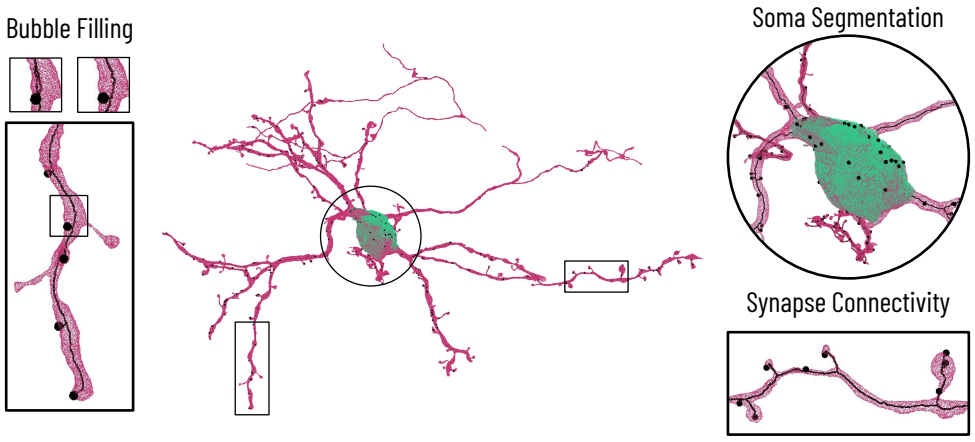


Figure 5.3: Our block-based, topological thinning strategy for generating expressive skeletons leverages domain-specific knowledge of the underlying biology. Automatic neural reconstruction techniques often produce millions of bubbles in the output segmentation. Uncorrected, these bubbles cause the skeletons to deviate from the neurite center (inset, top left, without, then with, bubble filling). Since we know that neuronal processes fill the space enclosed by the cell membrane, we can safely fill these bubbles to produce more exact centerlines. We then identify the cell body (soma) for each neuron (inset, top right). Masking out the soma for a neuron increases throughput significantly and allows us to anchor the skeletons on the cell body's surface. Lastly, we introduce a set of topological thinning rules that guarantee connectivity between all synapses (inset, bottom right). Our thinning algorithm concurrently produces neurite widths and geodesic distances from synapses to the soma—two physical properties needed for accurate neural simulation.

Generating accurate skeletons of the segmented neurons has become a critical component of the connectomic pipeline with applications in analysis [29, 141], segmentation evaluation [49], visualization [85], and error correction [23, 76]. Most of this research uses a variant of the Tree-structure Extraction Algorithm for Accurate and Robust Skeletons (TEASER) [114, 145], although some work utilizes topological thinning strategies from the volume processing community [35, 68, 92]. The TEASER algorithm has a set of tunable parameters that offer a tradeoff between expressivity and simplicity. At the same time, topological thinning strategies typically require excessive computation time and memory for large datasets. Furthermore, both methods are agnostic to the underlying biology and do not impose restrictions on the generated skeletons.

As the physical volume sizes of reconstructed brain samples have approached and even ex-

ceeded a cubic millimeter [25, 71, 117, 141], more research considers the analysis of the extracted wiring diagrams. Despite the considerable algorithmic improvements along the entire connectomic pipeline, most of this analysis still occurs at a relatively coarse level. Current approaches typically construct a graph where each node corresponds to a neuron, and directed edges indicate a synaptic connection from one neuron to another [29, 141]. Weighted edges may indicate perceived synaptic strength, although these methods typically only consider the number of synapses between two neurons when assigned edge weights. This approximation of synaptic strength is an oversimplification of the actual connectivity between two neurons. From cable theory, we know that the electrical signal transferred from one neuron to another depends on the geodesic distance between the two cell bodies (somata) through the synapse and the neurite width along that path [61]. We can generate more accurate wiring diagrams by extracting these essential geometric properties from the skeletons to quantify synaptic strength better. Furthermore, wiring diagrams derived from synapse-aware skeletons can model interplays between neurite branches.

We propose a block-based, topological thinning approach for generating exact skeletons of reconstructed neuronal volumes. Our approach enforces various biological constraints on the skeletons and generates relevant geometric information useful for higher-level analysis (Figure 5.2). Current state-of-the-art reconstruction strategies such as flood-filling networks [49] can create segmentations with millions of bubbles, i.e., pockets of voxels incorrectly labeled within a single neuron. Since neuronal processes are solid volumes, we can safely fill these bubbles to improve the segmentation quality, generate more accurate neurite widths, and significantly speed up computation by removing spurious surface voxels during thinning (Figure 5.3, inset, left). We train a Convolutional Neural Network (CNN) with the familiar U-Net architecture [15, 110] to detect somata in the volume. We then anchor our skeletons to the cell body's surface to create more accurate geodesic distances between the synapses and the soma (Figure 5.3, inset, top right). Finally, we devise a topological thinning strategy that produces a nearly one-to-one correspondence

between synapses and skeleton endpoints (Figure 5.3, inset, bottom right). We produce accurate width estimates and geodesic distances to the cell body for each synapse during the thinning process.

We increase overall skeleton accuracy and significantly decrease the skeleton generation time needed compared to our previous work [77]. The introduction of the bubble filling and soma segmentation steps decreases the overall computation time for skeleton generation by 10.58× on a connectome volume that spans over two hundred billion voxels. Furthermore, we now divide each step of the skeleton generation process into an array of intensive data-parallel tasks and quick global recombination steps. This block-based approach allows us to distribute computation over a large number of CPUs, achieving throughputs of over one million voxels per second per CPU.

5.2 Related Works

An increasing amount of connectomics literature uses skeletons for analysis [125], segmentation evaluation [49, 102], visualization [85, 141], and error correction [23, 76]. Much of this research uses TEASER [114], a general-purpose skeleton generation approach for biomedical applications that iteratively finds distant points in the object to connect to a root voxel, or similar derivatives [145]. However, TEASER relies on repeated calls to Dijkstra's algorithm to find these distant points, a super-linear algorithm that cannot efficiently handle block sizes over 1280^3 voxels. Skeletons are more widely applicable in the medical image community with applications in extracting graphs from blood vessels [14], among others [50].

In volume processing, topological thinning approaches remove the need for parameters by relying on mathematical notions of curve-endpoints [68, 113] or curve-isthmuses [92] to reduce skeleton branching. In the volume processing community, skeletonization typically refers to reducing the dimensionality by one (i.e., to two dimensions). We produce centerlines or curve skele-

tons [51], which we interchangeably refer to as centerlines or skeletons. These strategies gradually erode a binary label volume's surface while preserving the topology using only local context around a given voxel [73]. Generally, these methods either remove voxels sequentially [92, 94] by continually verifying the topological correctness before each new deletion or in parallel [68, 93, 139], where sets of independent voxels are processed separately. However, these thinning strategies are incapable of efficiently processing terabyte size volumes as they require reading the entire voxel space into RAM. We propose a novel block-based approach that allows for large label volumes' rapid skeletonization, leveraging data-parallel computation strategies.

5.3 Biologically-Aware Skeleton Generation

Our method contains three main components to enforce specific biological properties and improve the accuracy of our generated skeletons' geometric attributes. We take two inputs: a label volume where each neuron is assigned a unique 64-bit integer and a list of synapse locations for each neuron (Figure 5.4, top left box). We first fill bubbles in the input segmentation to correct common errors during automatic segmentation (Figure 5.4, top right box). We divide this process into two data-parallel tasks with a global recombination step between them. Since neuronal processes are solid volumes, we can safely identify and correct these errors without creating additional ones. Next, we downsample the segmentation and use a modified U-Net to detect the cell bodies from the input label volumes alone (Figure 5.4, bottom left box). Finally, we thin the neurons using a block-based, synapse-aware strategy that connects all synapses to the cell body (Figure 5.4, bottom right box). We also divide the topological thinning process into two data-parallel tasks followed by a global recombination step. By distributing the most computationally expensive operations over a large number of CPUs, we can quickly generate skeletons on terabyte datasets.

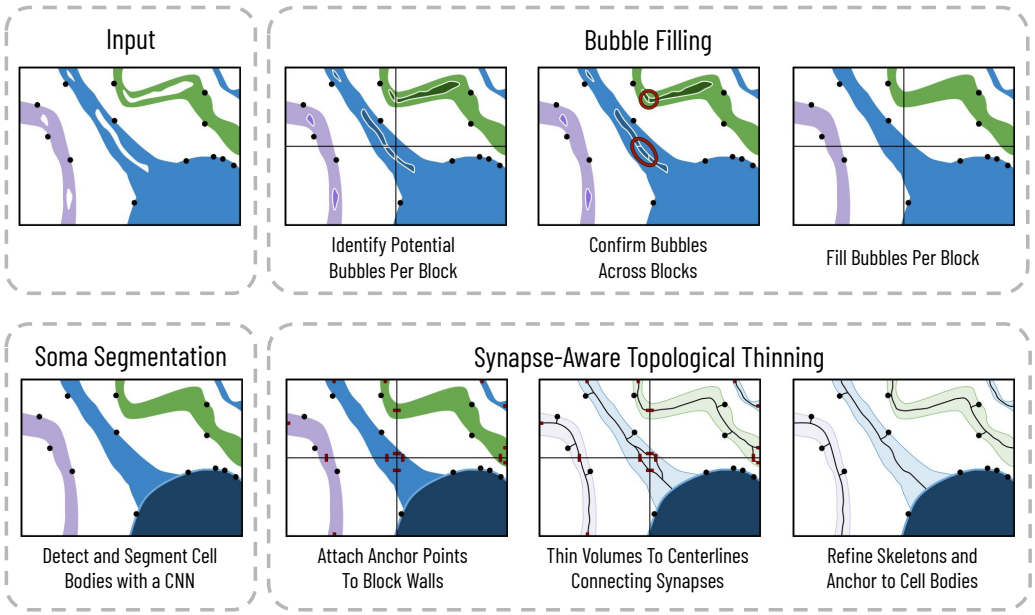


Figure 5.4: Our method contains three distinct steps to extract biologically-aware skeletons from an input label volume: bubble filling, soma segmentation, and synapse-aware topological thinning. We subdivide the bubble filling and topological thinning steps into a series of expensive data-parallel tasks and cheaper operations that require a global scope. Cross-sectional lines in an image indicate block-based operations for that step. We first identify bubbles in each block by finding the connected components with only one neighbor component. Since bubbles can span across block boundaries, we link the bubbles across the block boundaries and confirm that only one neuron encapsulates the entire bubble. After identifying the actual bubbles, we update their value to match the surrounding neuron. Next, we detect and segment cell bodies using a CNN. In the first part of topological thinning, we attach anchor points to each block's sides to guarantee that the skeletons span across blocks. Next, we thin the volumes in each block with a topological thinning approach that maintains connections between all synapses and anchor points. Lastly, we refine the skeletons globally to attach each neurite's skeleton to the cell body.

5.3.1 Bubble Filling

Many current state-of-the-art segmentation algorithms [32, 49] tend to generate bubbles, i.e., groupings of voxels incorrectly labeled inside a given neuron, of various shapes and sizes. For example, membrane detection algorithms occasionally misclassify mitochondria as cell boundaries. These mislabeled boundaries can cause bubbles in the output during an agglomeration step that transforms these pixel affinities into a segmentation. Since we know that neuronal processes are

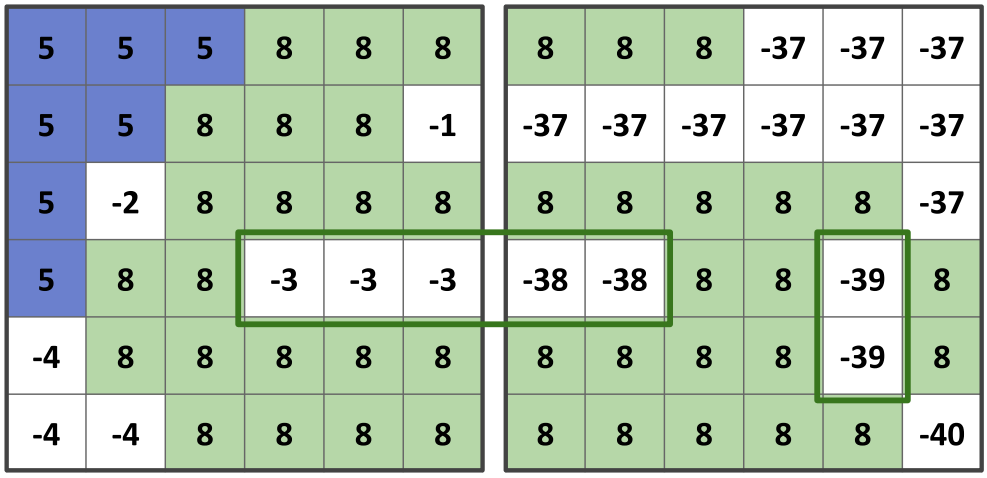


Figure 5.5: We illustrate our block-based bubble-filling algorithm on a 2D example. We cluster all "background" voxels (i.e., those that do not belong to a neuron) into components using a two-pass connected component labeling algorithm. We label these components with a unique negative integer. Since a bubble can span across multiple block boundaries (labels -3 and -38), a global agglomeration step links these components over the entire volume to identify those that are entirely encapsulated by a single neuron. Components that either share a boundary with two neurons (label -2) or leave the volume (labels -4, -1, -37, -40) remain background.

solid volumes, the segmentation should not contain any of these bubbles. We define a bubble as a background component that is entirely encapsulated by a single label. Using the notation from the volume processing community (Figure 5.7), a bubble is a 6-connected background component within a 26-connected object.

We divide the task of bubble filling into three steps, the first and third of which are data-parallel (Figure 5.4 top right box). Since these bubbles in the segmentation can span over multiple blocks, the second step requires global scope. Figure 5.5 illustrates a simple two-dimensional example of the process with two blocks of 36 voxels each and two neuron labels (5 and 8). The white voxels indicate background components that do not belong to any neuron. First, we identify all 6-connected background components per block using a modified version of the two-pass connected component labeling algorithm [111, 138]. In the figure, there are four identified background components in each block. Each background component receives a globally unique, de-

creasing, negative label assigned in raster order. The labels for background components in the second block begin with -37 since there are 36 voxels in the first block, and therefore no background component in that first block could receive label -37. This labeling method extends to all blocks; the total number of voxels in all blocks preceding the current one determines the starting index. This process guarantees that each discovered background component has a globally unique label both within and across blocks. The second step in the bubble filling framework requires global information to link the background components across block boundaries. We link together neighboring background components in different blocks and keep track of the number of neuron neighbors. By definition, any of these background components that have more than one neuron neighbor are not bubbles (Figure 5.5, label: -2). Similarly, any background component that leaves the volume (Figure 5.5, labels: $-1, -4, -37, -40$) are not classified as bubbles. We finally create a mapping between the background components that are bubbles and the corresponding neuron to which it belongs (Figure 5.5, labels: $-3, -38, -39$). With this mapping, we fill the bubbles for each block in parallel across a distributed system.

5.3.2 Soma Segmentation

The synaptic strength between two neurons relies heavily upon the two cell bodies' geodesic distance through a given synapse shared by the two neurons. Therefore, precise somata segmentation is critical for exploring the interplay between two or more neurons. Furthermore, identifying the cell bodies can significantly reduce the time for topological thinning as we can omit their interior points. On two representative datasets, the somata account for approximately 65% of all labeled voxels (Section 5.5.2). Since somata have various shapes and sizes, geometry-based detection algorithms do not adequately identify them.

We train a fully convolutional neural network (CNN) based on a slight modification of the U-Net [110] to identify cell bodies in the input volume. We first downsample the label volume by a

constant factor in x, y, and z. For each query xy image tile, we extract a nine-channel tensor where each channel corresponds to the four nearest xy tiles in both directions. We find that using a nine-channel tensor instead of the 3D U-Net [15] increases inference throughput while maintaining high accuracy. We further increase throughput while maintaining high accuracy by reducing the number of filters per layer by a factor of four. For each neuron label in the query tile, we construct a binary mask for that label as input into our CNN, padding the input tensor with zeros at the boundaries as needed. The CNN outputs a proposed segmentation mask for the cell body for that label. We overlay the outputs for all labels to create a somata segmentation mask. Since neurons have only one cell body, we discard all components in the somata segmentation outside the largest one per neuron.

5.3.3 Synapse-Aware Topological Thinning

We divide the synapse-aware topological thinning strategy into three distinct components (Figure 5.4, bottom right box). The computationally expensive first two steps are data-parallel, while the final step requires global scope. We elaborate on each of the three steps in our thinning framework and the methods for generating geometric attributes about each neurite.

Anchor Point Computation. Our generated skeletons need to be continuous across block boundaries. Therefore, we cannot thin each block entirely independently. Instead, we need to ensure that each neuron's skeleton remains connected across all blocks. Existing block-stitching approaches fail to connect the skeletons across blocks since there is no guarantee that any skeleton points on the block surface will be 26-connected with the adjacent block's skeleton points. Furthermore, the thinned centerline may not even contain any points on the block surface. Therefore, we identify anchor points along each block surface that guarantees that each generated skeleton within a block connects to neighboring blocks (Figure 5.6).

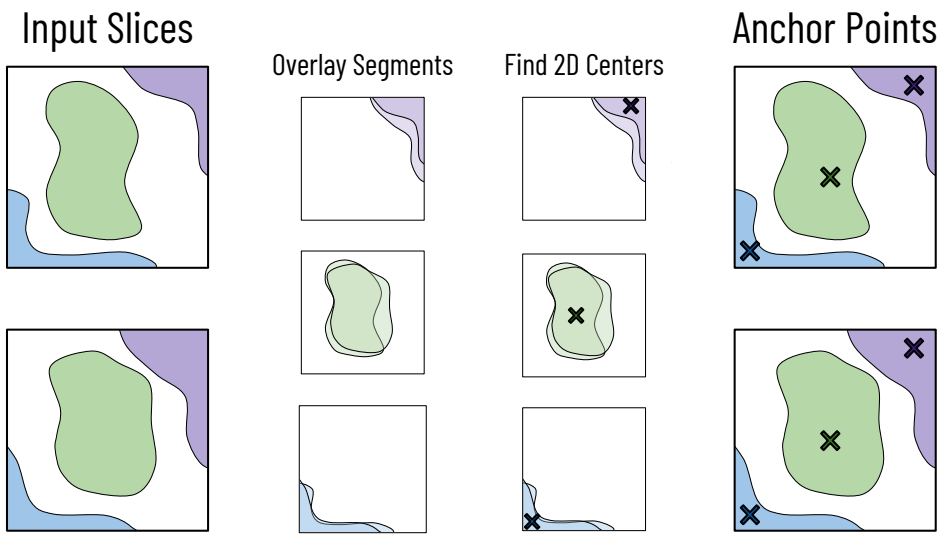


Figure 5.6: We compute anchor points on each block surface to guarantee that the skeletons generated in each block connect across block boundaries. For each pair of adjacent block surfaces, we overlay all labels. We then identify central points using a 2D shrinking strategy for each labeled component on the overlayed block surfaces. These points are non-deletable during the forthcoming 3D thinning to guarantee connectivity across block boundaries.

To find these anchor points between two blocks, we consider the adjacent pair of surfaces for the blocks. We intersect these surfaces to find the set of object points that are 6-connected across the block boundary (Figure 5.6, middle). We calculate these intersected surfaces for each pair of adjacent blocks in the volume. Then, we find the center point for each component on each of the intersected slices in the entire volume. These center points are the geometric centers of the corresponding shapes. We compute these center points using a 2D shrinking approach [87] (algorithm FP-E0). This algorithm guarantees that the computed anchor points fall within the intersected component, even for non-convex shapes. If an object's cross-section spans more than one surface (e.g., an object leaves a block at one of the six corners), we locate anchor points on each surface independently to guarantee continuity across all neighboring blocks. Although this can introduce loops (Figure 5.4), we eliminate these loops at a later step.

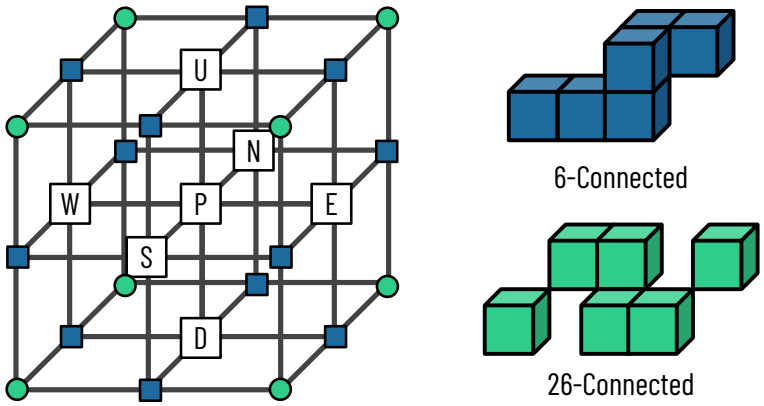


Figure 5.7: Topological thinning algorithms rely only on the local neighborhood around a point p to determine deletion (left). $N_6(p)$ contains the point p and the six points labeled **U**, **D**, **N**, **S**, **E**, and **W**. $N_{18}(p)$ includes the points in $N_6(p)$ and the twelve ■ points. $N_{26}(p)$ is the set of $N_{18}(p)$ and the eight points marked by ●. Examples of 6- and 26-connected points (right).

Topological Thinning. We assume as input a series of binary volumes where the value of '1' is assigned to a voxel if it belongs to a specific neuron. Thus, we create a distinct binary volume for every neuron that we then thin independently to produce a centerline for that neuron. $\mathcal{P}$ is the set of object points in the examined volume; $\overline{\mathcal{P}}$ is the set of background points with a value of '0' is assigned to them.

We define three neighborhoods around each point p which we call $N_6(p)$, $N_{18}(p)$, and $N_{26}(p)$ (Figure 5.7). $N_6(p)$ contains the six points labeled **U**, **D**, **N**, **S**, **E**, and **W**. $N_{18}(p)$ contains the points in $N_6(p)$ and the twelve ■ points. Similarly, $N_{26}(p)$ contains the points in $N_{18}(p)$ and the eight ● points. A sequence of distinct points $\langle p_0, p_1, \ldots, p_m \rangle$ is called a *j-path* from p_0 to p_m in a non-empty set of points X if each point of the sequence is in X and p_i is j-adjacent to p_{i-1} for each $i = 1, 2, \ldots, m$. (Note that a single point is a j-path of length 0.) Two points are said to be *j-connected* in a set X if there is a j-path in X between them. A set of points X is *j-connected* in the set of points $Y \supseteq X$ if any two points in X are j-connected in Y. A *j-component* in a set of

points X is a maximal (with respect to inclusion) j-connected subset in X. Under this notation, an object is a maximal set of object points that are 26-connected, and a background component is a maximal set of background points that are 6-connected. An object point p is called a surface point if $N_6(p) \cap \overline{\mathcal{P}} \neq \emptyset$.

Simple points are object points whose removal from the set $\mathcal{P}$ does not alter the topology. Malandain and Bertrand [73] prove the following theorem to determine if an object point p is simple by examining the set $N_{26}(p)$ (i.e., the simpleness is a local property):

Theorem 1 *A point $p \in \mathcal{P}$ is simple if and only if all of the following conditions hold:*

1. *The set $N_{26}(p) \cap (\mathcal{P} \setminus p)$ contains exactly one 26-component.*
2. *The set $N_6(p) \cap \overline{\mathcal{P}}$ is not empty.*
3. *Any two points in $N_6(p) \cap \overline{\mathcal{P}}$ are 6-connected in the set $N_{18}(p) \cap \overline{\mathcal{P}}$.*

All simple points are surface points by Condition 2 of Theorem 1. An endpoint $p \in P$ contains exactly one object point in $N_{26}(p)$. By Theorem 1, every endpoint is also a simple point. Therefore, with no further deletion restrictions, successive deletion of simple points can reduce an object without any bubbles and tunnels, such as a neuron, to a single point. Therefore, to generate expressive skeletons rather than trivial single point reductions for such an object, researchers have introduced a series of additional constraints to Theorem 1. These additions range from merely preserving endpoints [8] to defining another class of geometric constraints such as non-simple curve-isthmuses [92]. We differ from these previous approaches by introducing additional biologically-inspired constraints that synapse points and somata surface points are always non-simple and thus non-deletable. As discussed previously, we also preserve all anchor points to guarantee connectivity across the entire volume. We remove any other points in each block if

they adhere to the three requirements of Theorem 1. Therefore, our resultant skeleton connects all synapses to the cell body. All skeleton endpoints are anchor points, synapse points, or points on the soma surface at this stage. This synapse-aware skeleton generation strategy produces centerlines that are better suited for higher-level analysis.

We employ a sequential thinning procedure to erode the surface uniformly in all directions [92] for each block. Each iteration consists of six sub-iterations where we consider surface points for possible deletion whose corresponding neighbor at location **U**, **N**, **E**, **S**, **W**, and **D** is a background point (Figure 5.7, left). By Condition 2 of Theorem 1, we know that all simple points must be on the surface. Therefore, we start by collecting all the surface voxels into a set. We iterate over the set of surface voxels for each sub-iteration and only consider those whose neighbor in the proposed direction is a background point. Note, we do not consider voxels outside the block to be background points. We create a set of potentially deletable points (i.e., simple points). After collecting all the simple points in a given sub-iteration, we begin deleting points in this set in order if they are still simple. This dual-pass approach of creating simple points and reconfirming their simplicity is necessary since a point may lose its simple designation as we delete its neighbors. After we delete a point, we add any neighbors now on the surface to the set of surface voxels. Once we consider all potentially deletable points, we move to the next sub-iteration. Figure 5.8 shows a cross-section of a volume that requires four thinning iterations to produce a skeleton point.

We estimate the neurite width at each point along the centerline during this step. Since our topological thinning algorithm gradually erodes the surface evenly in all directions, the output skeleton falls in the neuron's center. The final skeleton point itself can vary slightly based on the thinning order (i.e., the sequence of the sub-iterations: **U**, **N**, **E**, **S**, **W**, and **D**). In Figure 5.8, the two object points immediately west and south of the skeleton point could have remained with a different sub-iteration ordering. Thus, we define the neurite's width at a given skeleton point as two times the closest distance between it and the surface. For each object point, we maintain

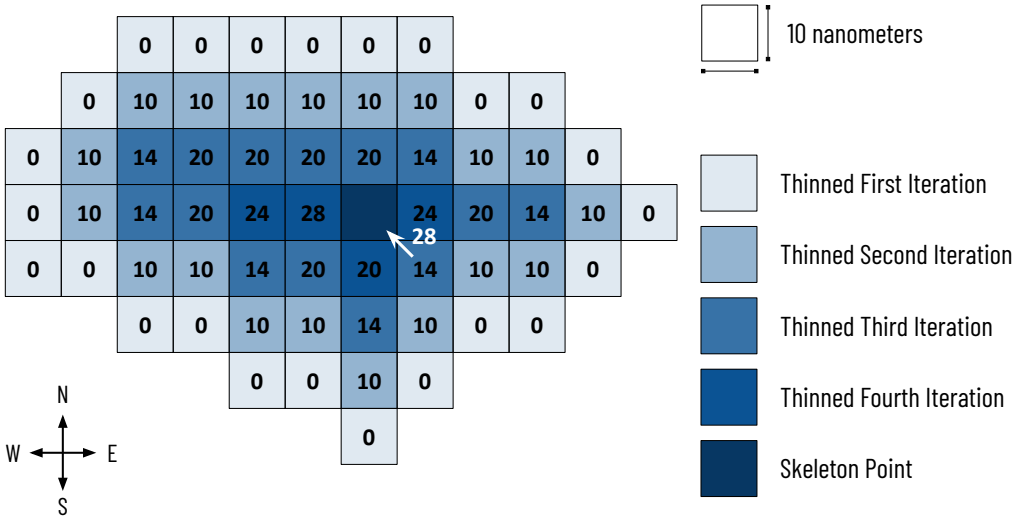


Figure 5.8: We generate estimates for neurite width during thinning. Initially, surface voxels have a distance-to-surface value of zero and all interior voxels ∞. When we remove a surface voxel, we update the distance-to-surface value of neighboring voxels if there is a new shortest path to the surface through the removed voxel. A single skeleton point with a distance of 28 nanometers remains after four thinning iterations in this example. The neurite width is twice the distance-to-surface value.

an estimate for the distance from that point to the surface. These estimates are initially zero for all surface voxels and ∞ for all interior voxels. When a surface point p is deleted during a thinning iteration, we look at its neighborhood $N_{26}(p)$ and update its neighbor's distance-to-surface estimate if the distance to p plus the distance estimate at p is less than its current value. As the surface continues to erode, our estimates reflect the distance from a central point to the surface more accurately. Figure 5.8 shows the skeleton point's final update, which occurs when we delete its southeastern neighbor.

Skeleton Refinement. Although we fill bubbles in the input volume, tunnels that carve through the surface remain since the neuron does not entirely encapsulate them. These tunnels cause loops in the skeleton. Furthermore, our anchor points can cause loops for segments intersecting a corner or edge of a block (Figure 5.4). Since neuronal processes are acyclic, these skeleton loops are

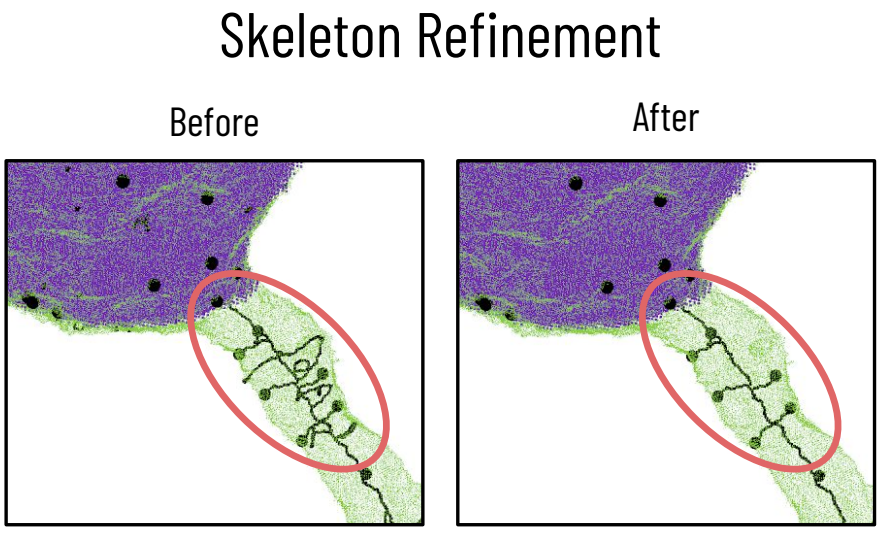


Figure 5.9: Our skeleton refinement step removes any loops in the skeleton caused by tunnels in the segmentation (circled). This step also calculates the geodesic distances between every synapse (black spheres) and the cell body (purple volume).

artifacts of noisy input data and our block linking methodology. We enforce an acyclic constraint on the skeleton and simultaneously produce geodesic distance from each synapse to the cell body during our skeleton refinement phase. We run Dijkstra's shortest path algorithm on the skeleton using the cell body surface as the source. We remove any skeleton point that does not belong on any shortest path from a synapse to the cell body (Figure 5.9). This process eliminates all loops since the shortest paths constructed by Dijkstra's algorithm cannot contain loops. We further produce the geodesic distances from each synapse to the surface of the cell body. This refinement step also removes any endpoints that are not synapses, e.g., anchor points at the periphery of the volume.

Table 5.1: We evaluate our method on three datasets from three different species: rat, fruit fly, and zebra finch. Across all datasets, we generate skeletons for 1,255 neurons and neuron fragments. None of the neuron fragments for FIB-25 contain soma since the cell bodies for the fruit fly lie on the boundary of the brain.

Name	Species	Volume	Resolution	No. Neurons	No. Synapses
JWR	Rat	$106 \times 106 \times 93\,\mu m^3$	$32 \times 64 \times 30\,nm^3$	85	50,334
FIB-25 [124]	Fruit Fly	$36 \times 29 \times 69\,\mu m^3$	$10 \times 10 \times 10\,nm^3$	763	84,157
J0126 [62]	Zebra Finch	$96 \times 98 \times 114\,\mu m^3$	$18 \times 18 \times 20\,nm^3$	407	91,465

5.4 Experimental Set Up

5.4.1 Datasets and Baselines

We evaluate our methods on three large-scale connectomic datasets from three different species: rat, fruit fly, and zebra finch (Table 5.1). Neuroscientists manually segmented and identified the synapses for the JWR (rat) dataset, the smallest of the three with only 34 billion voxels. The FIB-25 (fruit fly) dataset's segmentation and synapses underwent automatic segmentation and detection, followed by extensive human proofreading over the 72 billion voxels. Fully automatic techniques segmented neurons [49] and identified synapses [24] in the most extensive dataset, J0126 (zebra finch), that spans over 165 billion voxels.

We compare our proposed method against two baselines: TEASER [114] and an isthmus-based topological thinning approach [92]. We use the `Kimimaro` implementation of the TEASER algorithm from the Seung lab with the default parameters.* Isthmus (and other topological) thinning methods are particularly susceptible to surface noise, generating many spurious end-points [76, 77]. Therefore, we downsample all three datasets for this baseline to a resolution of $100 \times 100 \times 100\,nm^3$. This downsampling also enables us to compare isthmus thinning against all neurons in our three datasets; without downsampling, CPU memory constraints limit the total number of realizable skeletons.

We ran all timing experiments for our ablation studies on an Intel Core i7-6800K CPU 3.40

*https://github.com/seung-lab/kimimaro

GHz with 64 GB of RAM. All code is written in Python and C++ with Cython wrappers and is freely available.[†] We ran all additional timing analyses on a cluster of 18 Intel Xeon Platinum 8268 CPUs running at 2.90GHz with 188 GB of RAM.

5.4.2 Implementation Details

We train our soma segmentation CNN on 49.6% of the JWR dataset for 40 epochs. We use stochastic gradient descent with ADAM optimizer; learning rate 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$, and $\varepsilon = 10^{-07}$. We downsample the JWR and J0126 datasets by a factor of 4 and 8 in each dimension during soma segmentation.

We test our method on a wide range of block sizes to evaluate the computational efficiency as a function of the number of voxels. For the computational tests summarized in Table 5.3, we use a block size of $896 \times 896 \times 896$ voxels, which is approximately the largest size that could comfortably fit in the memory constraints of our workstation. Outside of CNN training and block size selection, our method requires no other parameters.

5.4.3 Evaluation Metrics

We evaluate our results using three metrics. First, we use the Neural Reconstruction Integrity (NRI) [104] score to evaluate how well each skeleton generation method maintains the brain's underlying wiring diagram. A perfect NRI score of 1.0 indicates that the method preserved all intracellular pathways between synapses without introducing additional connections. This metric illustrates the correspondence between synapses and endpoints. For our baselines, we link synapses to endpoints that fall within 1600 nanometers of each other. Although a cross-section's width at a given location is not well-defined, existing TEASER methods produce estimates for the largest sphere that could fit inside the volume at a given skeleton point [114, 145]. We thus

[†]https://www.rhoana.org/blockbased_synapseaware

Table 5.2: We outperform the baseline methods on the NRI score on all three metrics by significant margins. We improve on the width estimates by 67%, 27%, and 85% over the TEASER algorithm on the three datasets. Although TEASER produces fewer points on the JWR and FIB-25 datasets, we are relatively close to their values.

Method	JWR			FIB-25			J0126		
	NRI (↑)	Width (↓)	Points (↓)	NRI (↑)	Width (↓)	Points (↓)	NRI (↑)	Width (↓)	Points (↓)
Proposed	0.9988	43.03 nm	26,752	0.9952	14.42 nm	11,755	0.9997	25.55 nm	25,562
TEASER	0.1011	120.69 nm	18,250	0.2477	19.78 nm	10,216	0.1729	171.33 nm	33,022
Isthmus Thinning	0.2574	N/A	1,645,966	0.3158	N/A	39,873	0.2454	N/A	1,089,923

evaluate the mean absolute error of our width predictions by finding the closest surface points to each skeleton point, akin to predicting how well we estimate a sphere that could fit inside the neuron centered at that skeleton point. As a measure of simplicity, our final metric counts the number of skeleton points. All other things equal, we prefer fewer skeleton points [92] since, among other reasons, this can significantly reduce the computational costs for algorithms that use the skeletons [23, 76].

5.5 Results

5.5.1 Benchmark Comparison

Table 5.2 summarizes the results on the three datasets of our method and two baselines.

Neural Reconstruction Integrity. Our method achieves a near-perfect one-to-one correspondence between endpoints and synapses. By design, each endpoint in our skeleton is a synapse. However, occasionally, when two synapses are close together, only one synapse will be a skeleton endpoint as the skeleton traverses through the other towards the cell body. Both the TEASER and the isthmus thinning strategies have significantly lower NRI scores ranging from 0.1011 to 0.3158.

Width Estimation. We achieve a mean absolute error of our width (i.e., twice the distance-to-surface) estimation of 43.03 nm, 14.42 nm, and 25.55 nm on the JWR, FIB-25, and J0126 datasets, respectively. The TEASER algorithm outputs a radius for each point, roughly corre-

sponding to the largest sphere wholly contained in the volume centered at that point. The mean absolute error for TEASER's neurite width estimate is 120.69 nm, 19.78 nm, and 171.33 nm for the three datasets, factors of 2.17, 1.37, and 6.71 worse, respectively. The isthmus thinning baseline does not produce width estimates.

Skeleton Simplicity. The TEASER algorithm produces the fewest skeleton points on the JWR and FIB-25 datasets, while our method has the fewest points on J0126. The isthmus thinning strategy produces skeletons with many more points because of the input volumes' tunnels and bubbles. A refinement step similar to the proposed would significantly simplify these skeletons.

Geodesic Distance Calculation. The geodesic distances calculated during skeleton refinement produce more accurate estimates for the distance between a synapse and the cell body than other baseline approximations such as the euclidean distance. On average, the geodesic distance between a synapse and the cell body is 58.50% and 66.16% greater than the euclidean distance on the JWR and J0126 datasets, respectively. Over the entire dataset, the differences in estimated physical path length amount to 20 321.96 nm and 17 238.31 nm per synapse.

Computational Complexity. We evaluate the total CPU time required on our cluster to skeletonize all three datasets using our method and TEASER on blocks of size $1024 \times 1024 \times 1024$. Our method generated skeletons in 8.43 h, 38.49 h, and 62.88 h on the JWR, FIB-25, and J0126 datasets. TEASER took 14.38 h, 35.24 h, and 96.63 h on the JWR, FIB-25, and J0126 datasets. Our method significantly outperforms TEASER on the JWR and J0126 datasets (1.71× and 1.54× quicker) because, in part, the masking of the detected somata dramatically reduces the number of voxels to process during thinning. TEASER outperforms our method on FIB-25 (1.09× quicker), which does not contain any cell bodies.

Qualitative Results. Figure 5.10 shows a skeleton generated for a complete neuron from the J0126 dataset. The black spheres indicate synapse locations. Our skeleton refinement process an-

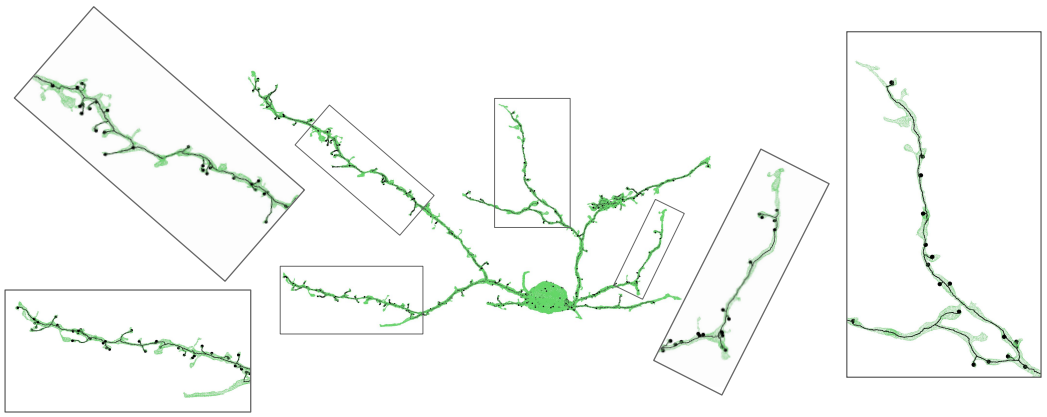


Figure 5.10: Our biologically-aware skeleton generation strategy produces centerlines anchored to the cell bodies that maintain connections between all synapses.

chors the skeletons to the cell body and removes any self-loops in the skeletons caused by tunnels through the neuron surface.

5.5.2 Ablation Studies

Bubble Filling. Many factors contribute to the number of bubbles and their relative sizes in a label volume. The semi-automatic approach used to segment the FIB-25 image volume produces relatively few bubbles (9,525) at just 0.08% of the neuron volume. The JWR and J0126 datasets have 117,568 and 24,149,518 bubbles, accounting for 0.51% and 0.80% of the total volume, respectively. In particular, the flood filling reconstruction algorithm [49] used for J0126 left millions of tiny bubbles—over 85% of the bubbles in the volume contain fewer than five voxels. The topological thinning algorithm forms shells around these bubbles to preserve the number of background components. By eliminating these bubbles, we speed up this step in our pipeline by 11.74%, 1.17%, and 57.16% on the JWR, FIB-25, and J0126 datasets, respectively. The benefits of bubble filling are minimal for FIB-25 since there are only 9, 525 total bubbles. Bubble-filling is computationally less expensive than topological thinning, achieving a throughput of around six

Table 5.3: We provide an analysis of the computational improvements while using the entire skeleton generation pipeline. On J0126, adding bubble filling and soma segmentation reduces the total CPU time by 10.26×.

Method	JWR	FIB-25	J0126
Proposed	5.19 h	N/A	45.29 h
Without Bubble Filling	4.03 h	N/A	78.85 h
Without Soma Segmentation	10.37 h	33.54 h	113.82 h
Topological Thinning Only	20.56 h	30.72 h	479.19 h

million voxels per second. However, since there are relatively few bubbles in the JWR and FIB-25 datasets, this extra bubble filling step increases the total time to skeletonize the volume by 28.78% and 9.17%, respectively (Table 5.3). On the other hand, filling the bubbles in the J0126 volume decreases the total run time by 42.56%. For future datasets, one can sample a small section of the total volume to determine the relative number of bubbles and the potential value of bubble filling for each specific label volume.

Soma Segmentation. Our CNN predicts which voxels belong to cell bodies with 99.28% accuracy (true positive rate: 99.77%, false positive rate: 0.76%) on the saved testing half of the JWR dataset. There are no cell bodies in the FIB-25 dataset. We reduce the time for topological thinning by 49.95% for JWR and 60.21% for J0126 by masking out the detected cell bodies. These cell bodies contain many voxels (64.46% of the total volume, on average) and are not structurally interesting for skeletonization purposes.

Synapse-Aware Topological Thinning. Figure 5.11 illustrates the relationship between topological thinning time and the number of voxels belonging to neurons in a volume. For this set of experiments, we produced skeletons using varying block sizes ranging from $512 \times 512 \times 512$ to $2048 \times 2048 \times 2048$. The y-axes only indicate the time for topological thinning. We achieve an average throughput between $65,000$ and $85,000$ object voxels per second per CPU. These data points consider the number of voxels that belong to a neuron in a given block, not the total number of voxels in a block. Typically, around 85-95% of each dataset's voxels remain unlabeled

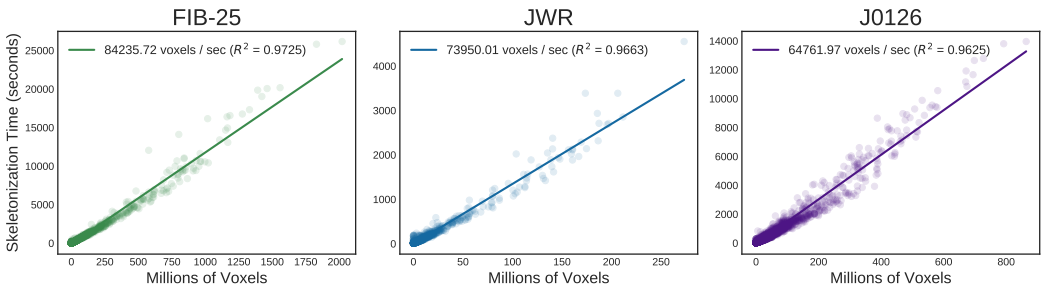


Figure 5.11: Our block-based topological thinning strategy runs linearly with the number of non-zero voxels in the volume. Here, we show the timing results for the three datasets on various block sizes.

because neuroscientists prioritize the reconstruction and proofreading of specific neurons. Without bubble filling and soma segmentation, the end-to-end skeletonization time for the JWR and J0126 datasets increases by a factor of 3.96 and 10.58, respectively (Table 5.3).

5.6 Conclusions

Rapid skeleton generation of reconstructed neural volumes has become an increasingly important component in the connectomic pipeline used for analysis, segmentation evaluation, visualization, and error correction. We propose an efficient, biologically-aware skeleton generation method that produces accurate centerlines while maintaining the neurites' critical geometric properties for further analysis. Our method divides our pipeline into a series of computationally intensive data-parallel tasks and faster recombination steps that require a global scope. Our method significantly improves over existing methods, particularly on automatically generated segmentations that produce an abundance of bubbles. We report over a 10× speed up on one of our datasets over our previous work [77]. Our topological thinning running time per block is empirically linear to the number of voxels in the block, allowing us to scale to more massive datasets without an exploding the number of blocks. Our scalable method can extract biologically-aware skeletons from massive connectomic volumes by distributing computation across an array of CPUs.

6

Conclusions and Future Work

Connectomics has quickly evolved in the last half-century. The process of extracting even a relatively small wiring diagram from *C. elegans* once took over a dozen years of manual labor from a lab of domain experts. Since then, algorithms work concurrently with, and at times replace, humans in the pipeline. These technological advancements have enabled much larger partial connectomes of more evolved species such as fruit fly, mice, birds, and even humans. These exciting new datasets are only possible through the rapid advancements of the algorithms that reconstruct,

proofread, and analyze.

This dissertation proposes four biologically-aware algorithms along the connectomics pipeline that address a handful of the computational challenges. We design our algorithms with careful consideration of the underlying biology to improve scalability and accuracy. One major initial challenge concerns the mere storage of the label volumes. These volumes currently exceed terabytes in size uncompressed, with neuroscientists eagerly planning for datasets 1000× larger. We examine the challenges of working with these massive datasets and propose a method to compress them by over 700× (Chapter 2). Our algorithm relies on the structural properties of neurons and exploits the natural redundancies along their boundaries. Convolutional neural networks have achieved remarkable results in segmenting the image volumes into label volumes. However, even the best methods produce errors at scale. We reformulate the problem of correcting *split errors* as a multicut graph optimization problem to leverage both local and global context (Chapter 3). We use biological constraints while constructing our graph to improve throughput and accuracy (i.e., hand-designed geometric constraints and machine-learned morphologies). After complete reconstruction and synapse detection of the raw image, neuroscientists can produce the wiring diagram. One goal of analyzing these connectomes is to identify motifs—frequently occurring subgraphs with biological significance. We consider the subgraph enumeration problem and augment our input graphs with additional biological attributes to produce more precise and accurate results (Chapter 4). We publish the most extensive summary of motifs in the connectome to date with over 26 trillion subgraphs to aid further inquiries. Along this entire pipeline, skeletonized representations have become increasingly helpful. We introduce a biologically-aware skeleton generation strategy that produces expressive skeletons that maintain vital geometric statistics of the neurites while ensuring desirable biological guarantees (Chapter 5). Our method can help produce more realistic wiring diagrams by measuring the synaptic connectivity between two neurons more accurately.

6.1 Future Work

We present in this dissertation a handful of small steps along the long walk towards a better understanding of the connectome. There is significant room for advancements along each of the discussed dimensions. The primary goal of Compresso is to reduce the disk storage needed. However, visualization of the label volumes can enable better human evaluation of the segmented results. These visualizations, in turn, can lead to additional insights into how to improve the segmentation and error correction components of the pipeline. In a future iteration of Compresso, users should choose the goal of the compression: long-term disk storage or interactive displays. Thus, the next Compresso would allow for random access, which would enable faster interactions with the data in a web-based visualizer, albeit at the cost of reduced compression.

Our hand-designed geometric constraints for error correction work well on mammalian brains. However, other species, such as fruit flies, have neurons that twist and turn in a convoluted mess (Figure B.4). These species require specialized hand-designed constraints separate from those discussed in Chapter 3. Furthermore, synapses and other organelles can provide crucial additional information during the error correction process. We can prevent the local merging of an axon and a dendrite by classifying neuron fragments as either axons or dendrites using the synapses. Eventually, neuroscientists hope to automatically classify neuron cell types using information from the image data, such as mitochondria density. With automatic classification, we could similarly prevent two distinct types of neurons from merging. These enhancements represent just a tiny subset of possible additional biological constraints that we can add.

We extensively enumerate all subgraphs in eleven connectomes from two different species. Within the next decade, there will be partial (or even complete) wiring diagrams at the same scales for many more species. Comparing the distribution of motifs between species is another significant milestone in subgraph analysis. These longitudinal studies will enable us to identify which

frequently occurring subgraphs appear universally. Even intraspecimen, we can contrast brain regions not only by functionality but also by motif frequency. Further, we briefly discuss in Chapter 5 how to improve the wiring diagrams. Simply using the number of synapses is insufficient as the distance to the cell body can significantly alter the perceived synaptic strength. Future analyses should redefine the strengths of synaptic connections using neurite width and geodesic distances.

6.2 Concluding Remarks

The previous decade has seen a rapid increase in partial connectomes from a wide range of species. Spurred by advancements in image acquisition, automatic reconstruction, and synapse detection, neuroscientists now look for more evolved species and more sizable brain regions. These exciting developments do not come without their challenges. At every step of the connectomics pipeline, the forthcoming petabyte datasets appear daunting, demanding new and improved algorithms. Guiding algorithmic design with biological principles can help us meet these challenges while improving accuracy, efficiency, and fidelity. Despite these arising difficulties, daily improvements from researchers worldwide push the boundaries of our knowledge of the brain. With complete connectomes of new species on the horizon, the future of connectomics is bright!

A

Error Correction

A.1 Node Generation and Reduction

Traditional two-step agglomeration strategies tend to create tiny segments at locations where the affinities are noisy, usually at very thin locations. In particular, the waterz agglomeration strategy tends to create many small segments (over 85% of segments are smaller than 0.01 μm^3). Many of these small segments are completely contained within a single z slice. These *singletons* often

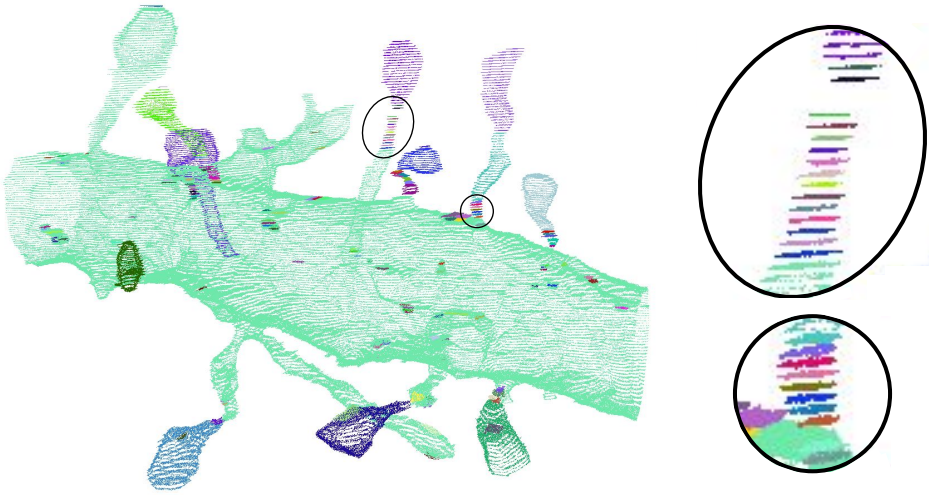


Figure A.1: Current agglomeration strategies tend to produce a large number of singletons in thin locations. Here we show an oversegmented dendrite and two spines with many singletons.

occur on dendritic spines and other thin locations for neuronal processes. Figure A.1 shows an oversegmented dendrite with two enlarged spines, each containing several singletons each. To reduce the number of singletons, we first identify all segments that lie entirely in one z slice. We then superimpose these singleton segments onto the neighboring slices and merge the singleton with any other segments with an Intersection over Union score above 0.30. This threshold reduces a large number of singletons while introducing very few merge errors.

After removing most of the singletons, we further reduce the number of nodes in our graph by identifying small segments and merging them with a large neighbor. We define the threshold for small based on the skeleton generation strategy used during edge generation. Our topological thinning algorithm for skeletonization [92] often erodes very small volumes to a single point. Since we need multiple points in the skeleton to generate our directional vectors, these nodes would not receive edges (Figure A.6, bottom). Therefore we analyze the number of expressive (non-single point) skeletons above and below several thresholds (Figure A.2). We find that over 84.1% of skeletons for segments larger than 0.010 36 μm^3 are expressive, and 90.5% of skeletons

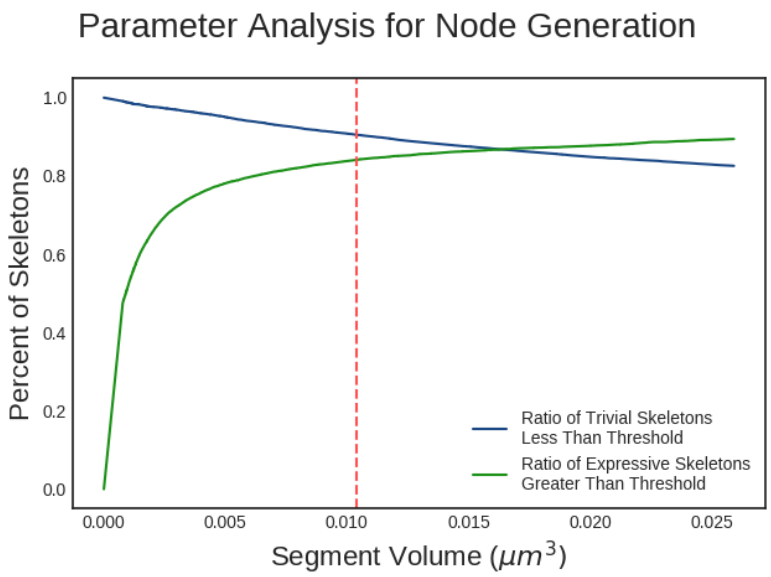


Figure A.2: The percent of trivial and expressive skeletons decreases and increases respectively as the volume of segments increases. At $t_{vol} = 0.010\,36\ \mu\text{m}^3$ nearly 85% of larger segments have expressive skeletons and 90% of smaller ones are trivial.

smaller are trivial. This volume corresponds to $20,000$ voxels in the PNI datasets. On three testing datasets, $50 - 80\%$ of all segments are below this threshold.

A.2 Node Convolutional Neural Network

We experimented with twelve different network architectures and configurations (Table A.1). We evaluate region of interest (ROI) diameters of 800 nm, 1200 nm, and 1600 nm. We find that the overall accuracy of the training, validation, and testing datasets decreases as the ROI increases. We also vary the input size to the network, which changes the number of nodes in the first fully connected layer. As the input size increases, the training accuracy increases, but the validation and testing accuracy increase then decrease. Thus, we see that the highest validation accuracy occurs with a 400 nm ROI and a three-channel input with (60, 60, 20) voxels in each channel. Each

Table A.1: We experiment with twelve different networks for the node CNN. We vary the input size to the network and the diameter of the cubic region of interest. Each experiment ran for 2,000 epochs. Our optimal configuration on the validation data uses an 800 nm ROI with an input size of (60, 60, 20).

800nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9482	0.9343	0.9332
(3, 60, 60, 20)	0.9575	**0.9427**	**0.9451**
(3, 68, 68, 22)	0.9644	0.9346	0.9369
(3, 76, 76, 24)	**0.9752**	0.9386	0.9403
1200nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9224	0.9096	0.9120
(3, 60, 60, 20)	0.9318	0.9144	0.9190
(3, 68, 68, 22)	0.9417	0.9264	0.9265
(3, 76, 76, 24)	0.9553	0.9257	0.9227
1600nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.8960	0.8851	0.8814
(3, 60, 60, 20)	0.9170	0.9097	0.9100
(3, 68, 68, 22)	0.9176	0.9043	0.9053
(3, 76, 76, 24)	0.9423	0.9085	0.9117

ROI from the input segmentation is upsampled and downsampled to fit into this input size. By resampling, we can use identical architecture sizes for new datasets with different resolutions. We find that very limited finetuning is needed to adapt a network to a new dataset.

The general architecture for both networks follows the same design (Figure A.3). Both have three *VGG-style* blocks with two convolutions of (3, 3, 3) followed by a max-pooling operation [13]. The first two max-pooling operations are anisotropic with downsampling only in the x and y dimensions. The final max-pooling is isotropic. A dropout of 0.2 follows each of these blocks, and there is a final dropout of 0.5 after the last fully connected layer. Each activation function is leaky ReLU after every convolution [41] with $\alpha = 0.001$. The final activation is a sigmoid function. We initialize all weights with Xavier initialization [33].

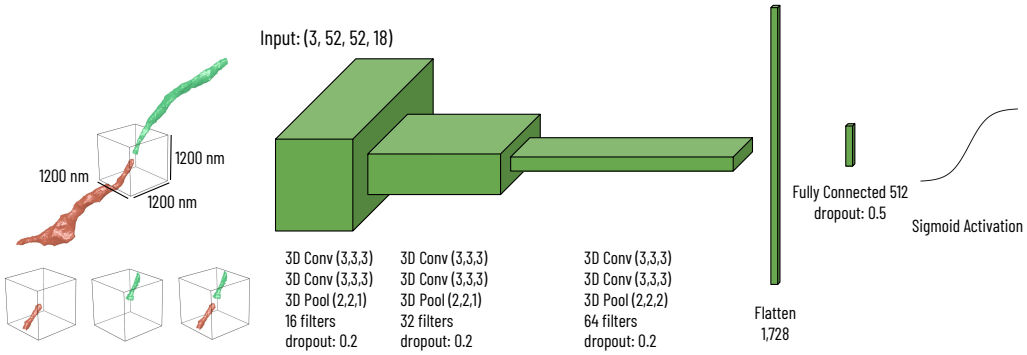


Figure A.3: Both neural networks follow the same general architectures with three *VGG-style* convolution blocks with double convolutions followed by a max-pooling operation. Three input channels correspond to if a voxel has label one, label two, or either. The max-pooling of the convolution blocks is anisotropic for the first two and isotropic for the last.

A.3 Skeleton Benchmark and Generation

Some research focuses on skeletons for quicker connectomics analysis [144] and error correction [23]. Most connectomics literature use the TEASER algorithm [114] or a slight variant from the NeuTu package [145].* A significant amount of research in the computer graphics and volume processing communities considers the problem of extracting the medial axis from a 3D volume [68, 92, 93, 95].

Before this work, to our knowledge, no one has done an extensive analysis of various skeletonization approaches on connectomics data. We create and publish a benchmark dataset for skeletonization on connectomics datasets and evaluate three state-of-the-art 3D skeleton extraction techniques. We consider the 500 largest segments for the ground truth from the Kasthuri training volume for our benchmark. These 500 segments correspond to 95.4% of the volume of the labeled ground truth data. For each of these segments, we identify all endpoints (Figure A.4). We then identify all of the endpoints for each skeleton generation strategy (i.e., those skeleton

*https://github.com/janelia-flyem/NeuTu

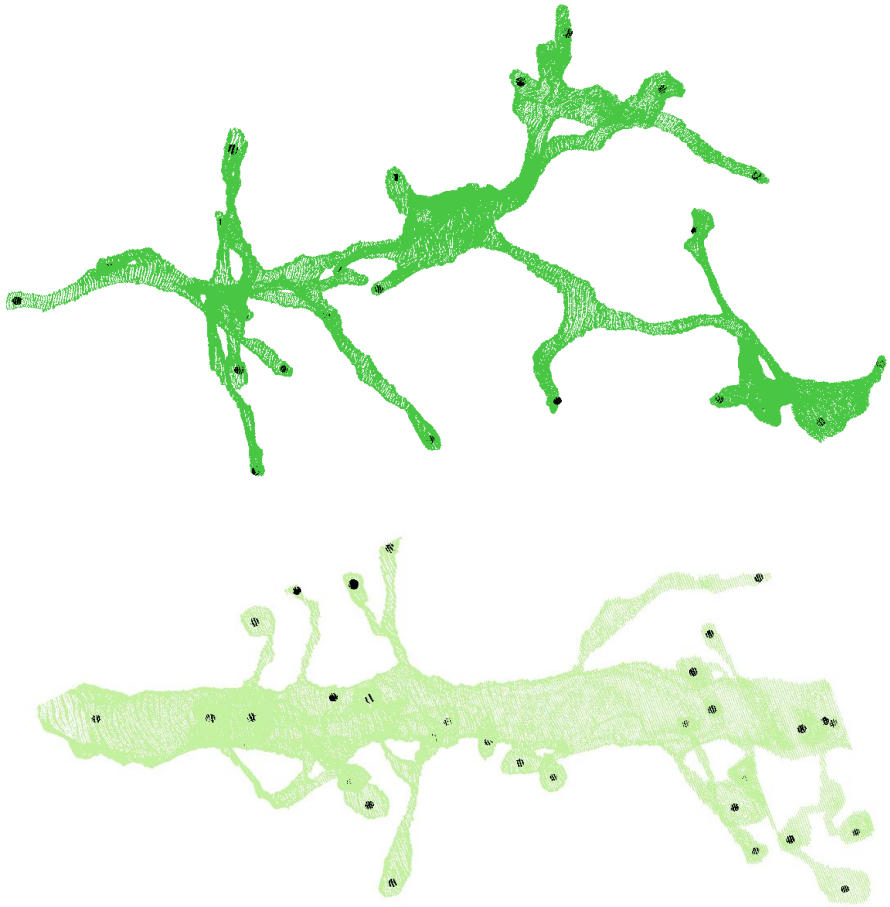


Figure A.4: To evaluate various skeletonization methods, we create and publish a skeleton benchmark. We take a ground truth segmentation from the publically available Kasthuri dataset and label the endpoints for the 500 largest segments. Here we show two ground truth segments and their corresponding labeled endpoints.

points with one or fewer neighbors). For each segment and skeleton strategy, we look at all of the ground truth and generated endpoints and use a Hungarian matching algorithm to create a one-to-one mapping between the two sets [86]. Endpoint pairs closer than 800 nm are considered a match, and ones farther are not.

We evaluate three skeleton generation strategies on our benchmark dataset [68, 92, 114]. The

medial axis algorithm from Lee *et al.* is the built-in scipy skeletonization strategy. We publish the benchmark dataset[†] and all code.[‡] The TEASER algorithm has two tunable parameters, *scale* and *buffer*, which indicate how much pruning to do during generation. Neither the medial axis algorithm nor the topological thinning one has any parameters. However, we consider skeleton generation on segments downsampled to isotropic resolutions between 30 nm and 200 nm per sample. We achieve the highest F-score with the topological thinning approach [92] after downsampling each segment to a resolution of (80, 80, 80) (precision: 94.7%, recall: 86.7%, F-score: 90.5%). In total, we evaluated nearly one thousand different parameter settings on the benchmark dataset. The medial-axis algorithm and the topological thinning approach achieve similar results. However, the topological thinning algorithm is an order of magnitudes faster (4.9 seconds versus 39.8 seconds on the volume downsampled to 80 nm in each dimension). The TEASER algorithm is the slowest, with a running time of 306 seconds on that same volume.

We generate the skeleton endpoint vectors in the following way. Consider an endpoint with one neighbor. That neighbor joint is the parent of the endpoint. We can find the grandparent of the endpoint (i.e., the neighbor of the joint that is not the endpoint). With one more iteration, we find the great-grandparent of an endpoint. The vector between the endpoint and its great-grandparent becomes the endpoint vector used to estimate the direction of the skeleton at endpoint termination.

A.4 Edge Generation

Figure A.5 shows the effect on the number of true split errors identified and the number of total edges as a function of t_{edge}—the maximal allowable distance between a skeleton endpoint and the other neighboring volumes. We consider a wide range of possible distances ranging from 50 nm

[†]https://www.rhoana.org/skeletonbenchmark
[‡]https://www.rhoana.org/biologicalgraphs

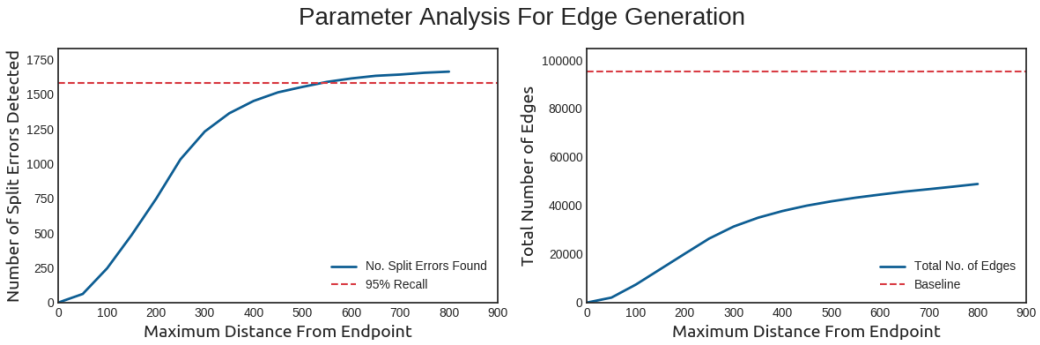


Figure A.5: When considering values for t_{edge}—the maximal distance a voxel can be from an endpoint for edge generation—we look at the recall of true split errors and reduction in total edges. At 500 nm we have 95% recall compared to 800 nm with 12% fewer edges.

to 800 nm on four training datasets. Figure A.5, left, shows the number of edges corresponding to split errors as a function of t_{edge}. The number increases quickly until around 300 nm before increasing more gradually until 800 nm. We do not consider distances farther than 800 nm since we found the remaining missing edges are where the algorithm itself cannot find them (Figure 3.9, bottom; Figure A.6, bottom). The dotted line shows the location of 95% recall from the $t_{edge} = 800$ nm results. Our proposed method uses a value of 500 nm since that is roughly at the 95% recall location.

As we can see from Figure A.5, right, the number of edges in our graph increases significantly for every increase in the distance until around 400 nm where the increase becomes more gradual. For that graph, the baseline is determined from the adjacency matrix. A distance of 500 nm has approximately 12% fewer total edges than using $t_{edge} = 800$ nm.

Figure A.6 shows an additional success and failure case of the proposed edge generation strategy. On the top, we see a neuronal process incorrectly segmented into two. The circled region shows that each segment has a nearby endpoint with a vector pointing towards the incorrectly split neighbor. On the bottom, we see a failure case where the small segment receives a singleton skeleton that is not expressive of the overall shape. This trivial skeleton occurs on a spine where

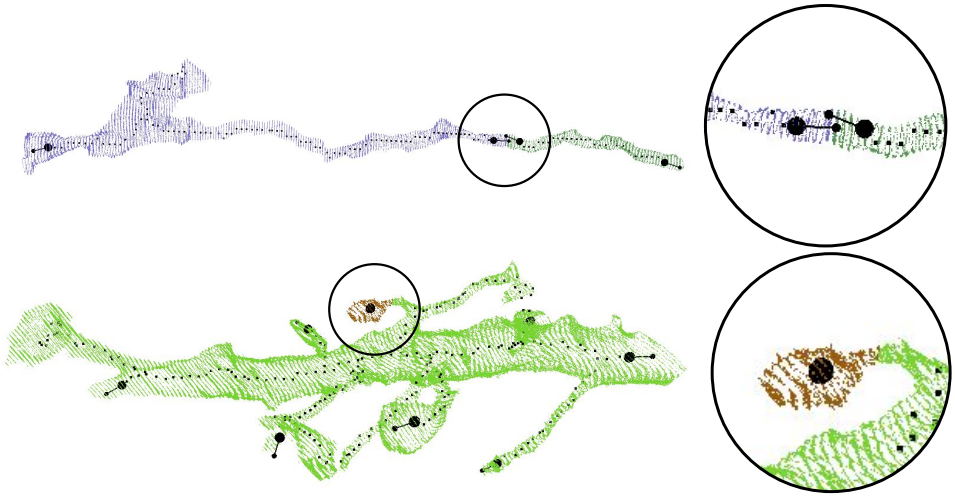


Figure A.6: Here we show two more typical success and failure cases for the edge generation strategy. On the top we see a split neuronal process where both skeletons have endpoints with vectors pointing towards their wrongly split neighbor. On the bottom is an example of a failure case where the segment receives a trivial skeleton with a single endpoint. Since the skeleton is not expressive of the segment shape we have no corresponding vector and this pair of split segments do not receive an edge.

most of the spine correctly merged with the dendrite, and just the top end of the spine is segmented. Since there are no neighboring skeleton endpoints, these two neighboring segments do not receive an edge.

A.5 Edge Weights

There are two components to determining the weights for each edge in the graph. First, we need to generate probabilities that two segments belong to the same neuronal process. Second, we need to transform these probabilities into weights for edges.

Edge CNN. The edge CNN, which determines if two large segments belong to the same neuron, has the same general structure as the node CNN (Figure A.3). There are three *VGG-style* convolution blocks [13] with convolutions of size (3, 3, 3) followed by a max-pooling operation

Table A.2: We decide upon our final edge generation network after looking at eighteen different network configurations. We vary the input size among three options, consider three different cubic ROI diameters, and train both from scratch and finetuning with the node network. The finetuned results are in the even rows. The best validation results occur with a cubic region of 1200 nm in diameter and an input size of (52, 52, 18).

800nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9736	0.9543	0.9576
(3, 52, 52, 18)	0.9750	0.9541	0.9582
(3, 60, 60, 20)	0.9722	0.9586	0.9598
(3, 60, 60, 20)	0.9771	0.9601	0.9626
(3, 68, 68, 22)	0.9740	0.9565	0.9587
(3, 68, 68, 22)	**0.9784**	0.9611	0.9633
1200nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9761	**0.9637**	**0.9638**
(3, 52, 52, 18)	0.9679	0.9553	0.9552
(3, 60, 60, 20)	0.9706	0.9543	0.9546
(3, 60, 60, 20)	0.9703	0.9540	0.9549
(3, 68, 68, 22)	0.9694	0.9544	0.9543
(3, 68, 68, 22)	0.9763	0.9618	0.9629
1600nm Cubic Regions of Interest			
Input Size	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
(3, 52, 52, 18)	0.9597	0.9476	0.9501
(3, 52, 52, 18)	0.9516	0.9377	0.9388
(3, 60, 60, 20)	0.9643	0.9511	0.9526
(3, 60, 60, 20)	0.9589	0.9447	0.9454
(3, 68, 68, 22)	0.9566	0.9363	0.9367
(3, 68, 68, 22)	0.9630	0.9482	0.9476

(anisotropic for the first two blocks and isotropic for the third). All weights have Xavier initializations [33] and each activation is leaky ReLU with $\alpha = 0.001$ [41].

We consider a series of architectures, cubic regions, and initial conditions for the edge CNN and choose the one with the best validation loss (Table A.2). We consider three different input sizes, three different diameters for the cubic region-of-interest (ROI), and finetune a network on

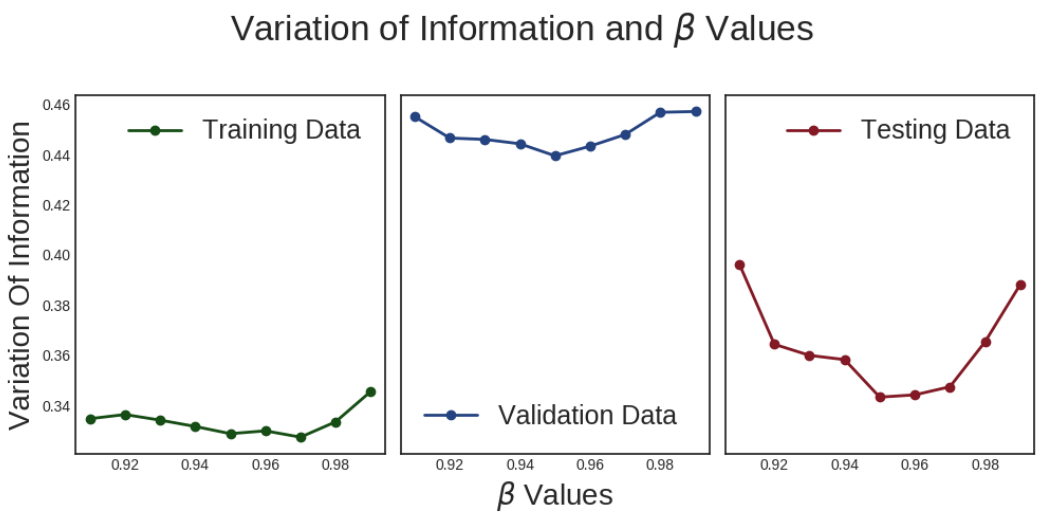


Figure A.7: We consider a wide range of possible β values and see how each affects the total variation of information score. A β value of 0.95 creates the most significant reduction in variation of information on three validation datasets.

the node CNN and train one from scratch. Amazingly, the node CNN produces good results for the new input, with accuracies of 96.3%, 96.6%, and 94.4% on the two PNI testing and one Kasthuri testing datasets. The best validation accuracy has an ROI diameter of 1200 nm and an input size of (52, 52, 18) voxels. As before, the cubic region is extracted from the input segmentation and upsampled and downsampled as needed to fill the input channels for the network.

From Probabilities to Weights. After generating a probability that two nearby segments belong to the same neuron, we need to transform the probability into a weight. As discussed in the Chapter 3, we use the following equation to transform the probabilities into edge weights [57]:

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

Figure A.7 shows the equation as a function of p_e and β. β is a tunable parameter that encourages over- or undersegmentation. Based on the validation data we use $\beta = 0.95$ (Figure A.8). We find that this β value creates the most significant reduction in variation of information on three PNI

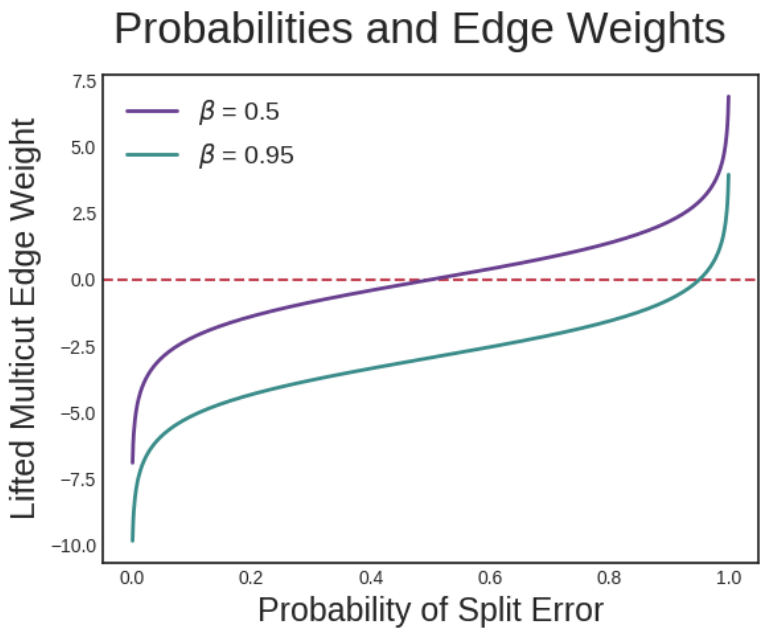


Figure A.8: This function converts probabilities between 0 and 1 to edge weights. β is a tunable parameter that encourages over- or undersegmentation. Based on the results on three validation datasets we set $\beta = 0.95$.

validation datasets.

B

Skeleton Generation

In this appendix, we provide visual comparisons between our synapse-aware skeleton generation method and the two baselines, as well as six additional examples (two from each dataset) of our method.

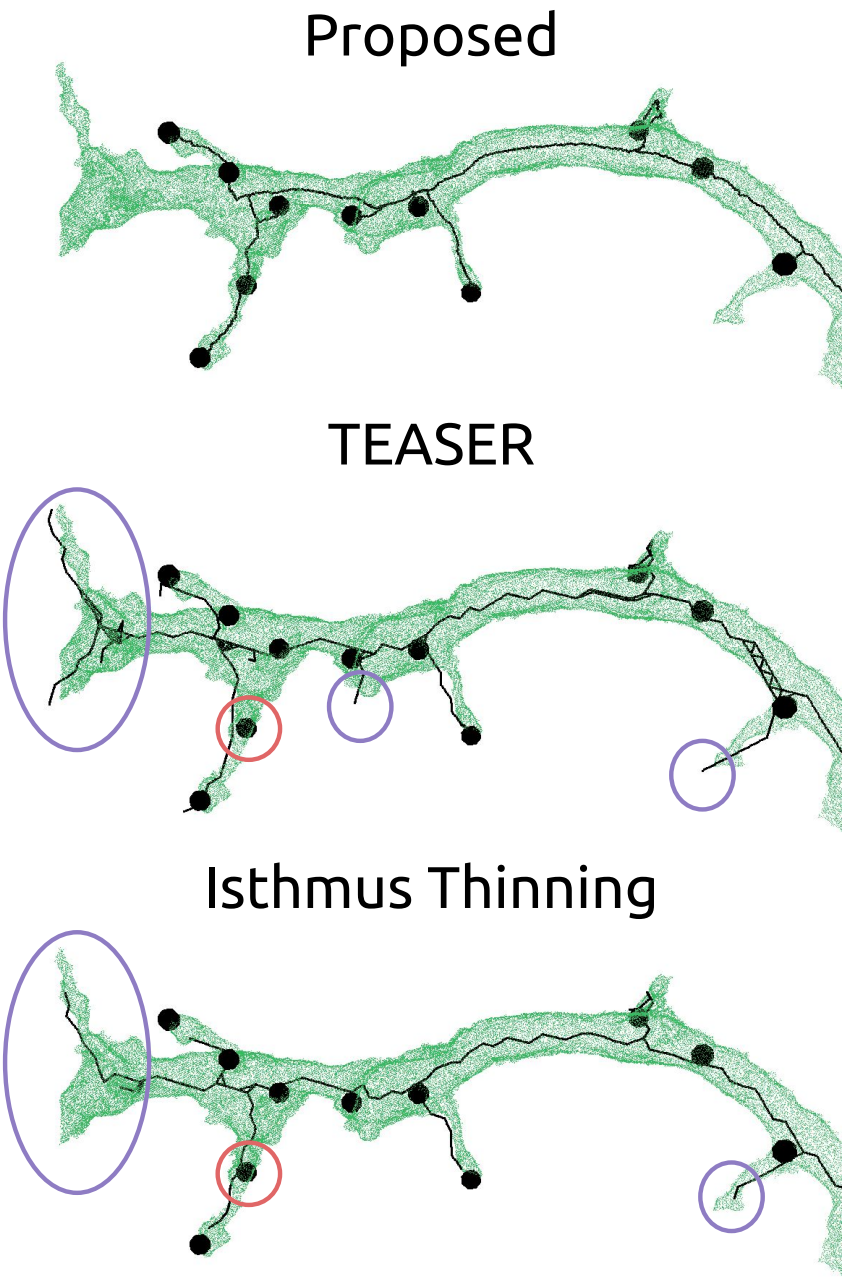


Figure B.1: Here is a side-by-side comparison of our proposed method compared to our two baselines on a neuron fragment from the FIB-25 dataset. Our method guarantees endpoints only at synapse locations. The TEASER and isthmus thinning methods introduce new endpoints (purple circles) and miss others (red circles).

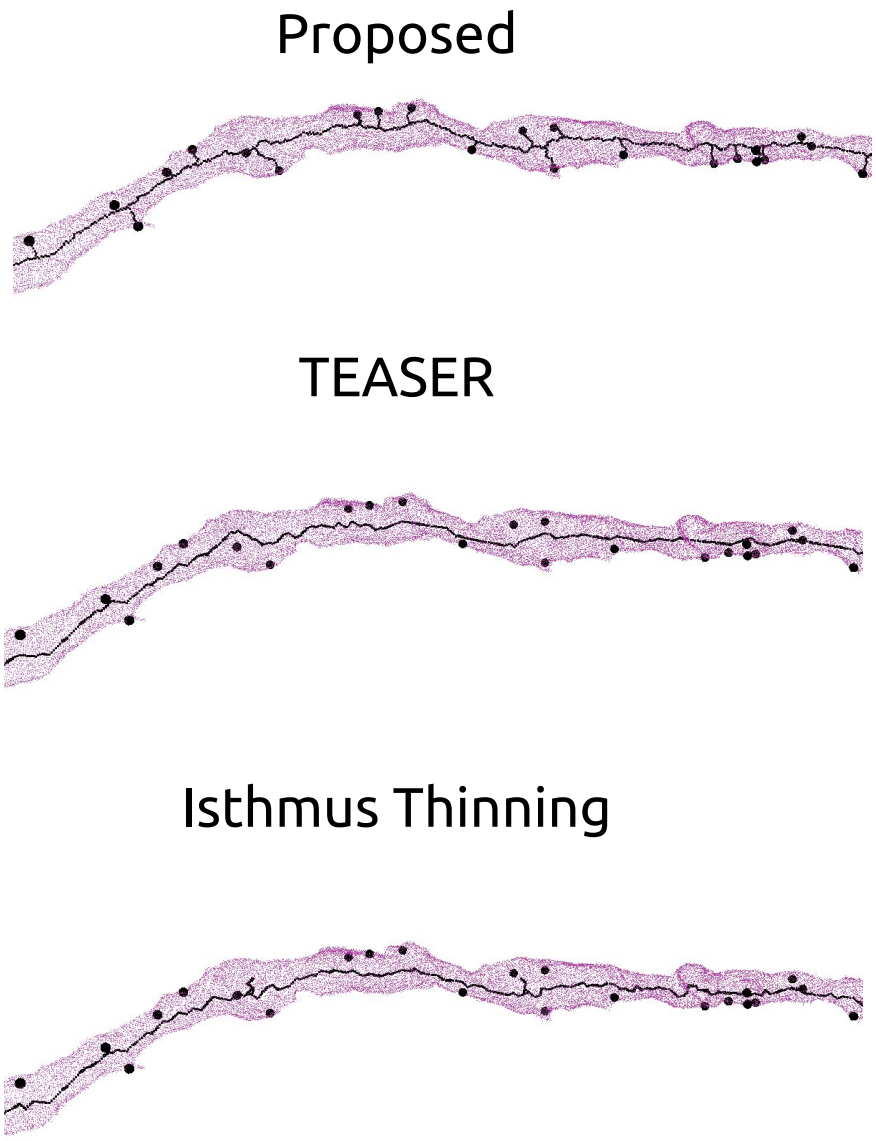


Figure B.2: The two baseline strategies fail to connect any synapses to the skeleton in this segment from JWR. Under the proposed strategy, there is a path between each synapse and the center-line.

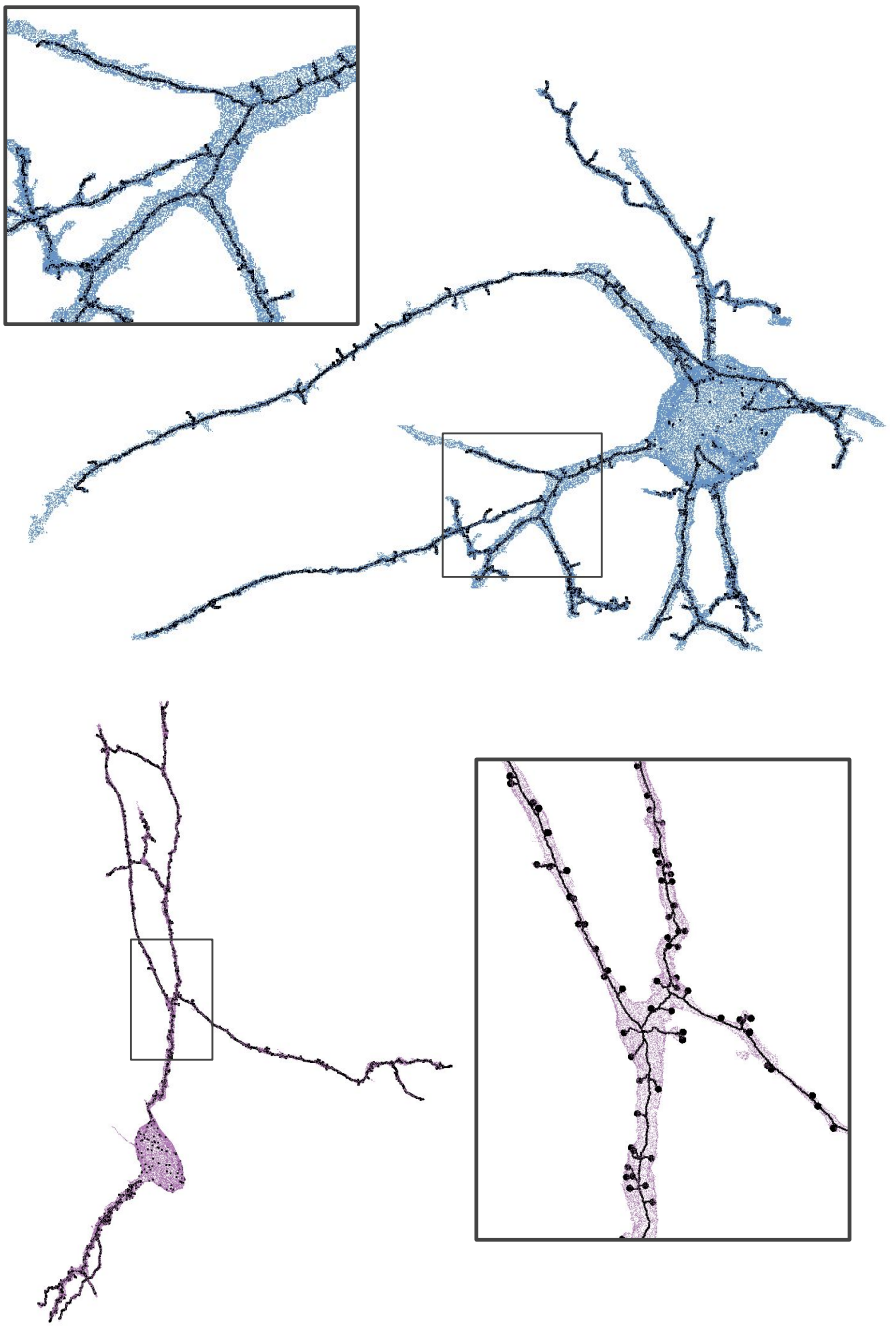


Figure B.3: Here we see two additional examples from the JWR dataset (rat). For both neurons, we also zoom in to one location to show the finer details.

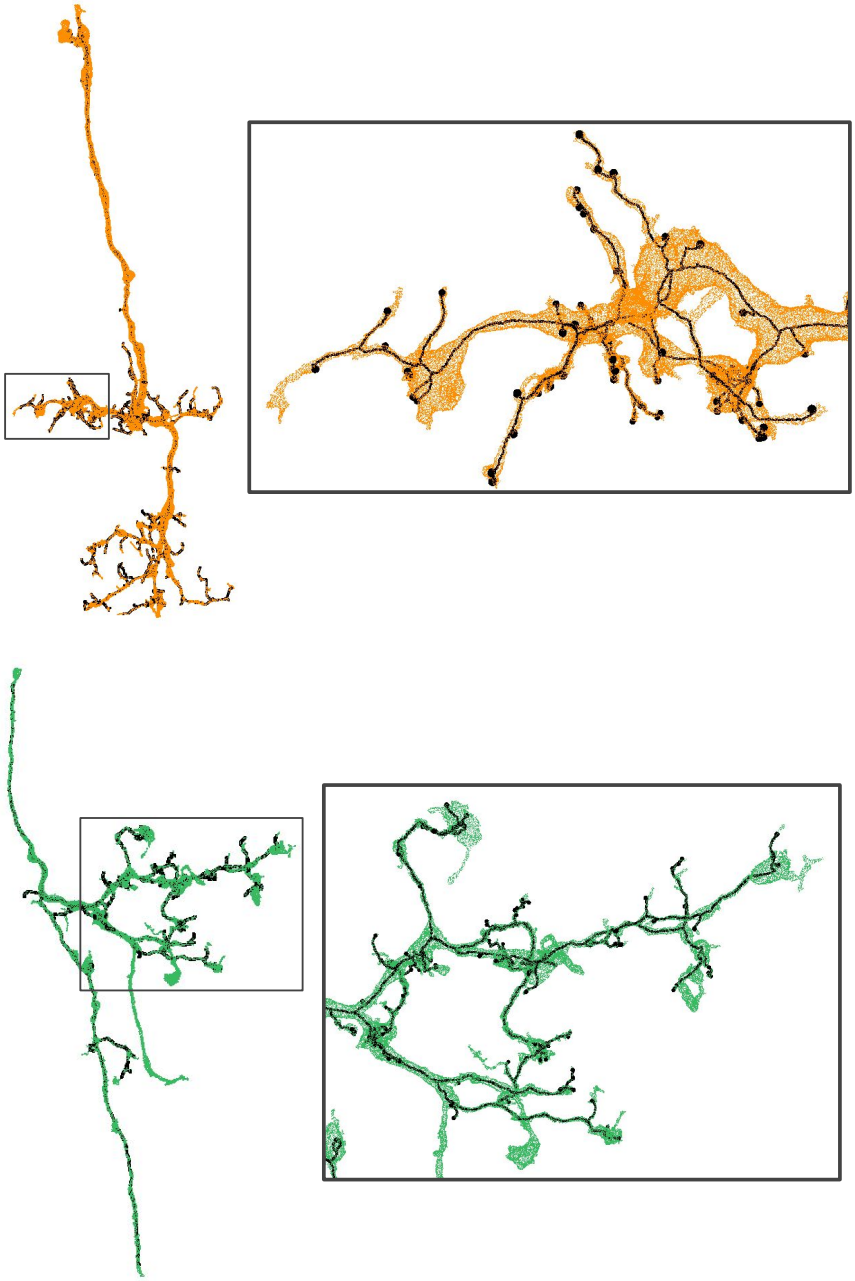


Figure B.4: Here we see two additional examples from the FIB-25 dataset (fruit fly). For both neurons, we also zoom in to one location to show the finer details.

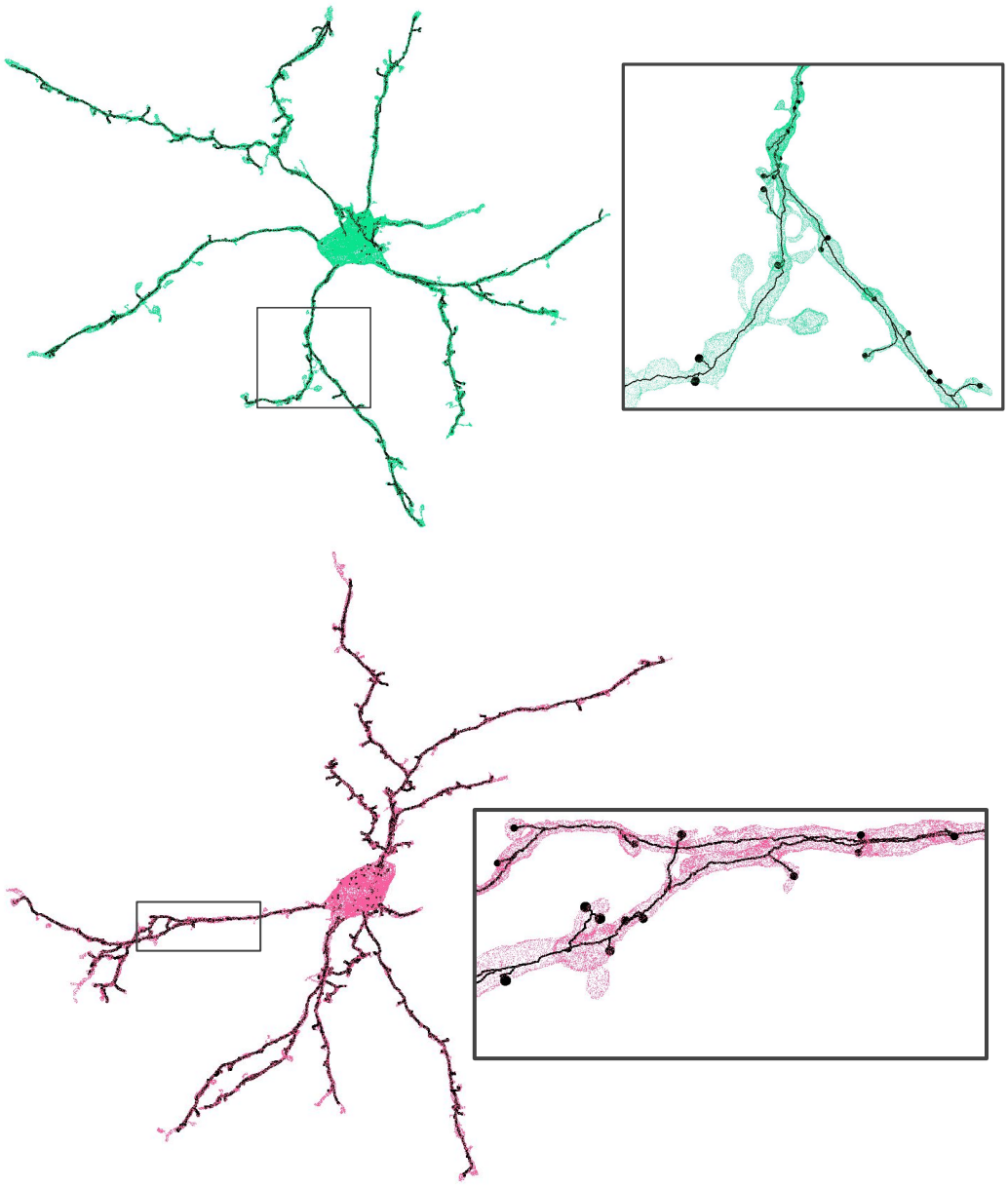


Figure B.5: Here we see two additional examples from the J0126 dataset (zebra finch). For both neurons, we also zoom in to one location to show the finer details.

References

[1] Larry F Abbott, Davi D Bock, Edward M Callaway, Winfried Denk, Catherine Dulac, Adrienne L Fairhall, Ila Fiete, Kristen M Harris, Moritz Helmstaedter, Viren Jain, et al. The mind of a mouse. *Cell*, 182(6):1372–1376, 2020.

[2] Christoph Adami, Jifeng Qian, Matthew Rupp, and Arend Hintze. Information content of colored motifs in complex networks. *Artificial Life*, 17(4):375–390, 2011.

[3] Jyrki Alakuijala, Evgenii Kliuchnikov, Zoltan Szabadka, and Lode Vandevenne. Comparison of brotli, deflate, zopfli, lzma, lzham and bzip2 compression algorithms. *Google Inc*, 2015.

[4] Jyrki Alakuijala, Andrea Farruggia, Paolo Ferragina, Eugene Kliuchnikov, Robert Obryk, Zoltan Szabadka, and Lode Vandevenne. Brotli: A general-purpose data compressor. *ACM Transactions on Information Systems (TOIS)*, 37(1):1–30, 2018.

[5] Bjoern Andres, Thorben Kroeger, Kevin L Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Koethe, and Fred A Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, pages 778–791. Springer, 2012.

[6] László Babai, William M Kantor, and Eugene M Luks. Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*, pages 162–171. IEEE, 1983.

[7] Thorsten Beier, Constantin Pape, Nasim Rahaman, Timo Prange, Stuart Berg, Davi D Bock, Albert Cardona, Graham W Knott, Stephen M Plaza, Louis K Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature methods*, 14(2):101–102, 2017.

[8] Gilles Bertrand and Zouina Aktouf. Three-dimensional thinning algorithm using subfields. In *Vision Geometry III*, volume 2356, pages 113–124. International Society for Optics and Photonics, 1995.

[9] John A Bogovic, Gary B Huang, and Viren Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013.

[10] Kevin Briggman, Winfried Denk, Sebastian Seung, Moritz Helmstaedter, and Srinivas C Turaga. Maximin affinity learning of image segmentation. *Advances in Neural Information Processing Systems*, 22:1865–1873, 2009.

[11] Vincent Casser, Kai Kang, Hanspeter Pfister, and Daniel Haehn. Fast mitochondria detection for connectomics. In *Medical Imaging with Deep Learning*, pages 111–120. PMLR, 2020.

[12] J Leonie Cazemier, Francisco Clascá, and Paul HE Tiesinga. Connectomic analysis of brain networks: Novel techniques and future directions. *Frontiers in neuroanatomy*, 10: 110, 2016.

[13] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[14] Yufei Chen, Cristina Oyarzun Laura, and Klaus Drechsler. Generation of a graph representation from three-dimensional skeletons of the liver vasculature. In *2009 2nd International Conference on Biomedical Engineering and Informatics*, pages 1–5. IEEE, 2009.

[15] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[16] Dan Ciresan, Alessandro Giusti, Luca Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, 25:2843–2851, 2012.

[17] Yann Collet and Chip Turner. Smaller and faster data compression with zstandard. *Facebook Code [online]*, 2016.

[18] Steven J Cook, Travis A Jarrell, Christopher A Brittin, Yi Wang, Adam E Bloniarz, Maksim A Yakovlev, Ken CQ Nguyen, Leo T-H Tang, Emily A Bayer, Janet S Duerr, et al. Whole-animal connectomes of both caenorhabditis elegans sexes. *Nature*, 571(7763):63–71, 2019.

[19] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE transactions on pattern analysis and machine intelligence*, 31(8):1362–1374, 2008.

[20] Brian G Cragg. The density of synapses and neurons in normal, mentally defective ageing human brains. *Brain: a journal of neurology*, 98(1):81–90, 1975.

[21] Sofie Demeyer, Tom Michoel, Jan Fostier, Pieter Audenaert, Mario Pickavet, and Piet Demeester. The index-based subgraph matching algorithm (isma): fast subgraph enumeration in large networks using optimized search trees. *PloS one*, 8(4):e61183, 2013.

[22] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. Technical report, RFC 1950, May, 1996.

[23] Konstantin Dmitriev12, Toufiq Parag, Brian Matejek, Arie E Kaufman12, and Hanspeter Pfister. Efficient correction for em connectomics with skeletal representation. *British Machine Vision Conferemce (BMVC)*, 2018.

[24] Sven Dorkenwald, Philipp J Schubert, Marius F Killinger, Gregor Urban, Shawn Mikula, Fabian Svara, and Joergen Kornfeld. Automated synaptic connectivity inference for volume electron microscopy. *Nature methods*, 14(4):435–442, 2017.

[25] Sven Dorkenwald, Nicholas L Turner, Thomas Macrina, Kisuk Lee, Ran Lu, Jingpeng Wu, Agnes L Bodor, Adam A Bleckert, Derrick Brittain, Nico Kemnitz, et al. Binary and analog variation of synapses between cortical pyramidal neurons. *BioRxiv*, 2019.

[26] Scott W Emmons. The beginning of connectomics: a commentary on white et al.(1986)'the structure of the nervous system of the nematode caenorhabditis elegans'. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 370(1666):20140309, 2015.

[27] Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 8, 2011.

[28] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[29] Alex Fornito, Andrew Zalesky, and Michael Breakspear. Graph analysis of the human connectome: promise, progress, and pitfalls. *Neuroimage*, 80:426–444, 2013.

[30] Alex Fornito, Andrew Zalesky, and Michael Breakspear. The connectomics of brain disorders. *Nature Reviews Neuroscience*, 16(3):159–172, 2015.

[31] Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989.

[32] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1669–1680, 2018.

[33] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[34] Felix Gonda, Donglai Wei, and Hanspeter Pfister. Consistent recurrent neural networks for 3d neuron segmentation. *arXiv preprint arXiv:2102.01021*, 2021.

[35] Weixin Gong and Gilles Bertrand. A simple parallel 3d thinning algorithm. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume 1, pages 188–190. IEEE, 1990.

[36] Google. Neuroglancer compression, url: https://github.com/google/neuroglancer/blob/master/src/neuroglancer/sliceview/ compressed_segmentation/readme.md, 2016. Accessed: 21-October-2016.

[37] Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.

[38] Daniel Haehn, Seymour Knowles-Barley, Mike Roberts, Johanna Beyer, Narayanan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE transactions on visualization and computer graphics*, 20(12):2466–2475, 2014.

[39] Daniel Haehn, John Hoffer, Brian Matejek, Adi Suissa-Peleg, Ali K Al-Awami, Lee Kamentsky, Felix Gonda, Eagon Meng, William Zhang, Richard Schalek, et al. Scalable interactive visualization for connectomics. In *Informatics*, volume 4, page 29. Multidisciplinary Digital Publishing Institute, 2017.

[40] Daniel Haehn, Verena Kaynig, James Tompkin, Jeff W Lichtman, and Hanspeter Pfister. Guided proofreading of automatic segmentations for connectomics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9319–9328, 2018.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[42] Lifeng He, Yuyan Chao, Kenji Suzuki, and Kesheng Wu. Fast connected-component labeling. *Pattern recognition*, 42(9):1977–1987, 2009.

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

[44] Jane Anne Horne, Carlie Langille, Sari McLin, Meagan Wiederman, Zhiyuan Lu, C Shan Xu, Stephen M Plaza, Louis K Scheffer, Harald F Hess, and Ian A Meinertzhagen. A resource for the drosophila antennal lobe provided by the connectome of glomerulus va1v. *Elife*, 7:e37550, 2018.

[45] Gary B Huang, Louis K Scheffer, and Stephen M Plaza. Fully-automatic synapse prediction and validation on a large data set. *Frontiers in neural circuits*, 12:87, 2018.

[46] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[47] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R Berger, Moritz N Helmstaedter, Kevin L Briggman, Winfried Denk, Jared B Bowden, John M Mendenhall, Wickliffe C Abraham, et al. Boundary learning by optimization with topological constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2488–2495. IEEE, 2010.

[48] Viren Jain, Srinivas C Turaga, K Briggman, Moritz Helmstaedter, Winfried Denk, and H Seung. Learning to agglomerate superpixel hierarchies. *Advances in Neural Information Processing Systems*, 24:648–656, 2011.

[49] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8): 605–610, 2018.

[50] Dakai Jin and Punam K Saha. A new fuzzy skeletonization algorithm and its applications to medical imaging. In *International Conference on Image Analysis and Processing*, pages 662–671. Springer, 2013.

[51] Dakai Jin, Cheng Chen, Eric A Hoffman, and Punam K Saha. Curve skeletonization using minimum-cost path. *Skeletonization*, pages 151–180, 2017.

[52] Tihana Jovanic, Casey Martin Schneider-Mizell, Mei Shao, Jean-Baptiste Masson, Gennady Denisov, Richard Doty Fetter, Brett Daren Mensh, James William Truman, Albert Cardona, and Marta Zlatic. Competitive disinhibition mediates behavioral choice and sequences in drosophila. *Cell*, 167(3):858–870, 2016.

[53] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[54] Zahra Razaghi Moghadam Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Saberi Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. Kavosh: a new algorithm for finding network motifs. *BMC bioinformatics*, 10(1):1–12, 2009.

[55] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

[56] Thomas Kemen, Matt Malloy, Brad Thiel, Shawn Mikula, Winfried Denk, Gregor Dellemann, and Dirk Zeidler. Further advancing the throughput of a multi-beam sem. In *Metrology, Inspection, and Process Control for Microlithography XXIX*, volume 9424, page 94241U. International Society for Optics and Photonics, 2015.

[57] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015.

[58] Khaled Khairy, Gennady Denisov, and Stephan Saalfeld. Joint deformable registration of large em image volumes: a matrix solver approach. *arXiv preprint arXiv:1804.10019*, 2018.

[59] Seymour Knowles-Barley, Mike Roberts, Narayanan Kasthuri, Dongil Lee, Hanspeter Pfister, and Jeff W Lichtman. Mojo 2.0: Connectome annotation tool. *Frontiers in Neuroinformatics*, 60:1, 2013.

[60] Seymour Knowles-Barley, Verena Kaynig, Thouis Ray Jones, Alyssa Wilson, Joshua Morgan, Dongil Lee, Daniel Berger, Narayanan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. Rhoananet pipeline: Dense automatic neural annotation. *arXiv preprint arXiv:1611.06973*, 2016.

[61] Christof Koch. *Biophysics of computation: information processing in single neurons*. Oxford university press, 2004.

[62] Jörgen Kornfeld, Sam E Benezra, Rajeevan T Narayanan, Fabian Svara, Robert Egger, Marcel Oberlaender, Winfried Denk, and Michael A Long. Em connectomics reveals axonal target variation in a sequence-generating network. *Elife*, 6:e24364, 2017.

[63] Michel Koskas, Gilles Grasseau, Etienne Birmelé, Sophie Schbath, and Stéphane Robin. Nemo: Fast count of network motifs. *Book of Abstracts for Journées Ouvertes Biologie Informatique Mathématiques (JOBIM)*, 2011:53–60, 2011.

[64] NE Krasowski, Thorsten Beier, GW Knott, Ullrich Köthe, Fred A Hamprecht, and Anna Kreshuk. Neuron segmentation with high-level biological priors. *IEEE transactions on medical imaging*, 37(4):829–839, 2017.

[65] Kirill Kryukov, Mahoko Takahashi Ueda, So Nakagawa, and Tadashi Imanishi. Sequence compression benchmark (scb) database—a comprehensive evaluation of reference-free compressors for fasta-formatted sequences. *GigaScience*, 9(7):giaa072, 2020.

[66] Kisuk Lee, Aleksandar Zlateski, Ashwin Vishwanathan, and H Sebastian Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary detection. *arXiv preprint arXiv:1508.04843*, 2015.

[67] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[68] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[69] Marc Lehmann. Liblzf, url: http://oldhome.schmorp.de/marc/liblzf.html, 2016. accessed: 13-October-2016.

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

[71] Jeff W Lichtman, Hanspeter Pfister, and Nir Shavit. The big data challenges of connectomics. *Nature neuroscience*, 17(11):1448–1454, 2014.

[72] Zudi Lin, Donglai Wei, Won-Dong Jang, Siyan Zhou, Xupeng Chen, Xueying Wang, Richard Schalek, Daniel Berger, Brian Matejek, Lee Kamentsky, et al. Two stream active query suggestion for active learning in connectomics. 2020.

[73] Grégoire Malandain and Gilles Bertrand. Fast characterization of 3d simple points. In *11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*, volume 1, pages 232–235. IEEE Computer Society, 1992.

[74] Dror Marcus and Yuval Shavitt. Rage–a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.

[75] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[76] Brian Matejek, Daniel Haehn, Haidong Zhu, Donglai Wei, Toufiq Parag, and Hanspeter Pfister. Biologically-constrained graphs for global connectomics reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2089–2098, 2019.

[77] Brian Matejek, Donglai Wei, Xueying Wang, Jinglin Zhao, Kálmán Palágyi, and Hanspeter Pfister. Synapse-aware skeleton generation for neural circuits. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 227–235. Springer, 2019.

[78] Brendan D McKay and Adolfo Piperno. Nauty and traces user's guide (version 2.5). *Computer Science Department, Australian National University, Canberra, Australia*, 2013.

[79] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

[80] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003.

[81] Yaron Meirovitch, Alexander Matveev, Hayk Saribekyan, David Budden, David Rolnick, Gergely Odor, Seymour Knowles-Barley, Thouis Raymond Jones, Hanspeter Pfister, Jeff William Lichtman, et al. A multi-pass approach to large-scale connectomics. *arXiv preprint arXiv:1612.02120*, 2016.

[82] Yaron Meirovitch, Lu Mi, Hayk Saribekyan, Alexander Matveev, David Rolnick, and Nir Shavit. Cross-classification clustering: An efficient multi-object tracking technique for 3-d instance segmentation in connectomics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8425–8435, 2019.

[83] Loren Merritt and Rahul Vanam. x264: A high performance h. 264/avc encoder. *online] http://neuron2. net/library/avc/overview_x264_v8_5. pdf*, 2006.

[84] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594): 824–827, 2002.

[85] Haneen Mohammed, Ali K Al-Awami, Johanna Beyer, Corrado Cali, Pierre Magistretti, Hanspeter Pfister, and Markus Hadwiger. Abstractocyte: A visual tool for exploring nanoscale astroglial cells. *IEEE transactions on visualization and computer graphics*, 24 (1):853–861, 2017.

[86] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[87] Gábor Németh, Péter Kardos, and Kálmán Palágyi. 2d parallel thinning and shrinking based on sufficient conditions for topology preservation. *Acta Cybernetica*, 20(1):125–144, 2011.

[88] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8): e71715, 2013.

[89] Juan Nunez-Iglesias, Ryan Kennedy, Stephen M Plaza, Anirban Chakraborty, and William T Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8:34, 2014.

[90] MFXJ Oberhumer. Lzo-a real-time data compression library. *http://www. oberhumer. com/opensource/lzo/*, 2008.

[91] Mark Ortmann and Ulrik Brandes. Efficient orbit-aware triad and quad census in directed and undirected graphs. *Applied network science*, 2(1):1–17, 2017.

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

[93] Kálmán Palágyi and Attila Kuba. A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, 19(7):613–627, 1998.

[94] Kálmán Palágyi, Emese Balogh, Attila Kuba, Csongor Halmai, Balázs Erdőhelyi, Erich Sorantin, and Klaus Hausegger. A sequential 3d thinning algorithm and its medical applications. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 409–415. Springer, 2001.

[95] Kálmán Palágyi, Gábor Németh, and Péter Kardos. Topology preserving parallel 3d thinning algorithms. In *Digital Geometry Algorithms*, pages 165–188. Springer, 2012.

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

[97] Toufiq Parag, Anirban Chakraborty, Stephen Plaza, and Louis Scheffer. A context-aware delayed agglomeration framework for electron microscopy segmentation. *PloS one*, 10(5): e0125825, 2015.

[98] Toufiq Parag, Fabian Tschopp, William Grisaitis, Srinivas C Turaga, Xuewen Zhang, Brian Matejek, Lee Kamentsky, Jeff W Lichtman, and Hanspeter Pfister. Anisotropic em segmentation by 3d affinity learning and agglomeration. *arXiv preprint arXiv:1707.08935*, 2017.

[99] Toufiq Parag, Daniel Berger, Lee Kamentsky, Benedikt Staffler, Donglai Wei, Moritz Helmstaedter, Jeff W Lichtman, and Hanspeter Pfister. Detecting synapse location and connectivity by signed proximity estimation and pruning with deep nets. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[100] Pedro Paredes and Pedro Ribeiro. Rand-fase: fast approximate subgraph census. *Social Network Analysis and Mining*, 5(1):17, 2015.

[101] Igor Pavlov. Lzma sdk (software development kit), 2007.

[102] Stephen M Plaza and Jan Funke. Analyzing image segmentation for connectomics. *Frontiers in neural circuits*, 12:102, 2018.

[103] Jifeng Qian, Arend Hintze, and Christoph Adami. Colored motifs reveal computational building blocks in the c. elegans brain. *PLoS One*, 6(3):e17013, 2011.

[104] Elizabeth P Reilly, Jeffrey S Garretson, William R Gray Roncal, Dean M Kleissas, Brock A Wester, Mark A Chevillet, and Matthew J Roos. Neural reconstruction integrity: A metric for assessing the connectivity accuracy of reconstructed neural networks. *Frontiers in neuroinformatics*, 12:74, 2018.

[105] Pedro Ribeiro and Fernando Silva. G-tries: an efficient data structure for discovering network motifs. In *Proceedings of the 2010 ACM symposium on applied computing*, pages 1559–1566, 2010.

[106] Pedro Ribeiro and Fernando Silva. G-tries: a data structure for storing and finding subgraphs. *Data Mining and Knowledge Discovery*, 28(2):337–377, 2014.

[107] Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. A survey on subgraph counting: concepts, algorithms and applications to network motifs and graphlets. *arXiv preprint arXiv:1910.13011*, 2019.

[108] Greg Roelofs. *PNG: the definitive guide*. O'Reilly Media, 1999.

[109] David Rolnick, Yaron Meirovitch, Toufiq Parag, Hanspeter Pfister, Viren Jain, Jeff W Lichtman, Edward S Boyden, and Nir Shavit. Morphological error detection in 3d segmentations. *arXiv preprint arXiv:1705.10882*, 2017.

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

[111] Azriel Rosenfeld and John L Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.

[112] Stephan Saalfeld, Richard Fetter, Albert Cardona, and Pavel Tomancak. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature methods*, 9(7):717–720, 2012.

[113] Punam K Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. *Skeletonization: Theory, methods and applications*. Academic Press, 2017.

[114] Mie Sato, Ingmar Bitter, Michael A Bender, Arie E Kaufman, and Masayuki Nakajima. Teasar: Tree-structure extraction algorithm for accurate and robust skeletons. In *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, pages 281–449. IEEE, 2000.

[115] Richard Schalek, Dong Lee, Narayanan Kasthuri, Adi Peleg, T Jones, Verena Kaynig, Daniel Haehn, Hanspeter Pfister, David Cox, and Jeff W Lichtman. Imaging a 1 mm 3 volume of rat cortex using a multibeam sem. *Microscopy and Microanalysis*, 22(S3): 582–583, 2016.

[116] Louis K Scheffer, C Shan Xu, Michal Januszewski, Zhiyuan Lu, Shin-ya Takemura, Kenneth J Hayworth, Gary B Huang, Kazunori Shinomiya, Jeremy Maitlin-Shepard, Stuart Berg, et al. A connectome and analysis of the adult drosophila central brain. *Elife*, 9: e57443, 2020.

[117] Casey M Schneider-Mizell, Agnes L Bodor, Forrest Collman, Derrick Brittain, Adam A Bleckert, Sven Dorkenwald, Nicholas L Turner, Thomas Macrina, Kisuk Lee, Ran Lu, et al. Chandelier cell anatomy and function reveal a variably distributed but common signal. *bioRxiv*, 2020.

[118] Julian Seward. bzip2 and libbzip2. *avaliable at http://www. bzip. org*, 1996.

[119] Saeed Shahrivari and Saeed Jalili. Fast parallel all-subgraph enumeration using multicore machines. *Scientific Programming*, 2015, 2015.

[120] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[121] Sen Song, Per Jesper Sjöström, Markus Reigl, Sacha Nelson, and Dmitri B Chklovskii. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol*, 3(3):e68, 2005.

[122] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS Comput Biol*, 1(4):e42, 2005.

[123] Adi Suissa-Peleg, Daniel Haehn, Seymour Knowles-Barley, Verena Kaynig, Thouis R Jones, Alyssa Wilson, Richard Schalek, Jeffery W Lichtman, and Hanspeter Pfister. Automatic neural reconstruction from petavoxel of electron microscopy data. *Microscopy and Microanalysis*, 22(S3):536–537, 2016.

[124] Shin-ya Takemura, C Shan Xu, Zhiyuan Lu, Patricia K Rivlin, Toufiq Parag, Donald J Olbris, Stephen Plaza, Ting Zhao, William T Katz, Lowell Umayam, et al. Synaptic circuits and their variations within different columns in the visual system of drosophila. *Proceedings of the National Academy of Sciences*, 112(44):13711–13716, 2015.

[125] Abhimanyu Talwar, Zudi Lin, Donglai Wei, Yuesong Wu, Bowen Zheng, Jinglin Zhao, Won-Dong Jang, Xueying Wang, Jeff Lichtman, and Hanspeter Pfister. A topological nomenclature for 3d shape analysis in connectomics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 986–987, 2020.

[126] Andreas S Thum and Bertram Gerber. Connectomics and function of a memory network: the mushroom body of larval drosophila. *Current opinion in neurobiology*, 54:146–154, 2019.

[127] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural computation*, 22(2):511–538, 2010.

[128] Vandevenne and Alakuijala. Zopfli compression algorithm, url: https://github.com/google/zopfli, 2016. Accessed: 11-October-2016.

[129] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the caenorhabditis elegans neuronal network. *PLoS Comput Biol*, 7(2):e1001066, 2011.

[130] Christopher S von Bartheld, Jami Bahney, and Suzana Herculano-Houzel. The search for true numbers of neurons and glial cells in the human brain: a review of 150 years of cell counting. *Journal of Comparative Neurology*, 524(18):3865–3895, 2016.

[131] Jianxin Wang, Yuannan Huang, Fang-Xiang Wu, and Yi Pan. Symmetry compression method for discovering network motifs. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(6):1776–1789, 2012.

[132] Tie Wang, Jeffrey W Touchman, Weiyi Zhang, Edward B Suh, and Guoliang Xue. A parallel algorithm for extracting transcriptional regulatory network motifs. In *Fifth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'05)*, pages 193–200. IEEE, 2005.

[133] Donglai Wei, Zudi Lin, Daniel Franco-Barranco, Nils Wendt, Xingyu Liu, Wenjie Yin, Xin Huang, Aarush Gupta, Won-Dong Jang, Xueying Wang, et al. Mitoem dataset: Large-scale 3d mitochondria instance segmentation from em images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 66–76. Springer, 2020.

[134] Terry A. Welch. Technique for high-performance data compression. *Computer*, (52), 1984.

[135] Sebastian Wernicke and Florian Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[136] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philos Trans R Soc Lond B Biol Sci*, 314(1165):1–340, 1986.

[137] Daniel Witvliet, Ben Mulcahy, James K Mitchell, Yaron Meirovitch, Daniel K Berger, Yuelong Wu, Yufang Liu, Wan Xian Koh, Rajeev Parvathala, Douglas Holmyard, et al. Connectomes across development reveal principles of brain maturation in c. elegans. *BioRxiv*, 2020.

[138] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Two strategies to speed up connected component labeling algorithms. Technical report, Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US), 2005.

[139] Wenjie Xie, Robert P Thompson, and Renato Perucchio. A topology-preserving parallel 3d thinning algorithm for extracting the curve skeleton. *Pattern Recognition*, 36(7):1529–1544, 2003.

[140] C Shan Xu, Kenneth J Hayworth, Zhiyuan Lu, Patricia Grob, Ahmed M Hassan, José G García-Cerdán, Krishna K Niyogi, Eva Nogales, Richard J Weinberg, and Harald F Hess. Enhanced fib-sem systems for large-volume 3d imaging. *Elife*, 6:e25916, 2017.

[141] C Shan Xu, Michal Januszewski, Zhiyuan Lu, Shin-ya Takemura, Kenneth Hayworth, Gary Huang, Kazunori Shinomiya, Jeremy Maitin-Shepard, David Ackerman, Stuart Berg, et al. A connectome of the adult drosophila central brain. *BioRxiv*, 2020.

[142] Wenjing Yin, Derrick Brittain, Jay Borseth, Marie E Scott, Derric Williams, Jedediah Perkins, Christopher S Own, Matthew Murfitt, Russel M Torres, Daniel Kapner, et al. A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nature communications*, 11(1):1–12, 2020.

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

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

[145] Ting Zhao, Donald J Olbris, Yang Yu, and Stephen M Plaza. Neutu: software for collaborative, large-scale, segmentation-based connectome reconstruction. *Frontiers in Neural Circuits*, 12:101, 2018.

[146] Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, et al. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell*, 174(3):730–743, 2018.

[147] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[148] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.

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

[150] Jonathan Zung, Ignacio Tartavull, Kisuk Lee, and H Sebastian Seung. An error detection and correction framework for connectomics. *arXiv preprint arXiv:1708.02599*, 2017.