Scalable Biologically-Aware Skeleton Generation for Connectomic Volumes

Brian Matejek, Tim Franzmeyer, Donglai Wei, Xueying Wang, Jinglin Zhao,
Kálmán Palágyi, Jeff W. Lichtman, and Hanspeter Pfister

***Abstract*—As connectomic datasets exceed hundreds of terabytes in size, accurate and efficient skeleton generation of the label volumes has evolved into a critical component of the computation pipeline used for analysis, evaluation, visualization, and error correction. We propose a novel topological thinning strategy that uses biological-constraints to produce accurate centerlines from segmented neuronal volumes while still maintaining biologically relevant properties. Current methods are either agnostic to the underlying biology, have non-linear running times as a function of the number of input voxels, or both. First, we eliminate from the input segmentation biologically-infeasible bubbles, pockets of voxels incorrectly labeled within a neuron, to improve segmentation accuracy, allow for more accurate centerlines, and increase processing speed. Next, a Convolutional Neural Network (CNN) detects cell bodies from the input segmentation, allowing us to anchor our skeletons to the somata. Lastly, a synapse-aware topological thinning approach produces expressive skeletons for each neuron with a nearly one-to-one correspondence between endpoints and synapses. We simultaneously estimate geometric properties of neurite width and geodesic distance between synapse and cell body, improving accuracy by 47.5% and 62.8% over baseline methods. We separate the skeletonization process into a series of computation steps, leveraging data-parallel strategies to increase throughput significantly. We demonstrate our results on over 1250 neurons and neuron fragments from three different species, processing over one million voxels per second per CPU with linear scalability.**

***Index Terms*—Skeleton generation, connectomics, biologically-constrained algorithms.**

I. Introduction

Increased throughput of electron microscopy imaging techniques [1] has enabled nanometer resolution image volumes of brain tissue exceeding terabytes [1], [2] and even petabytes [3], [4] in size. Since manual reconstruction and synapse annotation is infeasible at this scale, researchers employ automatic techniques to segment the volumes into individual neurons [5], [6] and identify synapses [7]. 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 [8], [9]. In turn, researchers hope to improve artificial neural networks [10], expand knowledge of neurological diseases [11], and better understand the brain's underlying computational mechanisms [9].

Manuscript received November 6, 2020; revised November 9, 2021 and January 11, 2022; accepted March 24, 2022. This work was supported in part by NSF grant IIS-1607800.

B. Matejek and T. Franzmeyer contributed equally to this work.

B. Matejek, D. Wei, and H. Pfister are with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA (email: bmatejek@seas.harvard.edu).

T. Franzmeyer is with the Department of Mechanical and Process Engineering at ETH Zurich, Zurich, Switzerland.

X. Wang, J. Zhao, and J. Lichtman are with the Center for Brain Science, Deperatment of Mollecular and Cellular Biology, Harvard University, Cambridge, MA, USA.

Kálmán Palágyi is with the Department of Image Processing and Computer Graphics, University of Szeged, Szeged, Hungary.

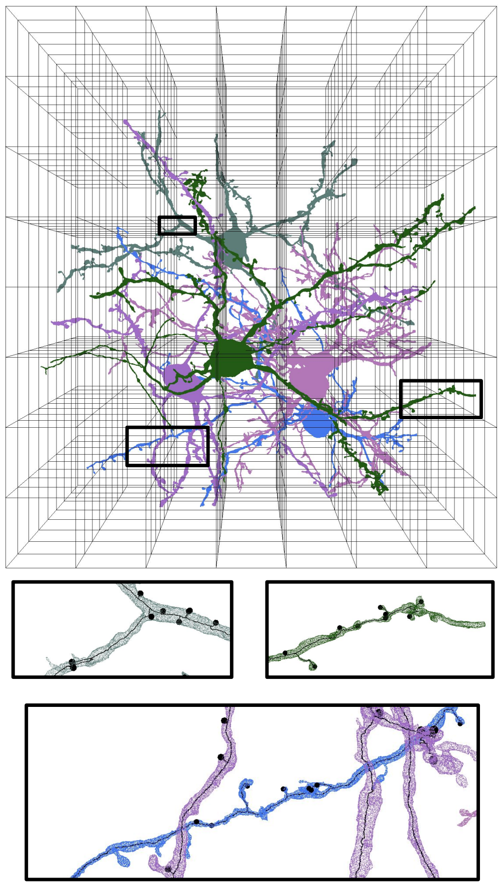

Fig. 1. Neural reconstructions of electron microscopy image stacks easily exceed billions of voxels in size. Here, we show five different neurons that span 270 billion voxels. Our block-based synapse-aware skeleton generation strategy produces expressive skeletons efficiently. We enlarge three regions to show our generated skeletons that connect all synapses, represented by black spheres, to the somata.

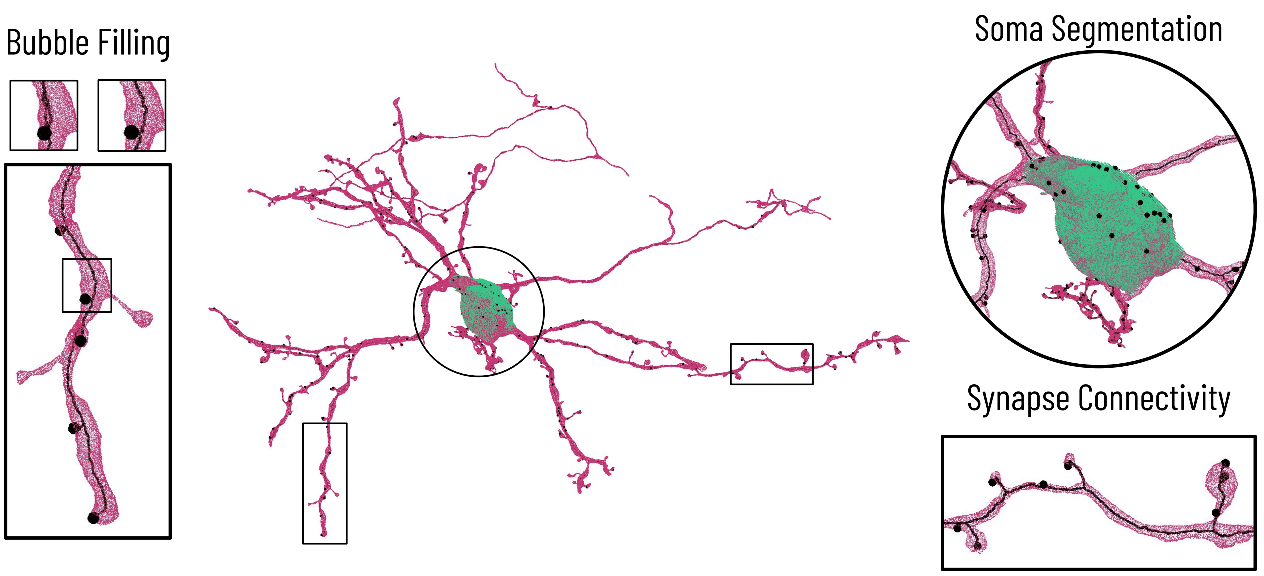

Fig. 2. Our block-based, topological thinning strategy for generating expressive skeletons leverages domain-specific knowledge of the underlying biology. Automatic neural reconstruction techniques often produce millions of bubbles in the output segmentation. Uncorrected, these bubbles cause the skeletons 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 [6], [12], segmentation evaluation [5], visualization [13], and error correction [14], [15]. Most of this research uses a variant of the Tree-structure Extraction Algorithm for Accurate and Robust Skeletons (TEASER) [16], [17], although some work utilizes topological thinning strategies from the volume processing literature [18]–[20]. 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 and proofread brain samples have approached and even exceeded a cubic millimeter [4], [6], [21], [22], 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 [6], [12]. Weighted edges may indicate perceived synaptic strength, although these methods typically only consider the number of synapses between two neurons when assigning edge weights. Researchers have used these graphical models of the brain to find motifs [23], i.e., frequently occurring subgraphs with biological importance, and simulate simple motor responses to an external stimulus [24]. However, 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 [25]. As an extreme example, two neurons A and B may have $3-4\times$ as many synaptic connections as neurons A and C, but C could have a stronger connection to A depending on the synapse locations. These powerful subtleties are currently missing from the wiring diagram models that merely count the number of synaptic connections between two neurons. We can significantly increase the fidelity of the extracted wiring diagrams by considering 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.

Building on our previous work [26], we propose a block-based, topological thinning approach for generating exact skeletons of reconstructed neuronal volumes. As opposed to existing methods [16]–[20], our approach enforces various biological-constraints on the skeletons and generates relevant geometric information useful for higher-level analysis (Fig. 1). Current state-of-the-art reconstruction strategies such as flood-filling networks [5] 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

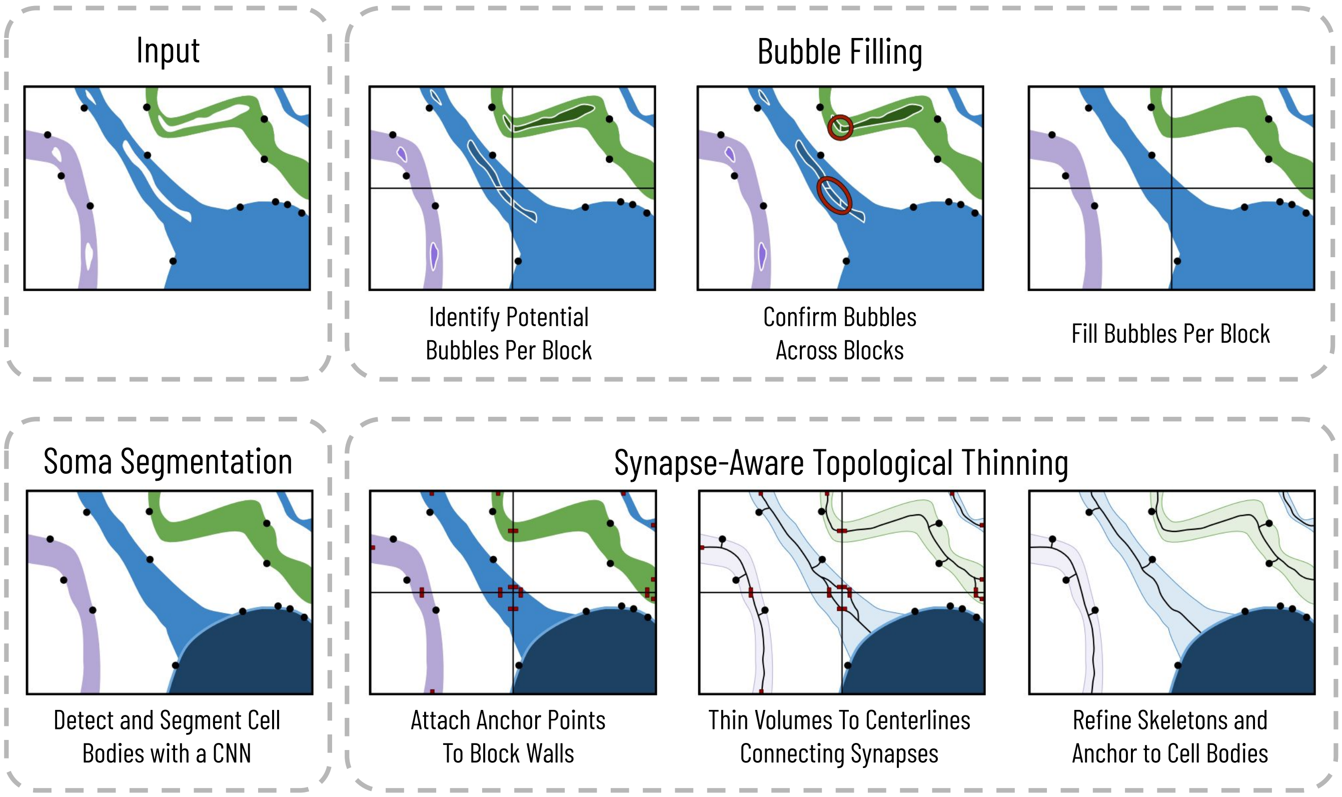
Input
Bubble Filling
Identify Potential Bubbles Per Block
Confirm Bubbles Across Blocks
Fill Bubbles Per Block
Soma Segmentation
Detect and Segment Cell Bodies with a CNN
Synapse-Aware Topological Thinning
Attach Anchor Points To Block Walls
Thin Volumes To Centerlines Connecting Synapses
Refine Skeletons and Anchor to Cell Bodies

Fig. 3. Our method contains three distinct steps to extract biologically-aware skeletons from an input label volume: bubble filling (Sec. III-A), soma segmentation (Sec. III-B), and synapse-aware topological thinning (Sec. III-C). 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. Perpendicular lines in an image panel indicate block-based operations for that computation 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 in the global scope to attach each neurite's skeleton to the cell body. Our previous work focused exclusively on the synapse-aware topological thinning components (bottom right box) and did not contain any parallel processing [26].

surface voxels during thinning (Fig 2, inset, left). We train a Convolutional Neural Network (CNN) with the familiar U-Net architecture [27], [28] to segment 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 (Fig. 2, inset, top right). The voltage transmitted to a cell body from an activated neighbor depends on the distance the signal travels from the synapse to the surface of the cell body. Finally, we devise a topological thinning strategy that produces a nearly one-to-one correspondence between synapses and skeleton endpoints (Fig. 2, inset, bottom right). We produce accurate width estimates and geodesic distances to the cell body for each synapse during the thinning process.

We propose three significant extensions on our previous work [26] that both improve biological fidelity and computational efficiency. First, we introduce a bubble filling process to remove the biologically-infeasible voids sometimes produced by automated segmentation methods. Second, we segment the somata and mask them from the topological thinning process to anchor the axon and dendrite skeletons onto the cell body. These two steps reduce the total computation time by a factor of $10.58\times$ on a connectome volume that spans over two hundred billion voxels. Lastly, we now divide each step of the skeleton generation process into a series 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 over one million voxels per second per CPU.

II. Related Works

An increasing amount of connectomics literature focuses on the use of skeletons for analysis [29], segmentation evaluation [5], [30], visualization [6], [13], and error correction [14], [15]. Much of this research uses TEASER [16], 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 [17]. 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 generally widely applicable in the medical image community with applications in extracting graphs from blood vessels [31], among others [32].

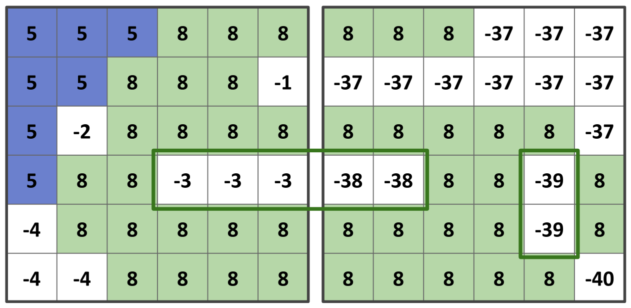

Fig. 4. We illustrate our block-based bubble-filling algorithm on a 2D example. Using a two-pass connected component labeling algorithm, we cluster all "background" voxels (i.e., those that do not belong to a neuron) into components. We label these components with a unique negative integer. Since a bubble can span across multiple block boundaries (labels $-\mathbf{3}$ and $-\mathbf{38}$), 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 $-\mathbf{2}$) or leave the volume (labels $-\mathbf{4}$, $-\mathbf{1}$, $-\mathbf{37}$, $-\mathbf{40}$) remain background.

In volume processing, topological thinning approaches remove the need for parameters by relying on mathematical notions of curve-endpoints [19], [33] or curve-isthmuses [20] 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 skeletons [34], 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 [35]. Generally, these methods either remove voxels sequentially [20], [36] by continually verifying the topological correctness before each new deletion or in parallel [19], [37], [38], where sets of independent voxels are processed separately. However, these thinning strategies cannot efficiently process 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.

III. Methodology

Our method contains three main components to enforce specific biological properties and improve the accuracy of the geometric attributes of our generated skeletons. 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 (Fig. 3, top left box). We first fill bubbles in the input segmentation to correct errors that are common during automatic segmentation (Fig. 3, 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 (Fig. 3, bottom left box). Finally, we thin the neurons using a block-based, synapse-aware strategy that connects all synapses to the cell body (Fig. 3, bottom right box). We also divide the topological thinning process into two data-parallel tasks followed by a global recombination step. In Fig. 3, the image panels with perpendicular lines represent block-based computation steps that are parallelizable. By distributing the most computationally expensive operations over a large number of CPUs, we can quickly generate skeletons on terabyte datasets. Our previous work focused exclusively on the synapse-aware topological thinning component of the pipeline (Fig. 3, bottom right box), with no parallel processing [26].

A. Bubble Filling

Many current state-of-the-art segmentation algorithms [5], [39] 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 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 (Fig. 6), 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 (Fig. 3 top right box). Since these bubbles in the segmentation can span over multiple blocks, the second step requires global scope. Fig. 4 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 [40], [41]. In the figure, there are four identified background components in each block. Each background component receives a globally unique, decreasing, 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 (Fig. 4, label: -2). Similarly, any background component that leaves the volume (Fig. 4, labels: $-4, -37, -40$) 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 (Fig. 4, labels: $-3, -38, -39$). With this
]

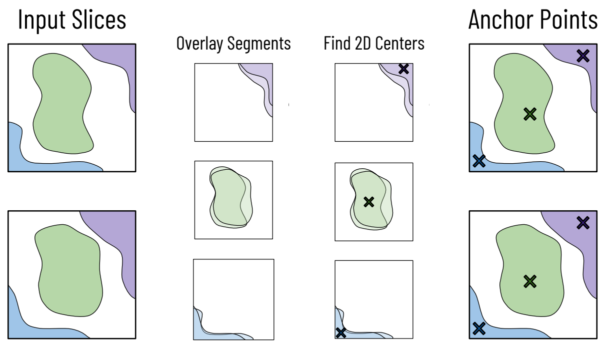

Fig. 5. We compute anchor points on each block surface to guarantee that the skeletons generated in each block connect across block boundaries. For each pair of adjacent block surfaces, we overlay all labels. We then identify central points using a 2D shrinking strategy for each labeled component on the overlayed block surfaces. These points are non-deletable during the forthcoming 3D thinning to guarantee connectivity across block boundaries.

mapping, we fill the bubbles for each block in parallel across a distributed system. This algorithm removes all bubbles, defined as a 6-connected background component within a 26-connected object, from the input segmentation. Note that this process does not remove tunnels, i.e., 6-connected background components entering and leaving the 26-connected object.

B. Soma Segegmentation

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 the somata's interior points. On two representative datasets, the somata account for approximately 65% of all labeled voxels (Sec. V-B.2). Since somata have various shapes and sizes, geometry-based segmentation algorithms do not adequately identify them.

We train a fully convolutional neural (CNN) based on a slight modification of the U-Net [27] to identify cell bodies in the input volume. We first downsample the label volume by a constant factor in x, y, and z. We extract a nine-channel tensor for each query xy image tile where each channel corresponds to the four nearest xy tiles in both directions. We find that using a nine-channel tensor instead of the 3-D U-Net [28] 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.

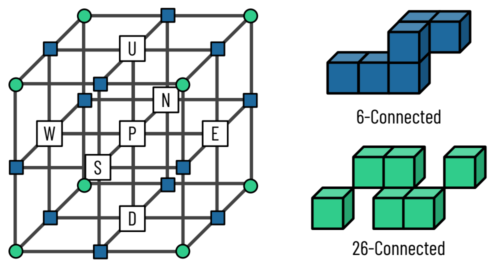

Fig. 6. Topological thinning algorithms rely only on the local neighborhood around a point $\boldsymbol{p}$ to determine deletion (left). $\boldsymbol{N_6(p)}$ contains the point $\boldsymbol{p}$ and the six points labeled **U**, **D**, **N**, **S**, **E**, and **W**. $\boldsymbol{N_{18}(p)}$ includes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

C. Synapse-Aware Topological Thinning

We divide the synapse-aware topological thinning strategy into three distinct components (Fig. 3, 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.

1) 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 (Fig. 5).

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 (Fig. 5, 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 [42] (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 (Fig. 3), we eliminate these loops at a later step (Sec. III-C.3).

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

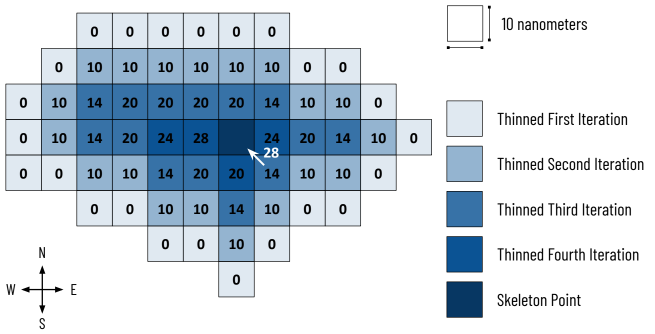

Fig. 7. We generate estimates for neurite width during thinning. Initially, surface voxels have a distance-to-surface value of zero and all interior voxels ∞. When we remove a surface voxel, we update the distance-to-surface value of neighboring voxels if there is a new shortest path to the surface through the removed voxel. In this example, a single skeleton point with a distance of 28 nanometers remains after four thinning iterations. The neurite width is twice the distance-to-surface value.

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 $N_{26}(p)$ (Fig. 6). $N_6(p)$ 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 [35] 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, successive deletion of simple points can reduce an object without any bubbles and tunnels, such as a neuron, to a single point, with no further deletion restrictions. 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 [43] to defining another class of geometric constraints as non-simple curve-isthmuses [20]. 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 in Sec. III-C.1, 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. At this stage, all skeleton endpoints are anchor points, synapse points, or points on the soma surface. 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 [20] 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 (Fig. 6, 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. Fig. 7 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 Fig. 7, 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 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

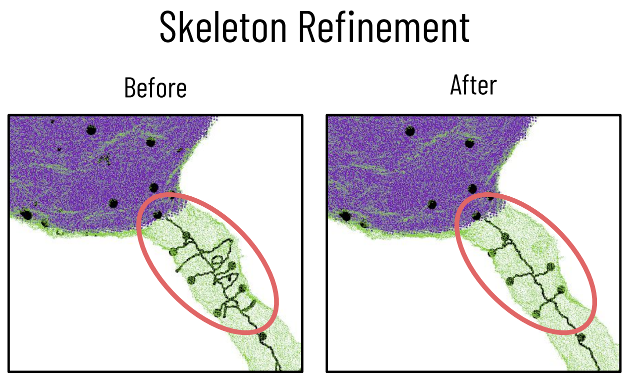

Fig. 8. Our skeleton refinement step removes any loops in the skeleton caused by tunnels in the segmentation (circled). This step also calculates the geodesic distances between every synapse (black spheres) and the cell body (purple volume).

estimates reflect the distance from a central point to the surface more accurately. Fig. 7 shows the skeleton point's final update, which occurs when we delete its southeastern neighbor.

3) 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 (Fig. 3). Since neuronal processes are acyclic, these skeleton loops are 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 (Fig. 8). This process eliminates all loops since the set of 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.

IV. Experiments

A. Datasets and Experimental Set Up

We evaluate our methods on three large-scale connectomic datasets from three different species: rat, fruit fly, and zebra finch (Table I).[1]. Neuroscientists manually segmented and identified the synapses in JWR (rat), the smallest dataset with 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 [5] and identified synapses [7] in the largest dataset, J0126 (zebra finch), that spans over 165 billion voxels.

We compare our proposed method against two baselines: TEASER [16] and an isthmus-based topological thinning approach [20]. We use the `Kimimaro` implementation of the TEASER algorithm from the Seung lab with the default parameters [2]. Isthmus (and other topological) thinning methods are particularly susceptible to surface noise, generating many spurious endpoints [15], [26]. Therefore, we downsample all three datasets for this baseline to a resolution of $100 \times 100 \times 100\,\mathrm{nm}^3$. Additionally, this downsampling 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 since these volume processing solutions require one to read all points into memory. Although we downsample these datasets to a resolution of $100 \times 100 \times 100\,\mathrm{nm}^3$, neurons in more sophisticated species have larger spans and would require more aggressive downsampling. However, reducing the resolution even further would eliminate some of the finer morphological details of neurons [47]. Therefore, most centerline extraction methods will not scale to the next generation of connectomes that come from larger mammals [48].

We ran all timing experiments for our ablation studies on an Intel Core i7-6800K CPU 3.40 GHz with 64GB of RAM. All code is Python and C++ with Cython wrappers and is freely available[3]. We also include in these repositories the weights for our soma segmentation model. We ran all additional timing analysis on a cluster of 18 Intel Xeon Platinum 8268 CPUs running at 2.90GHz with 188GB of RAM.

B. 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 $\epsilon = 10^{-07}$. 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 III, 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.

C. Evaluation Metrics

We evaluate our results using three metrics. First, we use the Neural Reconstruction Integrity (NRI) [49] 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. For us, this metric illustrates the correspondence between synapses and endpoints. For our baselines, we link synapses

[1]We provide links to each dataset that we evaluated on at the following URL: https://www.rhoana.org/blockbased_synapseaware

[2]https://github.com/seung-lab/kimimaro

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

TABLE I

We evaluated our method on three datasets from three 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 [44]	Rat	$106 \times 106 \times 93\,\mu m^3$	$32 \times 64 \times 30\,nm^3$	85	50,334
FIB-25 [45]	Fruit Fly	$36 \times 29 \times 69\,\mu m^3$	$10 \times 10 \times 10\,nm^3$	763	84,157
J0126 [46]	Zebra Finch	$96 \times 98 \times 114\,\mu m^3$	$18 \times 18 \times 20\,nm^3$	407	91,465

TABLE II

We outperform the baseline methods on the NRI score on all three metrics by significant margins. We improve 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 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

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 [16], [17]. We thus 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 average number of skeleton points per neuron. All other things equal, we prefer fewer skeleton points [20] since, among other reasons, this can significantly reduce the computational costs for algorithms that use the skeletons [14], [15].

V. Results

A. Benchmark Comparison

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

1) 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. Intuitively, we expect our method to far exceed the existing state-of-the-art on this metric. We designed our algorithm to preserve the intracellular pathways between synapses, the attribute that NRI scores evaluate. Previous works did not prioritize this preservation, and therefore have significantly lower NRI scores.

2) Width Estimation: We achieve a mean absolute error of our width (i.e., twice the distance-to-surface) estimation of $43.03\,\mathrm{nm}$, $14.42\,\mathrm{nm}$, and $25.55\,\mathrm{nm}$ on the JWR, FIB-25, and J0126 datasets, respectively. The TEASER algorithm outputs a radius for each point, roughly corresponding to the largest sphere wholly contained in the volume centered at that point. The mean absolute error for TEASER's neurite width estimate is $120.69\,\mathrm{nm}$, $19.78\,\mathrm{nm}$, and $171.33\,\mathrm{nm}$ for the three datasets, factors of 2.17, 1.37, and 6.71 worse, respectively. The bubbles in the input segmentation cause TEASER to have less accurate width estimates. As a first step, the TEASER algorithm generates a distance boundary from every voxel in the neuron to the closest background point. With many bubbles in the input segmentation, many of the interior voxels have distance estimates significantly less than the truth since there is a nearby "false" background voxel in a bubble nearby. The isthmus thinning baseline does not produce width estimates.

3) 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 Sec. III-C.3 would significantly simplify these skeletons.

4) 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 $20\,321.96\,\mathrm{nm}$ and $17\,238.31\,\mathrm{nm}$ per synapse.

5) 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 $8.43\,\mathrm{h}$, $38.49\,\mathrm{h}$, and $62.88\,\mathrm{h}$ on the JWR, FIB-25, and J0126 datasets. TEASER took $14.38\,\mathrm{h}$, $35.24\,\mathrm{h}$, and $96.63\,\mathrm{h}$ on the JWR, FIB-25, and J0126 datasets. Our method significantly outperforms TEASER on the JWR and J0126 datasets ($1.71\times$ and $1.54\times$ 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 ($1.09\times$ quicker), which contains no cell bodies.

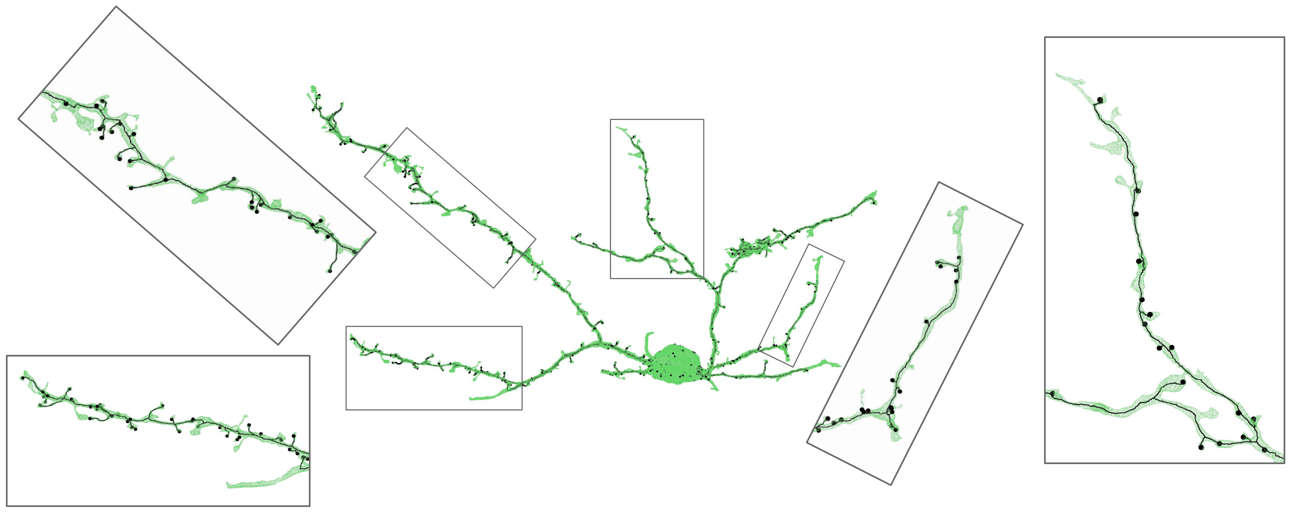

Fig. 9. Our biologically-aware skeleton generation strategy produces centerlines anchored to the cell bodies that maintain connections between all synapses. Here, we show one neuron from the JWR dataset and the generated skeleton. The black spheres represent synapses. We see that the skeleton contains all synapses, leading to the high NRI score, which measures the correspondence between skeleton endpoints and synapses.

6) Qualitative Results: Fig. 9 shows a skeleton generated for a complete neuron from the J0126 dataset. The black spheres indicate synapse locations. Our skeleton refinement process anchors the skeletons to the cell body and removes any self-loops in the skeletons caused by tunnels through the neuron surface.

B. Ablation Studies

Here, we discuss the improvements in computational complexity, width estimation, and the number of skeleton points when adding bubble filling and soma segmentation. The NRI score minimally varies when removing certain parts of the pipeline since the thinning algorithm, regardless of input segmentation (with bubbles, with soma, with neither, etc.), will create a near-perfect correspondence between synapses and endpoints.

1) 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 [5] 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 described in Sec. III-C.2 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 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 III). On the other hand, filling the bubbles in the J0126 volume decreases the total run time by 42.56%, when we also mask the cell bodies (Table III). 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.

It is difficult to quantify the effect that bubbles in the volume have on the width estimates since the bubbles cause the "centerlines" to divert from the actual middle of the neurite (Fig. 2, inset, top left). Since the thinning algorithm preserves all intracellular pathways between synapses, the same neurite branches will have centerlines regardless of bubbles in the segmentation. Therefore, we approximate the error introduced by bubble filling by considering the average skeleton point width along all neurites (i.e., excluding any skeleton points in the cell bodies). We find that the average width after bubble filling is 0.63%, 2.09%, and 26.31% greater than without that step on the FIB-25, JWR, and J0126 datasets, respectively. Again, we see a more pronounced effect on the J0126 dataset, which has a higher incidence of bubbles.

Bubble filling has a negligible effect on the number of skeleton points on the FIB-25 and JWR datasets, with a difference of less than 1.5%. However, for J0126, removing bubbles before masking the cell bodies and running topological thinning reduced the number of skeletons points by 18.82%. We expect this larger difference since the J0126 dataset contains millions of more bubbles at a higher fill rate than the other two datasets.

2) 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

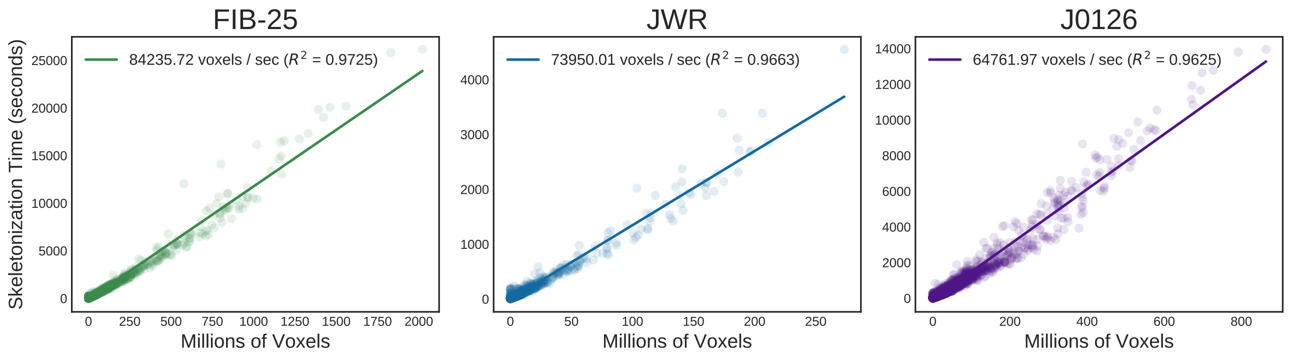

Fig. 10. Our block-based topological thinning strategy runs linearly with the number of non-zero voxels in the volume. Here, we show the timing results for the three datasets on various block sizes.

TABLE III

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 $10.58\times$.

Bubble Filling	Soma Segmentation	JWR	FIB-25	J0126
✓	✓	5.19 h	N/A	45.29 h
	✓	4.03 h	N/A	78.85 h
✓		10.37 h	33.54 h	113.82 h
		20.56 h	30.72 h	479.19 h

cell bodies, after filling all bubbles (Table III). These cell bodies contain many voxels (64.46% of the total volume, on average) and are not structurally interesting for skeletonization purposes. Removing the cell bodies from the segmentations before topological thinning significantly reduces the number of points in the skeletons. On the JWR and J0126 datasets, masking the somata reduced the number of skeleton points by 68.42% and 26.87%, respectively. Since the width of a skeleton point within a cell body is undefined and not particularly biologically relevant, we do not consider how soma segmentation affects the average width of skeleton points in the volume.

3) Synapse-Aware Topological Thinning: Fig. 10 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 belonging to a neuron in a given block, not the total number of voxels in a block. Typically, 85-95% of each dataset's voxels remain unlabeled 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 III). The CPU time required to extract centerlines from the J0126 dataset plummeted from 479.19 h to 45.29 h with the addition of bubble filling and soma segmentation. Furthermore, the heaviest computational parts of the algorithm can run in parallel.

VI. 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. Although we focus our methods on the giga- and tera-voxel datasets in the connectomics literature, our methods also extend to the general medical imaging community. Researchers have used skeletonized representations of blood vessels previously to extract graphs and determine the health of specific organs, such as the liver [31]. By adding geometric attributes such as blood vessel width, we could simulate blood flow in these organs and detect potential medical issues in a patient.

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 $10.58\times$ speed up on one of our datasets over our previous work [26]. Furthermore, we designed our solution so that the computation-heavy tasks run in parallel on a distributed system, enabling us to achieve a throughput of over one million voxels per CPU. 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 number of blocks. Our scalable method can extract biologically-aware skeletons from massive connectomic volumes by distributing computation across an array of CPUs.

Acknowledgment

We thank Jörgen Kornfeld and Winfried Denk's group for the J0126 data and synapses, and the Connectomics Group at Google led by Viren Jain for the segmentation. For the JWR dataset, we thank Jeff Lichtman's group at Harvard University for image acquisition, alignment, and ground truth labeling. We thank Gábor Németh at the Department of Image Processing and Computer Graphics at the University of Szeged for code concerning the 2D shrinking algorithm.

REFERENCES

[1] A. Suissa-Peleg *et al.*, “Automatic neural reconstruction from petavoxel of electron microscopy data,” *Microscopy and Microanalysis*, vol. 22, no. S3, pp. 536–537, 2016.

[2] Z. Zheng *et al.*, “A complete electron microscopy volume of the brain of adult drosophila melanogaster,” *Cell*, vol. 174, no. 3, pp. 730–743, 2018.

[3] W. Yin *et al.*, “A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy,” *bioRxiv*, p. 791889, 2019.

[4] S. Dorkenwald *et al.*, “Binary and analog variation of synapses between cortical pyramidal neurons,” *bioRxiv*, 2019.

[5] M. Januszewski *et al.*, “High-precision automated reconstruction of neurons with flood-filling networks,” *Nature methods*, vol. 15, no. 8, pp. 605–610, 2018.

[6] C. S. Xu *et al.*, “A connectome of the adult drosophila central brain,” *bioRxiv*, 2020.

[7] S. Dorkenwald *et al.*, “Automated synaptic connectivity inference for volume electron microscopy,” *Nature methods*, vol. 14, no. 4, p. 435, 2017.

[8] O. Sporns, G. Tononi, and R. Kötter, “The human connectome: a structural description of the human brain,” *PLoS computational biology*, vol. 1, no. 4, 2005.

[9] J. W. Lichtman and W. Denk, “The big and the small: challenges of imaging the brain’s circuits,” *Science*, vol. 334, no. 6056, pp. 618–623, 2011.

[10] M. Helmstaedter, “The mutual inspirations of machine learning and neuroscience,” *Neuron*, vol. 86, no. 1, pp. 25–28, 2015.

[11] A. Fornito, A. Zalesky, and M. Breakspear, “The connectomics of brain disorders,” *Nature Reviews Neuroscience*, vol. 16, no. 3, pp. 159–172, 2015.

[12] ——, “Graph analysis of the human connectome: promise, progress, and pitfalls,” *Neuroimage*, vol. 80, pp. 426–444, 2013.

[13] H. Mohammed *et al.*, “Abstractocyte: A visual tool for exploring nanoscale astroglial cells,” *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 853–861, 2017.

[14] K. Dmitriev, T. Parag, B. Matejek, A. E. Kaufman, and H. Pfister, “Efficient correction for em connectomics with skeletal representation,” *British Machine Vision Conferemce (BMVC)*, 2018.

[15] B. Matejek, D. Haehn, H. Zhu, D. Wei, T. Parag, and H. Pfister, “Biologically-constrained graphs for global connectomics reconstruction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2089–2098.

[16] M. Sato, I. Bitter, M. A. Bender, A. E. Kaufman, and M. Nakajima, “Teasar: Tree-structure extraction algorithm for accurate and robust skeletons,” in *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*. IEEE, 2000, pp. 281–449.

[17] T. Zhao, D. J. Olbris, Y. Yu, and S. M. Plaza, “Neutu: software for collaborative, large-scale, segmentation-based connectome reconstruction,” *Frontiers in neural circuits*, vol. 12, p. 101, 2018.

[18] W. Gong and G. Bertrand, “A simple parallel 3d thinning algorithm,” in *[1990] Proceedings. 10th International Conference on Pattern Recognition*, vol. 1. IEEE, 1990, pp. 188–190.

[19] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, “Building skeleton models via 3-d medial surface axis thinning algorithms,” *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[20] K. Palágyi, “A sequential 3d curve-thinning algorithm based on isthmuses,” in *International Symposium on Visual Computing*. Springer, 2014, pp. 406–415.

[21] J. W. Lichtman, H. Pfister, and N. Shavit, “The big data challenges of connectomics,” *Nature neuroscience*, vol. 17, no. 11, pp. 1448–1454, 2014.

[22] C. M. Schneider-Mizell *et al.*, “Chandelier cell anatomy and function reveal a variably distributed but common signal,” *bioRxiv*, 2020.

[23] L. K. Scheffer *et al.*, “A connectome and analysis of the adult drosophila central brain,” *Elife*, vol. 9, p. e57443, 2020.

[24] T. Jovanic *et al.*, “Competitive disinhibition mediates behavioral choice and sequences in drosophila,” *Cell*, vol. 167, no. 3, pp. 858–870, 2016.

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

[26] B. Matejek, D. Wei, X. Wang, J. Zhao, K. Palágyi, and H. Pfister, “Synapse-aware skeleton generation for neural circuits,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 227–235.

[27] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[28] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d u-net: learning dense volumetric segmentation from sparse annotation,” in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.

[29] A. Talwar *et al.*, “A topological nomenclature for 3d shape analysis in connectomics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 986–987.

[30] S. M. Plaza and J. Funke, “Analyzing image segmentation for connectomics,” *Frontiers in neural circuits*, vol. 12, p. 102, 2018.

[31] Y. Chen, C. O. Laura, and K. Drechsler, “Generation of a graph representation from three-dimensional skeletons of the liver vasculature,” in *2009 2nd International Conference on Biomedical Engineering and Informatics*. IEEE, 2009, pp. 1–5.

[32] D. Jin and P. K. Saha, “A new fuzzy skeletonization algorithm and its applications to medical imaging,” in *International Conference on Image Analysis and Processing*. Springer, 2013, pp. 662–671.

[33] P. K. Saha, G. Borgefors, and G. S. di Baja, *Skeletonization: Theory, methods and applications*. Academic Press, 2017.

[34] D. Jin, C. Chen, E. A. Hoffman, and P. K. Saha, “Curve skeletonization using minimum-cost path,” in *Skeletonization*. Elsevier, 2017, pp. 151–180.

[35] G. Malandain and G. Bertrand, “Fast characterization of 3d simple points,” in *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*. IEEE, 1992, pp. 232–235.

[36] K. Palágyi *et al.*, “A sequential 3d thinning algorithm and its medical applications,” in *Biennial International Conference on Information Processing in Medical Imaging*. Springer, 2001, pp. 409–415.

[37] K. Palágyi and A. Kuba, “A 3d 6-subiteration thinning algorithm for extracting medial lines,” *Pattern Recognition Letters*, vol. 19, no. 7, pp. 613–627, 1998.

[38] W. Xie, R. P. Thompson, and R. Perucchio, “A topology-preserving parallel 3d thinning algorithm for extracting the curve skeleton,” *Pattern Recognition*, vol. 36, no. 7, pp. 1529–1544, 2003.

[39] J. Funke, F. D. Tschopp, W. Grisaitis, C. Singh, S. Saalfeld, and S. C. Turaga, “A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs,” *arXiv preprint arXiv:1709.02974*, 2017.

[40] A. Rosenfeld and J. L. Pfaltz, “Sequential operations in digital picture processing,” *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 471–494, 1966.

[41] K. Wu, E. Otoo, and K. Suzuki, “Two strategies to speed up connected component labeling algorithms,” Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US), Tech. Rep., 2005.

[42] G. Németh, P. Kardos, and K. Palágyi, “2d parallel thinning and shrinking based on sufficient conditions for topology preservation,” *Acta Cybernetica*, vol. 20, no. 1, pp. 125–144, Jan. 2011.

[43] G. Bertrand and Z. Aktouf, “Three-dimensional thinning algorithm using subfields,” in *Vision Geometry III*, vol. 2356. International Society for Optics and Photonics, 1995, pp. 113–125.

[44] Z. Lin *et al.*, “Two stream active query suggestion for active learning in connectomics,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 103–120.

[45] S.-y. Takemura *et al.*, “Synaptic circuits and their variations within different columns in the visual system of drosophila,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 44, pp. 13 711–13 716, 2015.

[46] J. Kornfeld *et al.*, “Em connectomics reveals axonal target variation in a sequence-generating network,” *Elife*, vol. 6, p. e24364, 2017.

[47] D. Rolnick *et al.*, “Morphological error detection in 3d segmentations,” *arXiv preprint arXiv:1705.10882*, 2017.

[48] A. Shapson-Coe *et al.*, “A connectomic study of a petascale fragment of human cerebral cortex,” *bioRxiv*, 2021.

[49] E. P. Reilly *et al.*, “Neural reconstruction integrity: A metric for assessing the connectivity accuracy of reconstructed neural networks,” *Frontiers in Neuroinformatics*, vol. 12, 2018.