Topological Skeletonization and Tree-Summarization of Neurons Using Discrete Morse Theory

Suyi Wang* Xu Li† Partha Mitra† Yusu Wang*

Abstract

Neuroscientific data analysis has classically involved methods for statistical signal and image processing, drawing on linear algebra and stochastic process theory. However, digitized neuroanatomical data sets containing labelled neurons, either individually or in groups labelled by tracer injections, do not fully fit into this classical framework. The tree-like shapes of neurons cannot mathematically be adequately described as points in a vector space (eg, the subtraction of two neuronal shapes is not a meaningful operation). There is therefore a need for new approaches. Methods from computational topology and geometry are naturally suited to the analysis of neuronal shapes. Here we introduce methods from Discrete Morse Theory to extract tree-skeletons of individual neurons from volumetric brain image data, or to summarize collections of neurons labelled by localized anterograde tracer injections. Since individual neurons are topologically trees, it is sensible to summarize the collection of neurons labelled by a localized anterograde tracer injection using a consensus tree-shape. This consensus tree provides a richer information summary than the regional or voxel-based "connectivity matrix" approach that has previously been used in the literature.

The algorithmic procedure includes an initial pre-processing step to extract a density field from the raw volumetric image data, followed by initial skeleton extraction from the density field using a discrete version of a 1-(un)stable manifold of the density field. Heuristically, if the density field is regarded as a mountainous landscape, then the 1-(un)stable manifold follows the "mountain ridges" connecting the maxima of the density field. We then simplify this skeleton-graph into a tree using a shortest-path approach and methods derived from persistent homology. The advantage of this approach is that it uses global information about the density field and is therefore robust to local fluctuations and non-uniformly distributed input signals. To be able to handle large data sets, we use a divide-and-conquer approach. The resulting software *DiMorSC* is available on Github[40]. To the best of our knowledge this is currently the only publicly available code for the extraction of the 1-unstable manifold from an arbitrary simplicial complex using the Discrete Morse approach.

1 Introduction

Understanding the neuronal connectivity architecture of brains is an important goal in neuroscience. The primary approach brought to bear on this goal is the reconstruction of neuronal projections following injections of tracers within the brain. With the development of high throughput pipelines and technologies that can deal with large digital data sets, it is now possible to analyze whole brain datasets at a cellular resolution. However, there is a need for developing new methods that can facilitate the modeling and understanding of neuronal morphology and thus aid in extracting connectivity information from brain image volumes. The typical approach is to summarize the

*Computer Science and Engineering Department, The Ohio State University, Columbus, OH 43210.
†Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724

results in the form of regional, or voxel to voxel "connectivity matrices" or directed graphs, with the source region given by the tracer injection site and the target region containing tracer labelled axons or retrogradely labelled neurons. However, this connectivity matrix summary loses all information about the tree-like structure and shape of the projection neurons that constitute the tracer-labelled set. Here we introduce a conceptually distinct summary of an anterograde tracer injection (which labels a collection of neuronal somata concentrated at the injection site, together with the projecting axons and dendrites) in the form of a consensus tree that provides a geometrically and topologically meaningful summary of the collection of neurons labelled by the tracer injection.

The methodology developed here is general and applies also to the idealized case of a single labelled neuron, and therefore provides an additional method for skeletonization individually labelled neurons from volumetric image data sets. A large number methods exist for automatic neuron tracing, mostly for reconstructing single neurons, such as [4, 6, 8, 10, 13, 14, 24, 28, 29, 35, 36, 44, 43, 47, 46, 49, 48, 50, 55, 57, 58, 59, 61, 60]; see also surveys [2, 19, 31] and book [5] for more comprehensive discussions on neuron tracing methods. The advantage of the Discrete Morse based skeletonization for single neurons arises from the usage of global (as opposed to purely local) information, leading to reconstructions that have robustness to local variations and non-uniform signal distributions.

Current work. In this paper, we propose a pipeline to summarize the 3D image stacks based on topological methods. We apply the pipeline to trace neuronal projections following the anterograde injection of AAV tracer. We also provide the details about the accompanying software, DiMorSC. Our pipeline has three stages (see Figure 1). It first performs a pre-processing step and converts the input 3D image stack into a density field. It then extracts the skeleton from the density field using the topological concept of *1-(un)stable manifolds* from discrete Morse theory. Previously, 1-stable manifolds have been successfully applied to extract graph skeletons from 2D/3D data, such as extracting cosmic web from the simulated density of dark matter in $\mathbb{R}^3$ [45] and the reconstruction of road networks from a large collection of GPS trajectories [52]. We adapt this idea for summarizing 3D mesoscopic AAV tracer images. In order to handle the relative large size of image data, we propose a scalable divide-and-conquer strategy to compute this skeleton. Finally, after an initial graph skeleton is extracted, we develop a shortest path based approach to convert the skeleton to a summary tree which is rooted at the injection site. We further provide a simplification strategy, via ideas from the persistent homology from computational topology, to control the level of details of the final summarization. Our proposed method works directly on 3D image volumes. Leveraging the topological structure behind the density field, our method uses the global information from the input images and thus can model and summarize global connectivity paths resulting from tracer injections. It is robust to noise as well as non-uniformly distributed input signals.

The implementation of the algorithm presented here, is based on discrete Morse theory, which extracts the skeleton in a *combinatorial manner* rather than in a numerical manner. Overall, our approach provides a unified framework with a mathematical foundation (based on discrete Morse theory combined with topological persistence) for extracting a tree skeleton from input images. The *combinatorial nature* of our algorithm, combined with the *systematic simplification strategy*, allow us to extract a principal backbone from noisy input images containing complex signals. We compare results obtained from the topological approach to existing methods for skeletonization single neurons and show that it provides comparable or better performance (Section 4.1). We apply these methods to the novel summarization of anterograde tracer injection data from the Mouse Brain Architecture Project to map whole-brain connectivity at a mesoscopic scale. Compared with previous regional connectivity based data summaries, the tree-summary retains geometrical

and topological information about the projection patterns lost in the connectivity matrix summary. This should be also useful in connecting the tracer-injection data with single neuron projection data from the injection sites, as the tree summary provides a consensus tree structure that captures information about the population of neurons with somata localized at the injection site. The resulting software is publicly available at [40].

The remainder of the paper is organized as follows: Below we first briefly discuss some related work. In Section 2 we introduce our pipeline for neuron tracing and describe details of our method. In Section 3, we discuss the divide-and-conquer strategy to handle large data set that cannot fit in the available memory. Finally in section 4, we first demonstrate the effectiveness of our proposed software on single neuron reconstruction that we tested on datasets freely available from the DIADEM challenge [39]. We then show results on summarization of anterograde tracer injetions.

Related work. There is a large literature for single neuron reconstruction e.g [4, 6, 8, 10, 13, 14, 24, 28, 29, 35, 36, 44, 43, 47, 46, 49, 48, 50, 55, 57, 58, 59, 61, 60]. We refer the readers to surveys [2, 19, 31] and a book [5] for more comprehensive discussions on (single) neuron tracing methods, and we only mention a few representative ones below. We note that many of the algorithms are incorporated into the publicly available visualization platform Vaa3D [41].

Most of the neuron tracing algorithms either grow a neuron sequentially (such as growing a tree from the root), or connect (certain special) points in a non-sequential manner. A popular class of sequential tracing algorithms uses a shortest-path based approach, such as APP[37], APP2 [54] and SmartTracing [12]. These methods start with a given seed point and grow it into a tree by connecting new nodes to the existing tree through the shortest paths. For example, in APP2, all pixels with intensity lower than a predefined threshold are treated as background and the foreground pixels are processed by a fast marching method, which computes for every pixel its shortest distance to the background. The distance field of foreground is then used as the new 3D image, and intuitively, the intensity of pixels lying in the middle of a neuron cell is higher than those close to the boundary. After fast marching, APP2 sets a base point and computes the shortest path tree to the base point as an initially reconstructed neuron tree. The tree is then pruned according to the lengths of the branches. In SmartTracing [12], the result is further improved by a machine learning approach that better classifies a pixel as background, foreground or unknown. APP2 is efficient, however, the algorithm could stop at a gap in the signal. The authors resolve this issue by developing a search procedure to explore around the gap. The machine learning based SmartTracing potentially generates more accurate results, but is significantly slower than APP2. For our comparisons, we assume that APP2 and SmartTracing represent the state-of-the-art algorithms for single neuron tracing.

The tracing can also be performed sequentially only at the branch level. For example, in the active contour based approach [53] proposed by Wang et. al., a gradient vector is computed for each pixel in 3D. The authors then define an energy function using the gradient vector as the external force and a smoothness function as the internal force. The method iteratively identifies a foreground point using Frangis vesselness measure [23] and grows it to an arc on the neuron tree while minimizing the energy function. Upon obtaining a maximal arc, all nearby points are removed and this growing procedure is repeated until all foreground points are traced. Compared to the global sequential tracing methods such as APP2, this type of "semi-sequential tracing" (at the branch level) may be more accurate in generating individual branches, but assembling the branches into a single tree can be challenging.

The above approaches usually require the segmentation of the foreground from the background within the image. The quality of the reconstruction is dependent on the quality of this segmenta-

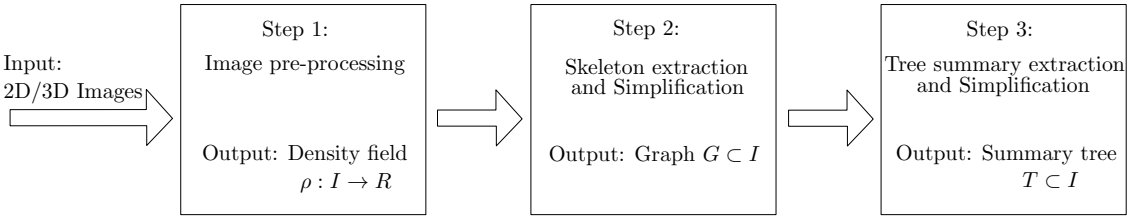


Figure 1: Computational fipeline for our tree-summarization framework of an input image.

tion. For images that are noisy, i.e., where intensity of signals are not always homogeneous, the segmentation quality often is problematic and tends to cause gaps and can even lead to broken branches.

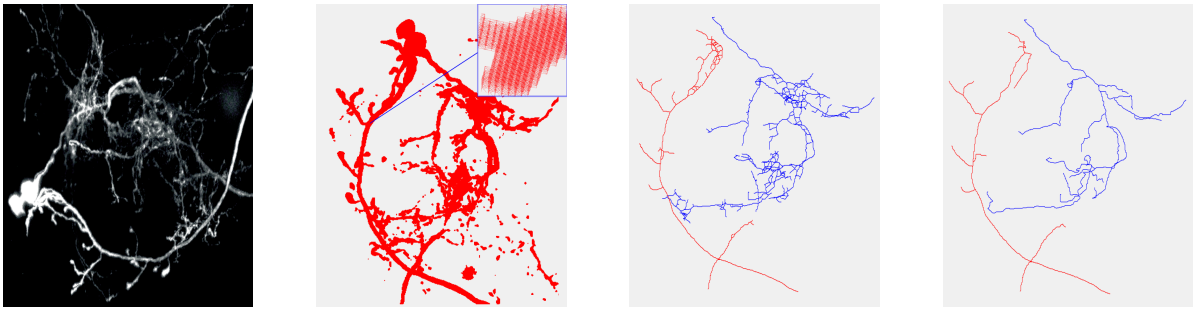
As described in **Our work**, our approach resolves this issue by taking a global view of the entire data, and traces the neuron based on the topological structure behind the given input 3D image data. Our neuron tracing approach builds upon the algorithm developed by Yuan et al. [56], which traces neurons by connecting the critical points in the input. However, there are major differences. (1) In Yuan *et al*, the critical points are connected by the trace induced by moving the saddle points along the gradient using a numerical method, while our approach uses the topological concept of "1-stable manifold" from Morse theory to connect them in a theoretically well-founded manner. Furthermore, we use the *discrete Morse theory* to compute such 1-stable manifolds in a robust combinatorial manner (in contrast to a numerical approach). (2) Yuan]it et al simplifies the initial result using a shortest path tree and further prunes the tree by an erosion scheme, while our approach uses topological persistence simplification in a systematic manner, and less important features are removed first. In summary, our approach provides a unified, conceptually simple, and mathematically sound framework to extract a tree skeleton from input image volumes. The combinatorial nature of our algorithm, combined with the systematic simplification strategy, allows us to extract the main backbone from noisy input images with complex signal content.

From a computer science perspective, our apporach builds upon prior work on using discrete Morse based methodology for extracting skeletons of images; see e.g, [1, 16, 42, 45, 52]. Our software is based on the work of Gyulassy [1, 26], of Sousbie [45], and Wang et al. [52]. Our algorithm focuses on the specific case of extracting graph skeletons from input data, and simplifies the previous approaches for this specific case (e.g., we do not need to handle the cancellation of edge-triangle pairs during the simplification stage). Our implementation can also take an arbitrary simplifical complex as input, while the previous implementation works for 2D / 3D images or Delaunay triangulations (which is a specific type of simplicial complex). This apporach of using an arbitrary simplicial complex input, improves the efficiency in handling large images, as the arbitrary simplicial complex allows us to consider only regions around signal, which can be rather sparse within the input images. Finally, this graph skeleton reconstruction step is combined with a tree-extraction and simplification step to produce the final summary.

2 Method

2.1 Overview

Our pipeline accepts 3D images and outputs a geometric tree in three steps as shown in Figure 1: image processing to convert the input image to a density field $f : I \to \mathbb{R}$ defined on the 3D cube I; extracting graph skeleton G from f, and tree-summary T extraction and simplification from G. An example illustrating the pipeline is given in Figure 2.



(a) (b) (c) (d)

Figure 2: Pipeline overview: the Input image volume in (a) is first converted to the 3D density map shown in (b). (c): initial graph extraction, and (d): tree extraction and simplification (red and blue trees are the final two output trees, reconstructing two neurons); all in 3D.

Step 1: **Image pre-processing.** This step converts the input from 3D image stacks to a density (grey scale) map. The density map $\rho : I \to \mathbb{R}$ is a function defined on the 3D cube $I = [0, 1]^3$. In the discrete case, this domain I is represented by a cubic grid, which is further triangulated and represented by a so-called *simplicial complex* K, consisting of a set of vertices, edges, triangles and tetrahedra. However, as we will see later: (i) we only need the vertices, edges and triangles of K for our graph skeleton and tree-summary extraction; thus from now on, we assume that K consists of only the vertices, edges and triangles from the triangulation of I. (ii) In cases when the input image is large in size, we can restrict K to a sub-complex of it which intuitively captures where signals lie. Given a triangulation K of I, the density map ρ is defined at vertices of K, and in what follows, we sometimes refer to it as the density map $\rho : \mathsf{K} \to \mathbb{R}$. Details of this step are described in Section 2.2.

Step 2: **Graph skeletonization.** This step extracts the graph skeleton of the density map $\rho : \mathsf{K} \to \mathbb{R}$ using the so-called 1-stable manifolds of f, which are computed via the discrete Morse theory. Intuitively, if we view the graph of the density map as a terrain defined on $\mathbb{R}^3 \times \mathbb{R}$, the 1-stable manifolds capture the network of "mountain ridges" of this terrain, representing the "center curves" of the local high density regions (corresponding to signals). This concept and its computation will be described in Section 2.3.

Step 3: **Tree extraction and summarization.** Given the graph skeleton G extracted in Step 2, this step converts it to a tree summary T: This tree can be rooted at a specific choice of root if desired (say the injection site in the input AAV tracer image). We further develop a simplification strategy to simply this tree and control its level of details in a systematic manner, using ideas from topological persistence. Details will be presented in Section 2.4.

2.2 Step 1: Preprocessing

The input is a 3D image volume I consisting of a stack of 2D images. The purpose of this step is to extract a density map from $\rho : I \to \mathbb{R}$, where the value of I is given at discrete grid points (pixels of the input images) and indicates the strength of the signal at this point.

The input image can be of different types. Potentially different image pre-processing techniques will be needed to handle different type of input images. The procedure we describe here was developed for fluorescent image stacks of mouse brains injected with a tracer substance (Adeno-

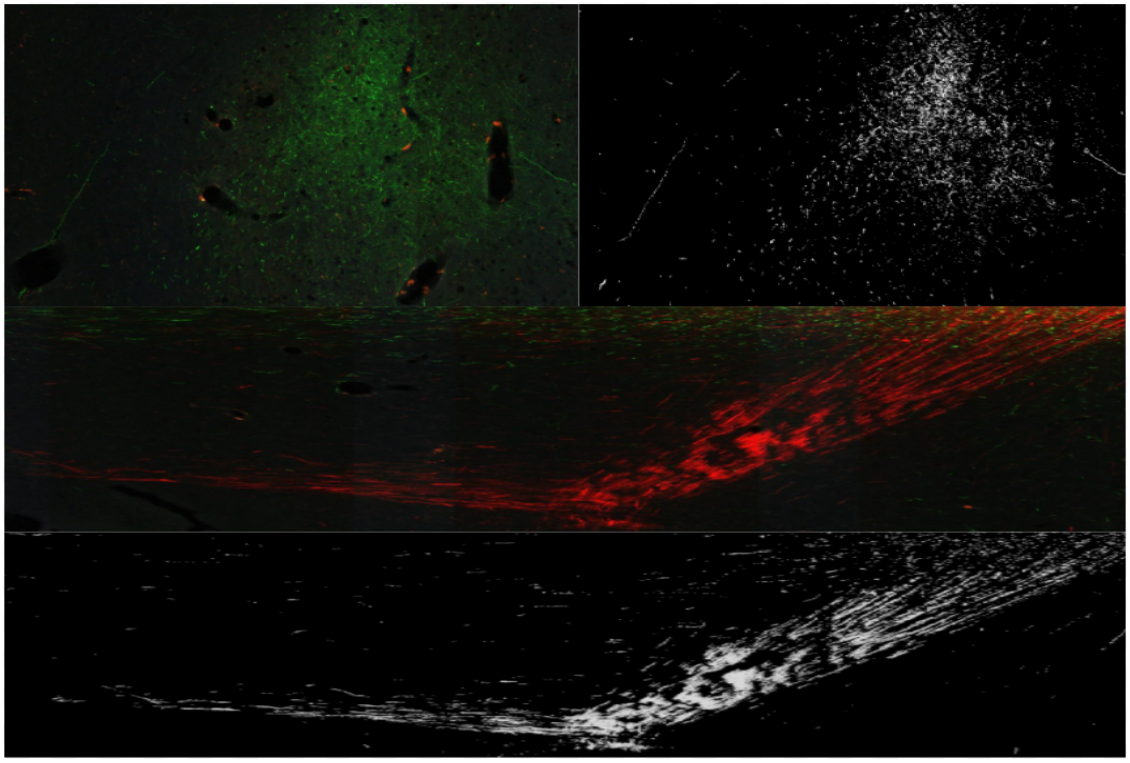


Figure 3: Preprocessing results for green (top two images) and red AAV tracer (bottom two images). The green and red signals result from fluorescent tracers visible in part of a tracer injected mouse brain. The black and white images are resulting density maps.

Associated Viruses carrying a Green or Red Fluorescent Protein payload) scanned on a Whole Slide Imager (a Hamamatsu Nanozoomer). The raw data consists of a 3D stack of RGB images, 12 bits/color channel, in-plane resolution of 0.46μ and section spacing of 40μ. The preprocessing for quantifying the green and red tracer are performed separately. For the green tracer, a Laplacian of Gaussian(LoG) filter is applied first to increase the signal to background ratio and sharpen the tracer signal. Then the image is converted to HSV and LAB colorspace to filter out noisy background. Blood vessel artifacts are detected and removed using Circle Hough transform. The images are registered onto a reference atlas using procedures described elsewhere.

For the red tracer, due to the presence of autofluorescence that is not related to the signal[7] in the brain, a simple k-nn clustering strategy was used to separate tracer signal from autofluorescence noise. In addition, the autofluorescence can be regarded as shot noise in direction perpendicular to the sections, so a median filter is also used. An example of the preprocessing result is demonstrated in Figure 3. We refer to the initial density field after this pre-processing as $\rho_1 : I \to \mathbb{R}$.

For later steps, we assume that our input domain is modeled by a triangulation K (consisting of vertices, edges, and triangles [1]) instead of having cubic cells. We consider the pixels as points on a regular grid and triangulate each cubic cell (the exact triangulation is not important, as long as triangulating neighboring cubes gives rise to consistent triangulation of the common (square) face they share).

Now we have a triangulation K of I with an initial density function $\rho_1 : vert(\mathsf{K}) \to \mathbb{R}$ defined at vertices $vert(\mathsf{K})$ of K (which are the grid points in I). By default, our algorithm performs one

[1]Strictly speaking, the triangulation of the 3D domain also include tetradedral cells. However, for our later algorithm, only the so-called 2-skeleton, which consists of vertices, edges and triangles, is needed and thus computed.

more smoothing step, by smoothing ρ_1 with a Gaussian kernel within a small neighborhood of each point. Let $\rho : vert(\mathsf{K}) \to \mathbb{R}$ be the resulting final density map.

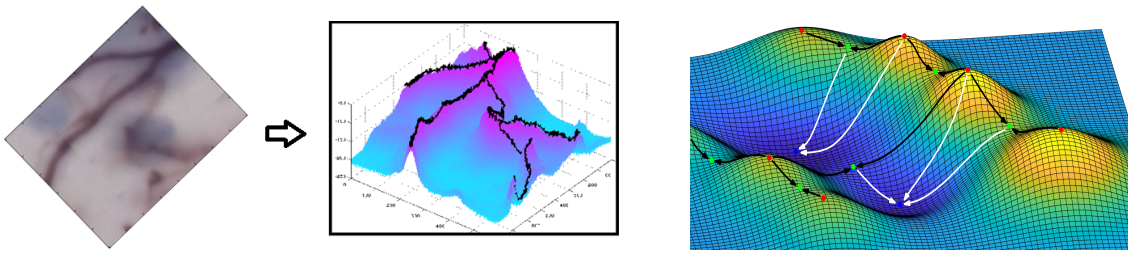
We perform this smoothing stage for the following reason: The initial density map ρ_1 might contain a signal plateau (flat top) area, on which the 1-stable manifold (mountain ridge) in the next step could be ambiguous. For example, in places where the signal is saturated, there could be a thick band of pixels with the same (highest) value, form a plateau (flat mountain peak) in the terrain formed by this function. This gives a degenerate case for defining/tracing "mountain ridges" in our later steps. The Gaussian smoothing help alleviate the situation: it would strengthen the signal in the center of such flat regions, while reduce the intensity of pixels in the boundary region. Furthermore, the preprocessing strategy may segment the input image and output a binary density field where points in the foreground have value 1 and those in the background having value 0. The Gaussian smoothing converts such an input into a smoother field, with points along "centerlines" of the foreground having higher function values, so that the "mountain ridges" of such a terrain can later be captured by our 1-stable manifolds approach.

Finally, we remark that in our pipeline, instead of using the triangulation K of the entire 3D image I, we can also use a subcomplex $\mathsf{K}' \subset \mathsf{K}$ spanned by only pixels whose density value is larger than a threshold to reduce the size of input. Note that if needed, a very low threshold can be used to remove the points that are obviously background, so as not to cause gaps in the remaining subcomplex. Allowing for adequate computing resources, this method would be more reliable on the full image without the removal of any data point.

2.3 Step 2: Graph skeletonization

The input to this step is a density field $\rho : \mathsf{K} \to \mathbb{R}$ where K is a triangulation of $I \subset \mathbb{R}^3$. Below we first explain the main idea for the continuous setting where ρ is assumed to be a smooth function defined on the domain (3D cube) $\rho : I \to \mathbb{R}$.

Formally, given any smooth function $f : \mathbb{R}^3 \to \mathbb{R}$, the gradient of a point $p \in \mathbb{R}^3$, $\nabla f(p) = -[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}]^T$ indicates the direction at p along which the rate of change of the function f is largest. A point $p \in \mathbb{R}^3$ is *critical* if the gradient at p is the zero vector, otherwise p is *regular*. In Morse theory [32], if the input function ρ is sufficient nice (more formally, it is a Morse function, ie with non-zero Hessians at the critical points), then there are four types of non-degenerate critical points for ρ defined on $\mathbb{R}^3$ - maxima, minima and two types of saddles of index 1 and 2, respectively.



(a) (b)

Figure 4: (a) An example of a 2D image converted to a density function with the graph (terrain) of this function shown in the right. (b) An example of 2D terrain (graph of 2D function): red points are maxima, green points are saddles, while blue points are minima. The white paths are some example of integral paths ending in minima. The black curves are a collection of 1-stable manifolds (between maxima and saddles).

Intuitively, if we view the graph of the density function $f : \mathbb{R}^3 \to \mathbb{R}$ as a terrain defined on

$\mathbb{R}^3 \times \mathbb{R}$, then maxima are the mountain peaks, while minima indicate valley basins. See Figure 4 where for illustration purpose, we provide an example for a 2D density function $f : \mathbb{R}^2 \to \mathbb{R}$: In this case, the terrain (graph of this function) is defined on $\mathbb{R}^2 \times \mathbb{R}$, consisting of points $(x, y, f(x, y))$ (that is, the height of a point $(x, y) \in \mathbb{R}^2$ represents its function value $f(x, y)$). For a function defined on $\mathbb{R}^2$, there are three types of non-degenerate critical points: maxima, minima and saddle points; see Figure 4 (b), where red dots are maxima, blue ones are minima, while the green dots are saddle points. We will use certain curves connecting maxima and index-2 saddle points in 3D case (or connecting maxima and saddle points for the 2D case) as a way to capture the hidden graph skeleton of this density function f. Below, we will first introduce the notations, and then provide the intuition behind the procedure.

An integral line $L : (0, 1) \to \mathbb{R}^d$ is a maximal path in $\mathbb{R}^d$, where tangent vectors are consistent with the gradient for all points on the line. Imagine putting a drop of water on the terrain: it will flow downwards following the gradient direction at any moment; if we negate the function value, then it will move upwards following the negation of the gradient direction. The trajectory of it (both downwards and upwards) forms the integral line passing through that point. The destination of an integral line is $dest(L) = \lim_{p\to 1} L(p)$ and the origin of an integral line is $ori(L) = \lim_{p\to 0} L(p)$. We also set by $dest(x)$ (resp. $ori(x)$) to be the destination (resp. the origin) of the integral line passing through x. The origin and destination of an integral line are necessarily critical points. See Figure 4 (b) for a 2D illustration. The stable manifold of a critical point p is defined as:

$$S(p) = \{p\} \cup \{x \in \mathbb{R}^d \mid \ dest(x) = p\}.$$

In other words, the stable manifold of a critical point p is the union of itself and all points whose integral lines eventually flow into p. Generically, for most points, the integral line passing through them will end at a minimum, forming a basin around this minimum. However, some of the integral line (starting from within a neighborhood of a maximum) will end at an index-$(d-1)$ saddle point, forming the separation between different basins around valleys. These integral lines are exactly the union of stable manifolds of index-$(d-1)$ saddle points. They form a network of connections between mountain peaks (maxima) to saddles then to other mountain peaks, separating different valley basins around minima. See Figure 4 (b) for a 2D example.

We use the stable manifolds of the index-2 saddles of f as the graph skeleton of the input density field $f : \mathbb{R}^3 \to \mathbb{R}$. Intuitively, along a hidden neuron branch, the density values should be higher than points off the neuron branch. Using this terrain illustration, this means that intuitively the mountain ridges of this terrain (connecting mountain peaks to saddles then to neighboring mountain peaks) correspond to where hidden neuron branches lie, as off the mountain ridges, the function values will decrease (flow into different minima / valley basins). The 1-stable manifolds of the index-2 saddle points capture such mountain ridges.

Remark. There is a dual concept of *unstable manifold* $U(p) = \{p\} \cup \{x \in \mathbb{R}^d \mid \ ori(x) = p\}$ of a critical point p, consisting of all points along integral lines originated from p (i.e, flowing away from p). In our case, for $d = 3$, the stable manifold of a p in ρ is identical to the unstable manifold of p in $-\rho$ (although the index of the critical point p changes from k to $3 - k$, for $k = 0, 1, 2, 3$). Computationally, as we will implement the above idea via discrete Morse theory in the discrete setting, it turns out that using the unstable manifolds of index-1 saddle points is much more efficient and simpler than using the stable manifolds for index-2 saddle points. Hence in our implementation, we will negate the density map and compute 1-unstable manifold for the function $-\rho : I \to \mathbb{R}$ instead. Note that such 1-unstable manifolds connect minima to index-1 saddles to other minima and separate different mountain peaks for the map $-\rho$.

Noise removal. Finally, note that the input images can be noisy, producing spurious critical points and thus spurious branches in the extracted graph skeleton (1-stable manifolds). Intuitively, we wish to identify such "spurious" critical points and ignore their corresponding 1-stable manifolds. To this end, we use the persistent homology, introduced in [20, 62], to identify these critical points. In particular, using the persistent homology induced by the so-called lower-star filtration w.r.to the density map $\rho : I \to \mathbb{R}$, one can obtain a *persistence* value for each critical point[2]. This "persistence" indicates the importance of the critical point in a meaningful manner, which intuitively corresponds to the amount of perturbation in the input function ρ one has to introduce in order to remove the (topological) feature introduced by this critical point. Critical points with small persistence are potentially caused by noise. Hence to remove noise, we will "smooth" these low-persistence critical points out, and consider only index-1 saddles whose persistence is larger than a given threshold $\tau \geq 0$, and output the union of 1-unstable manifolds of these saddles as the reconstructed graph skeleton $G \subset I$. For simplicity of illustration, we show an example of persistence induced by a 1D function in Figure 5. In our pipeline, here we typically use a very low threshold to remove obvious noise without disconnecting the reconstructed skeleton. More simplification is performed later after we retrieve the tree summary from this graph skeleton in Step 3.

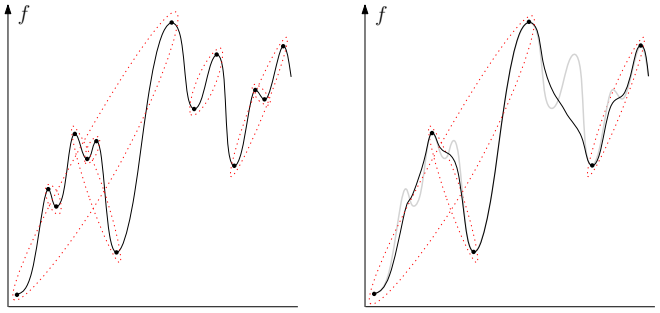


Figure 5: An example of the persistence pairing on a 1D function $f : \mathbb{R} \to \mathbb{R}$ as shown in (a): The persistence algorithm will pair up the critical points of function f (as indicated by dotted ellipses), and the height difference between each pair is the persistence (importance) of the two critical points involved. In (b), we simplify those low-persistence critical points and only important peaks/valleys remain.

Implementation in the discrete setting. In practice, we are given a triangulation K of the domain (3D cube) $I \subset \mathbb{R}^3$, and the density function ρ is given at the vertices of K. As mentioned earlier, our implementation only needs the so-called *2-skeleton* of K, that is, the collection of vertices, edges and triangles. Our software *DiMorSC(*K, ρ, τ*)* (DiMorSC stands for *Discrete Morse on Simplicial Complex*) will take $\rho : \mathsf{K} \to \mathbb{R}$ and a persistence threshold τ as input, and output the graph skeleton consisting of the 1-unstable manifolds for all index-1 saddles with persistence larger than τ. Following [1, 16, 45], the computation of the 1-unstable manifold for this discrete setting is done through the use of discrete Morse theory, which is combinatorial in nature – In particular, it only maintains the so-called *discrete gradient vector field on* K, which consists of a collection of vertex-edge and edge-triangle pairs, and never approximates "gradient vectors" in a numerical manner.

[2]We note that persistent homology is one of the most important development in the field of topological data analysis in the past two decades, and has already been applied to a broad range of applications [3, 11, 15, 21, 22, 27, 30, 38] due to its power in feature characterization and quantification.

The high-level description of our discrete-Morse based skeleton extraction algorithm in Algorithm 1. We note that this is similar to the *persistence-guided discrete Morse based* graph reconstruction framework introduced in [17] which builds upon [1, 16, 45] [3]. We thus will only provide a brief description below and for further details we refer the readers to [17] or to Chapter 5 and 6 of [51]. (We point out that our algorithm and its implementation in fact predates the work of [17].) To our best knowledge, our software is currently the only publicly available code to extract the 1-unstable manifold from an arbitrary simplicial complex. In Section 3, we will propose and incorporate a divide-and-conquer strategy into our software to handle large input images.

Algorithm 1 $G = \text{DiMorSC}(\mathsf{K}, f, \tau)$

1: Set $\hat{f} = -f$
2: $\mathcal{P}$ = persistence pairing induced by lower-star filtration of K w.r.t. $\hat{f}$
3: // $\mathcal{P}$ consists a set of vertex-edge and edge-triangle pairs
4: Initialize the discrete gradient vector field W on K
5: **for** each vertex-edge pair $\langle v, e\rangle \in \mathcal{R}$ s.t. $per(v, e) \leq \tau$ **do**
6: Cancel (simplify) this pair $\langle v, e\rangle$ and update the discrete gradient vector field W
7: **end for**
8: $G = \emptyset$
9: **for** each critical edge e with $per(e) > \tau$ **do**
10: $G = G \cup \{$ 1-unstable manifold(e) in W$\}$
11: **end for**
12: **return** G

In Algorithm 1, line 2 computes the persistence pairing to assign importance (persistence) of critical simplices (analogous to critical points in the smooth setting). The computation is done using the library PHAT [9], which provides state-of-the-art performance in computing persistence homology. Lines 3–7 simplify the discrete gradient vector fields by canceling (thus destroying) critical simplices with low persistence. This is analogous to the "smooth-out" of low-persistence critical points in the continuous setting as illustrated in Figure 5. Finally, lines 8–11 collect the 1-unstable manifolds for high-persistence critical edges (corresponding to index-1 saddle points) as the graph skeleton of density field ρ. The implementation details of Lines 3 – 11 can be found in the PhD dissertation of one of the co-authors, Suyi Wang [51].

2.4 Step 3: Summary tree extraction and simplification

The output of the above Morse-based skeletonization step is a geometric graph G and we will convert the graph into a (simplified) tree T as the summarization of the input 3D image I. We provide a further simplification strategy to control the level details of the final summary tree keeping in mind the application to skeletonizing tracer injections, so that skeletons that capture more or less detail can be produced.

Extraction of a rooted tree. Our input is the graph skeleton G extracted in Step 2. Note that each arc $e \in E$ in graph $G = (V, E)$ is realized by a polygonal path, consisting of edges from the input triangulation K. We now augment G to $\widehat{G} = (\widehat{V}, \widehat{E})$ so that its node set $\widehat{V}$ includes both those

[3] The work of [17] builds upon, but simplifies both conceptually and implementation speaking, the work of [1, 16, 45] for the specific case of extracting graph skeleton. However, those work are more general in the sense that they can compute higher dimensional (un)stable manifolds as well, while the framework formulated in [17] is specifically for 1-unstable manifolds and simpler.

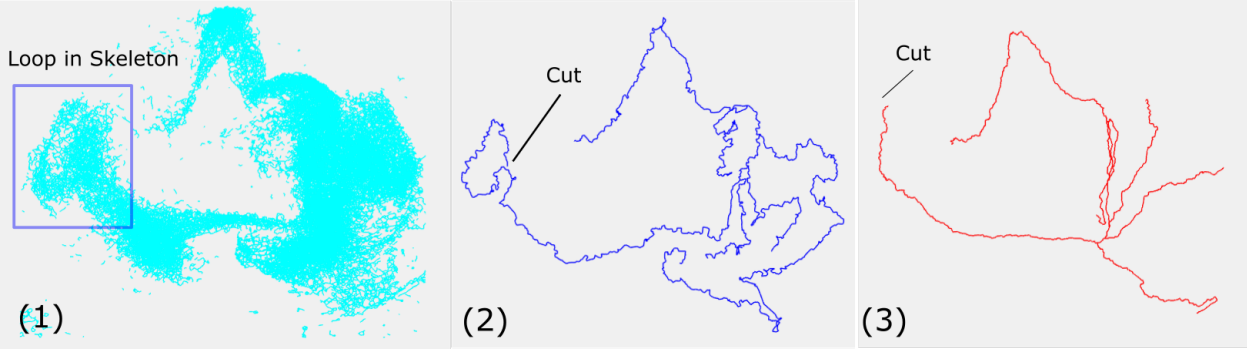


Figure 6: A comparison between the output tree extracted (and simplified for more clear view) via the minimum spanning tree approach, shown in (2), and via the shortest path tree approach, shown in (3); generated from the same input as shown (1). The two approaches resolve loop differently: a loop is always cut at a location locally furtherest away from the root for the shortest path tree, while there is less control of how the loop is cut for the minimum spanning tree.

from V and all vertices of the edges from each arc in E. See Figure 7 for an illustration where black dots are vertices in $\widehat{V}$. We now extract a rooted tree T from $\widehat{G}$, which is in fact a spanning tree of $\widehat{G}$ (that is, T contains all vertices in $\widehat{V}$, and edges of T are from $\widehat{E}$). Note that here we assume that $\widehat{G}$ is connected – if not, we will extract a tree summary for each connected component of $\widehat{G}$.

We also assume that we are given the choice of the tree root v – in our experiments, this is typically set to be the injection site, which is easy to infer due since the neuronal somata are concentrated at the injection site. In general, the choice of the root v can also be taken as the soma of a neuron in the single neuron reconstruction. We then take the shortest path tree w.r.to source v as the initial tree summary T, where the length of a path is measured by the number of edges in it. (That is, we assume that all edges in graph $\widehat{G}$ has weight 1.) It is possible to weight the edge by a quantity proportional to the inverse of the density ρ along this edge. Our software provides this option. However, our current choice appears to work well in current experiments.

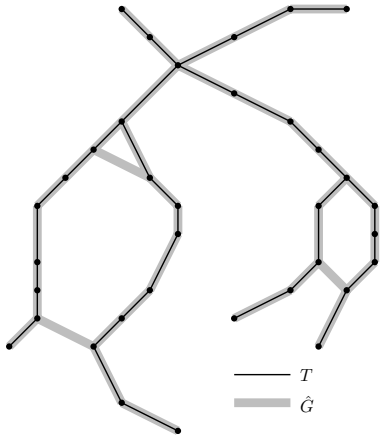


Figure 7

Note that it may also be possible to use the maximum spanning tree of $\widehat{G}$ (where the weight of each edge $(u, v) \in \widehat{E}$ equals $(\rho(u) + \rho(v))/2$). This strategy also makes sense as it encourages to include high density edges in the spanning tree. Indeed, we tested both methods in our experiments. The two strategies often generate similar results, although we observe that shortest path tree strategy tends to produce more natural trees while maximum-spanning-tree strategy sometimes introduces breaks in the middle of a long branch. Specifically, when a branch contains relatively weak signal (noise) in the middle resulting in its end point being mis-connected to other branches and thus forming a loop, making the maximum spanning tree more likely to cut the branch in the middle. In contrast, the shortest path tree (starting from a reliable choice of root) would cut the branch in the far end, which is more consistent to neuron morphology. An example is shown in Figure 6. In cases when there is no obvious choice of tree root, we have utilized the maximum spanning tree strategy.

Tree simplification. The initial tree T constructed above may contain an excess of detail. We further develop the following simplification strategy to allow the users to control the desired level of detail in the tree summary.

Specifically, given the tree $T = (\widehat{V}, E_T)$ rooted at v, we first assign a function $\mathsf{g} : \widehat{V} \to \mathbb{R}$ for all nodes in $\widehat{V}$ by, for any vertex $v \in \widehat{V}$,

$$\mathsf{g}(v) = \sum_{e \in \pi(\mathrm{v}, v)} length(e) * weight(e),$$

where $\pi(\mathrm{v}, v)$ is the unique tree path from the root v to node v. The length of an edge e, $length(e)$, is simply the Euclidean distance between the two endpoints of e. As for the weight of e, i.e, $weight(e)$, we provide two choices: (i) a uniform weight where $weight(e) = 1$ for all edges $e \in E_T$; and (ii) an intensity-based weight $weight(e) = (\rho(v_1) + \rho(v_2))/2$, where v_1, v_2 are adjacent vertices of edge e. Note that the function g is monotonically increasing along any root-to-leaf path in the tree T, and $\mathsf{g}(\mathrm{v}) = 0$.

Let a *leaf node* refer to any degree-1 node that is not the root, and a *junction node* denote any node with degree larger than 2. We now describe a natural *branch decompositon* procedure that partitions the tree T into a set of branches, and also assigns a measure of importance to each branch. During the simplification process, we simply remove those branches with small "importance". Interestingly, this decomposition as well as the importance assigned to each branch are exactly the information encoded in the persistent homology induced by the so-called *super-level set filtration* w.r.to the function g (viewed as a piecewise-linear function on T). This relation holds as the function g is monotone along any root-to-leaf path in T. Hence our simplification procedure essentially removes those branches (topological features) less important under persistent homology w.r.t. g. While we point out this connection to persistent homology here, in what follows, we will only describe the branch decomposition / persistence-assignment procedure.

Let $L = \{\ell_1, \ell_2, \ldots, \ell_t\} \subset \widehat{V}$ denote the leaf set. We will partition T into a set of $t = |L|$ branches $\Pi = \{\pi_1, \ldots, \pi_t\}$, each each branch π_i is a path from the leaf node ℓ_i to a junction node or to the root. The union of paths in Π equals T, while all branches are disjoint other than potentially at their two end points.

Specifically, at the beginning, we have a single tree T with root v. Let ℓ_1 be the leaf node with *largest* g function value, and let π_1 be the unique path from $\mathrm{v} = root(T)$ to ℓ_1. To continue, remove path π_1 from T, which will decompose T into a set of disjoint trees $T_1, \ldots, T_k$, the root of each of them will necessarily be a junction node in T. If $k = 0$, then this process terminates. Otherwise, we repeat the same procedure for each tree T_i, $i \in [1, k]$, recursively; and let Π denote the collection of all the branches obtained along the way. See Figure 8 for an example.

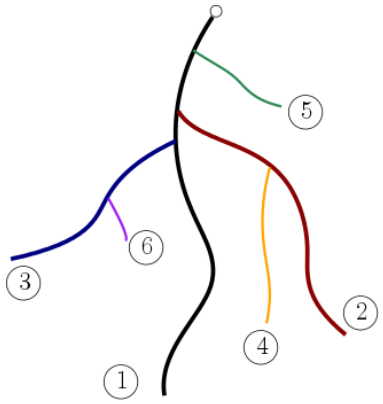


Figure 8: Branch decomposition of a rooted tree, where for simplicity assume $g(v)$ simply equals to distance (along the tree) to the root (white dot). The color paths are the decomposed branches, with indices indicate their persistence order (so number 1 indicates the branch with largest persistence).

After we compute the branch decomposition Π, we assign the persistence (importance) of each branch π_i as follows: Let ℓ_i and s_i be the two endpoints of π_i where ℓ_i is a leaf node and s_i is either a junction node or the root v of T. We set $per(\pi_i) = per(\ell_i) = per(s_i) = \mathsf{g}(\ell_i) - \mathsf{g}(s_i)$. As said earlier, it turns out that the set of pairings $\{(\ell_i, s_i) \mid i \in [1, t]\}$ is exactly the set of

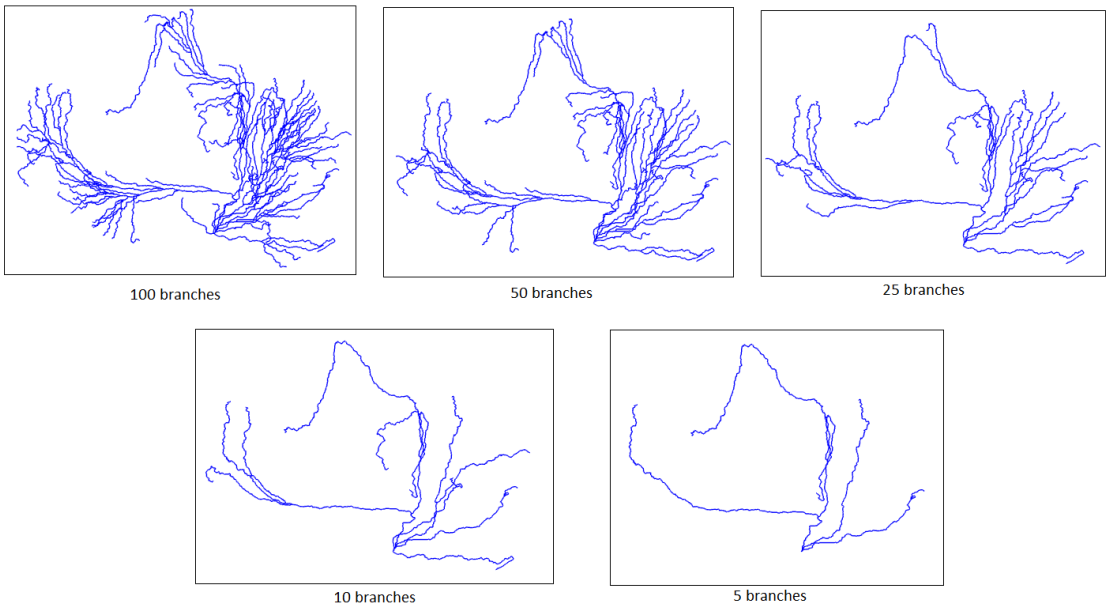


Figure 9: Simplification with branches of 100, 50, 25, 10 and 5, respectively.

persistence pairing produced by the persistent homology induced by the super-level set filtration of g, and the persistence of the corresponding branch equals to the persistence of the pair (ℓ_i, s_i).

Given a threshold τ, to remove unnecessary details in T, we simply output the subtree T_τ formed by the union of branches from Π with persistence larger than τ. It is easy to show that the union of these branches is necessarily connected (i.e, a single tree) if the input tree T is connected. See Figure 9 for examples.

3 Divide-and-conquer strategy for handling large images

The tracer-injected data sets have around 300 images of brain sections in each brain volume. Each of these section images have $18000 * 24000$ pixels creating a rather large dataset. This type of data is too large to fit into memory and to be processed all together at the same time. Hence we have developed the following divide-and-conquer approach to handle such large images.

In the high level, our approach (i) partitions the input image stack I into k small tiles (cubes) $\Omega_i, i \leq k$, (ii) extracts graph skeleton G_i in tile Ω_i and (iii) merges all the results G_i into a single graph skeleton G.

In our experiments, the size of each tile is chosen to be $512 * 512$ in xy-direction and we have not partitioned in the z-direction as the number of slices is typically only a few hundreds. (However, one can easily perform partitioning in z-direction as well if the needs arise.) This tile size represents a good trade off between information encoded in each tile and the processing time. we have also set a small overlapping area (5 pixels) between adjacent tiles.

See Figure 10 for an example, where different colors indicate different tiles and they share an overlapping area. Then the 1-stable manifold is extracted for each tile using Algorithm 1 presented in the previous section. The key step is (iii), the merging of the graph skeletons G_is into a single graph G representing the skeleton of the merged tiles. We describe how to perform the merging step now.

Let $\mathsf{K}_i \subseteq \mathsf{K}$ denote the subtriangulation of tile Ω_i, $i \in [1, k]$. Let V_i be the set of internal

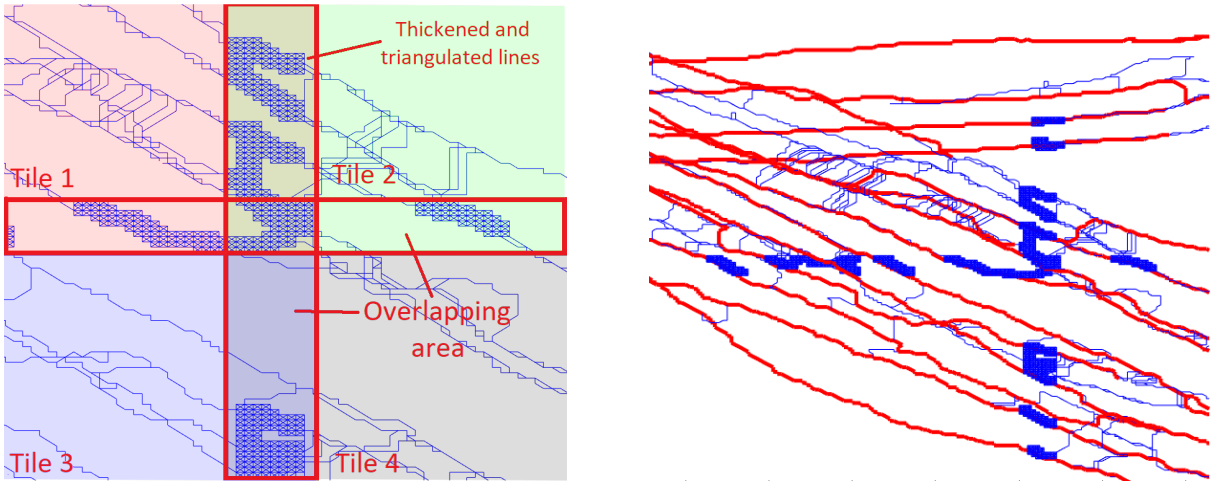


Figure 10: (a) The original domain is divided into four tiles (different colors). The blue graphs are the reconstruction within each tile, and the blue cells are those in K_{merge}. (b) Blue curves/cells are K_{merge} and the red graph is the merged graph and then simplified.

vertices for Ω_i and let V_i^b be the set of vertices in all overlapping area, $\bigcup_j \Omega_i \cap \Omega_j$, between Ω_i and any neighborhing tile Ω_j. Recall that the graph skeleton G_i is extracted from the density function $\rho_i : \mathsf{K}_i \to \mathbb{R}$, which is the restriction of ρ to K_i. Let E_i be the set of edges in G_i.

To merge G_is in a natural manner, we will leverage the 1-unstable manifold based framework again: First, we will build a new simplicial complex $\mathsf{K}_{merge} \subseteq \mathsf{K}$ as follows: For each $i \in [0, k]$, we take a small neighborhood (a small triangulated cubic region) around every vertex $v \in V_i^b$; let C_i denote the union of such triangulated neighborhoods for all vertices in V_i^b. We set $\mathsf{K}_{merge} = \bigcup_i (\mathsf{G}_i \cup C_i)$; see Figure 10 (a).

Next, we diffuse the function values of $\rho(v)$, for all $v \in \bigcup_i V_i^b$, to vertices in K_{merge} using a Gaussian kernel, and establish a function $\rho' : \mathsf{K}_{merge} \to \mathbb{R}$ on K_{merge}. In other words, for any $u \in \mathsf{K}_{merge}$, $\rho'(u) = \sum_{v \in \cup_i V_i^b} e^{-\frac{\|v-u\|^2}{2\sigma^2}} \rho(v)$, where σ^2 is the variance of the Gaussian kernel.

Finally, we perform Algorithm 1 on $\rho' : \mathsf{K}_{merge} \to \mathbb{R}$, and extract the final merged graph G.

Note that compared to the size of each tile Ω_i, the extracted 1-skeletons G_i from it is much smaller (as it intuitively only represents potential signals in input image). Hence the size of K_{merge} is potentially far smaller than the size of K, and the memory cost for processing an individual tile as well as the merging process is controlled at a reasonable amount, allowing our algorithm to handle large data not fit in memory. This approach is only a small modification of the original pipeline, but can greatly increase the size of data it can handle.

4 Results

4.1 Single neuron tracing

As proof of principle demonstration of our proposed pipeline, we first show the performance of our discrete Morse based framework in reconstructing neurons from the Olfactory Projection Fibers dataset (OP dataset) provided as part of the DIADEM challenge [39]. The dataset contains nine stacks of drosophila olfactory axonal fibers in Olfactory Bulb and the roughly 512*512*60 resolution

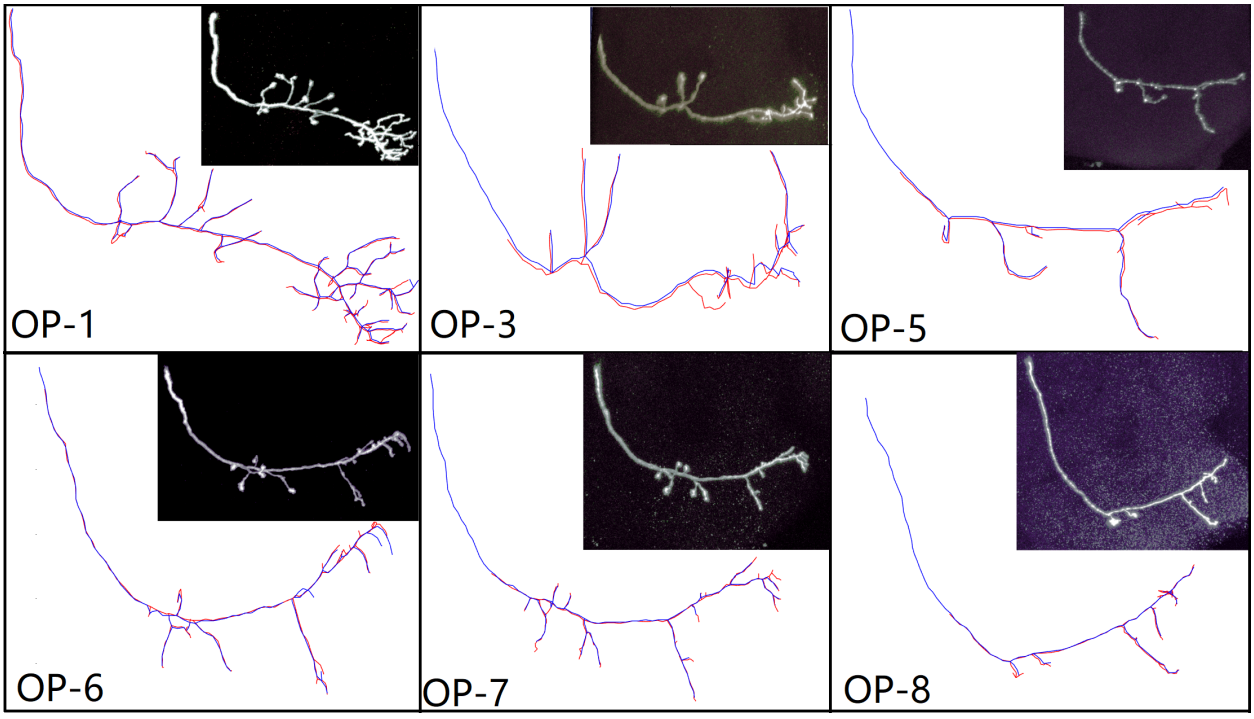
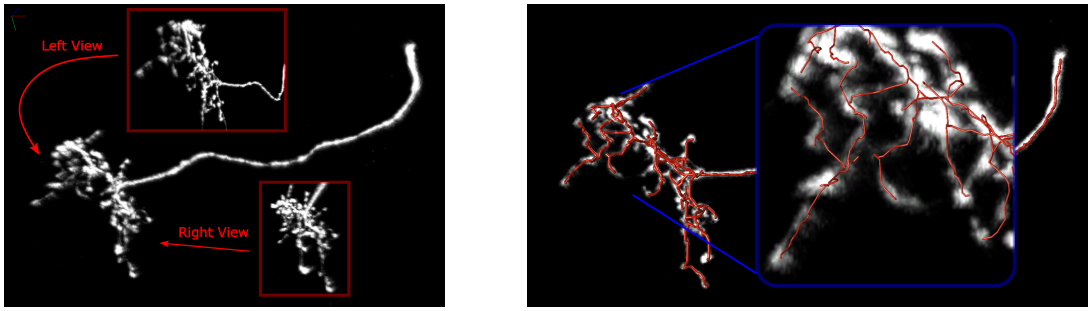


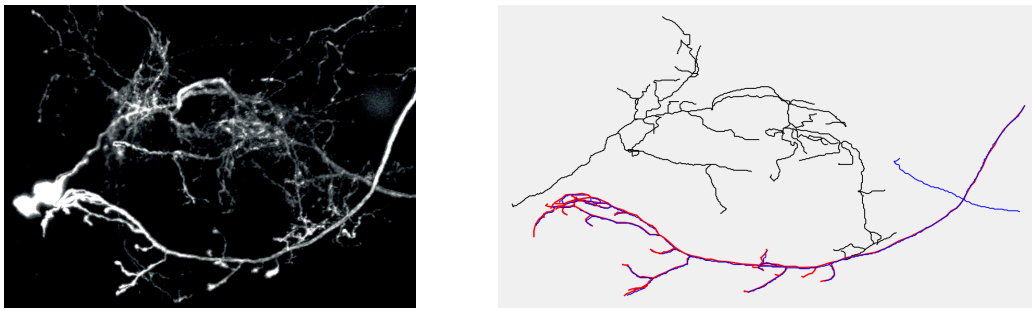
Figure 11: The result (blue) superpose on ground truth (red) of Olfactory Projection dataset in DIADEM challenge [39].



(a) (b)

Figure 12: (a) The input signal has non-uniform strength with gaps (the dataset is OP-9 from DIADEM dataset). However, the reconstruction (as shown in (b)) based on global topological structure of the signal density field can still connect through them.

images are acquired by 2-channel confocal microscopy method. This dataset is accompanied by manual tracing results which can serve as ground truth, allowing for qualitatively comparison of the output of reconstruction with the ground truth, called DIADEM score [25]. Below, we show our reconstruction visually, and compare them qualitatively with other popular methods using the DIADEM score.



(a) (b)

Figure 13: In the data set OP-2 from DIADEM, there are actually two neurons shown in the image, with the ground truth given for the front one. Our method reconstructs both neurons.

Figure 11 shows the tracing results of DiMorSC for OP dataset No. 1, 3, 5, 6, 7, and 8. Datasets OP-2 and OP-9 will be reported later in Figure 12 and 13. As we can see that even though the input signal may not always uniform (see Figure 12 for a detailed example on dataset No. 9 referred to as OP-9), as our method relies on the global structure of the signal density field, it still connect through these gaps. We also show the input and reconstruction of OP-2 data set in Figure 13, where as we can see that interestingly, there are actually two neurons in this image. The ground truth is given only for the front one. Our method can reconstruct both neurons as shown in blue and black colors.

Set	DiMorSC	APP2	Smart tracing	SNAKE
1	0.914	0.796	0.853	0.827
3	0.765	0	0.605	0.766
4	0.804	0.727	0.79	0.704
5	0.755	0.389	0	0
6	0.858	0.857	0.779	0.667
7	0.923	0.773	0.906	0.788
8	0.849	0.373	0.696	0.725
9	0.833	0.786	0.739	0.657

Table 1: DIADEM scores for various OP datasets.

For quantitative evaluation, in Table 1 we report the *DIADEM score* of our algorithm and that of the output by APP2 [54], SmartTracking [12] and SNAKE [53] algorithms (which are three state-of-the-art algorithms in single neuron tracing). The DIADEM score [25] is a metric in range (0,1) that measures the similarities between reconstruction and ground truth, where the higher the value is the more similar they are. This score has been used to evaluate the algorithms in the DIADEM challenge. As we can see, we obtained similar or better DIADEM score in all datasets. We note that the DIADEM score could be sensitive to both the root location and the geometric information of branches. For example, simply smoothing our result sometimes increases the DIADEM score,

Set	Data Loading	DM Step1	DM Step2	DM Step3	APP2	Smart tracing	SNAKE
1	0.67	8.6	19.07	0.37	<1s	6min	24
3	0.70	5.51	12.7	0.26	<1s	8min	-
4	0.73	6.76	16.51	0.35	<1s	8min13s	-
5	0.83	4.85	6.99	0.24	<1s	6m15s	-
6	1.02	5.60	6.49	0.29	<1s	4m07s	35
7	0.71	5.08	7.94	0.28	<1s	6m42s	20
8	0.95	7.56	13.28	0.30	<1s	13min13s	28
9	0.96	6.74	10.89	0.33	<1s	2m55s	31

Table 2: Running time in seconds for various OP datasets. "DM" stands for our DiMorSC algorithm, and columns 3–5 show the running time for step 1, 2, and 3 respectively.

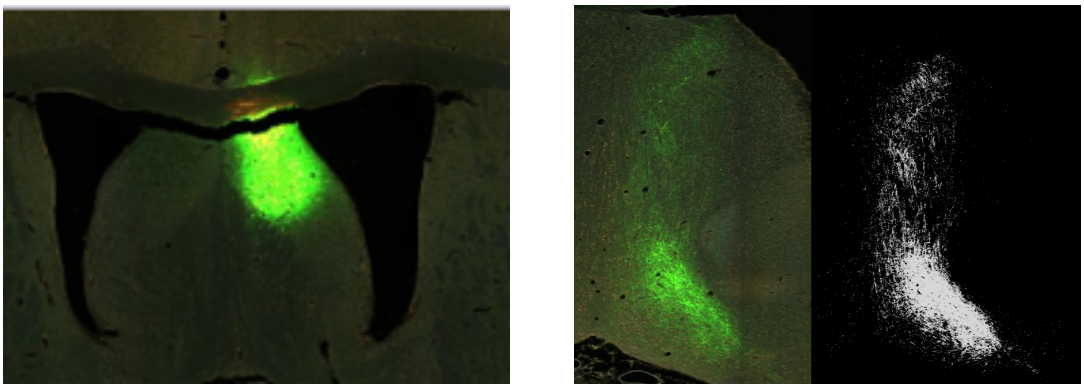


Figure 14: LSc injection (in the left image) and signal detection (right images)

however, the smoothed branches are visually less aligned with the ground truth. The root location provided in the data sometimes lies obviously in the middle of a long branch.

we also report the running time of our algorithm, as well as for the three algorithms mentioned above in Table 2. In general, our algorithm finishes in 15s for the OP dataset, which is slower than that of APP2 ($< 1s$), comparable to SNAKE (average 25s) and faster than Smart tracing (average 6 min). However, we note that this is a simple implementation of our algorithm and we believe that it can be further improved. Indeed, we note that recent observations in [18] can significantly simplify our algorithm and improve its time efficiency.

4.2 Mesoscopic summarization

We also report our results anterograde tracer injected data from the Mouse Brain Architecture Project[33], which will be referred to as an 'MBAP' brain. The MBAP brain dataset is obtained from whole mouse brains where specific brain areas have been injected with an AAV florescent tracer. The example brain dataset contains a stack of about 270 images of brain sections with 18000*24000 pixels per section. In every image slice (2D image), each pixel is 0.46 microns in both dimensions and vertically, the distance between two consecutive image slices is 20 microns. In the example below, a brain with a single injection (green fluoroscence) in the brain area, the Lateral septal nucleus-caudal part(LSc) is shown. In addition to the large size of the data and the anistropic sampling, there is background florescence in the image. Although it may be possible to identify the neuron signal visually in the images by careful inspection, automatic tracing of the

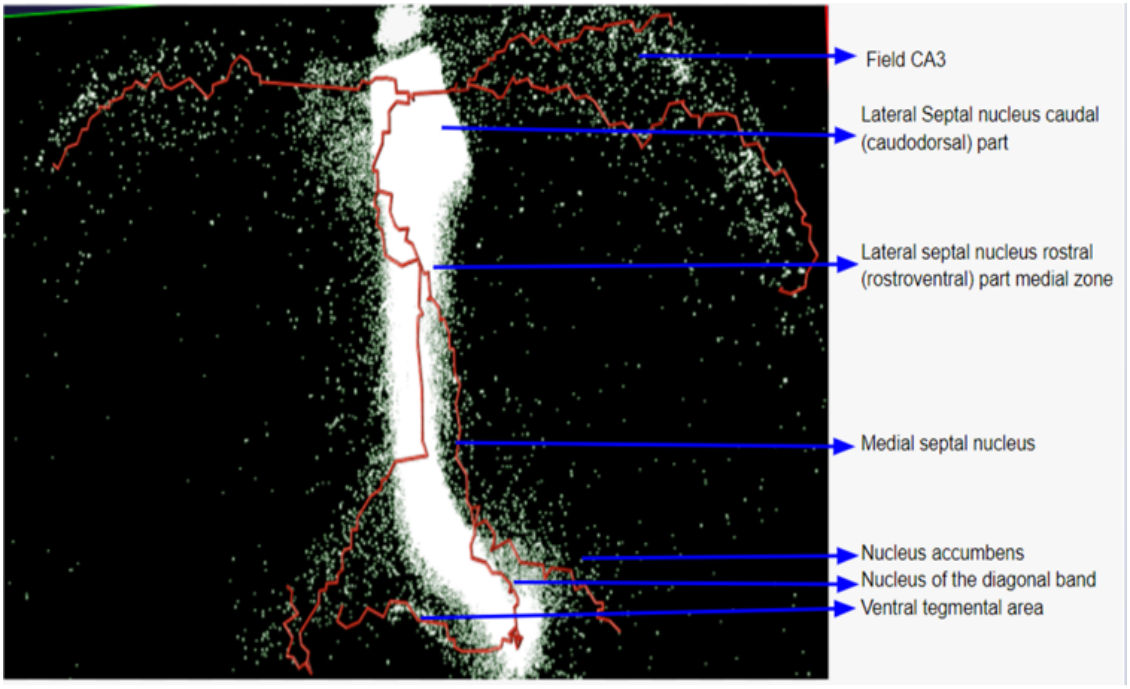


Figure 15: Skeletonization result on the preprocessing data

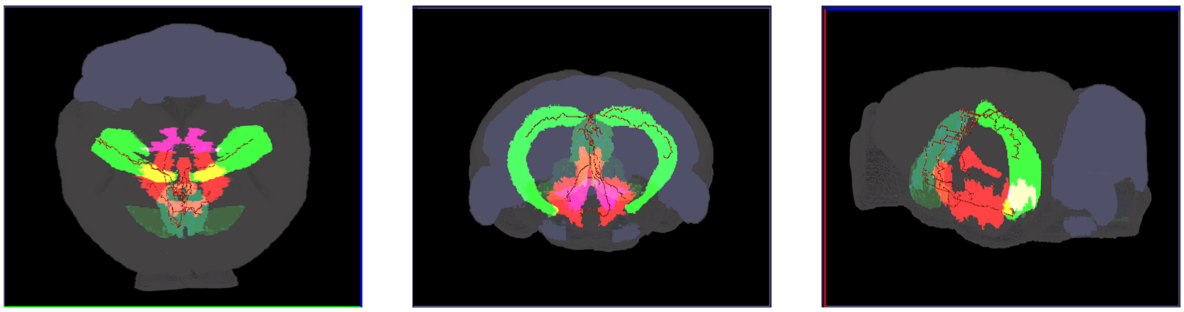


Figure 16: Neuron connectivity atlas

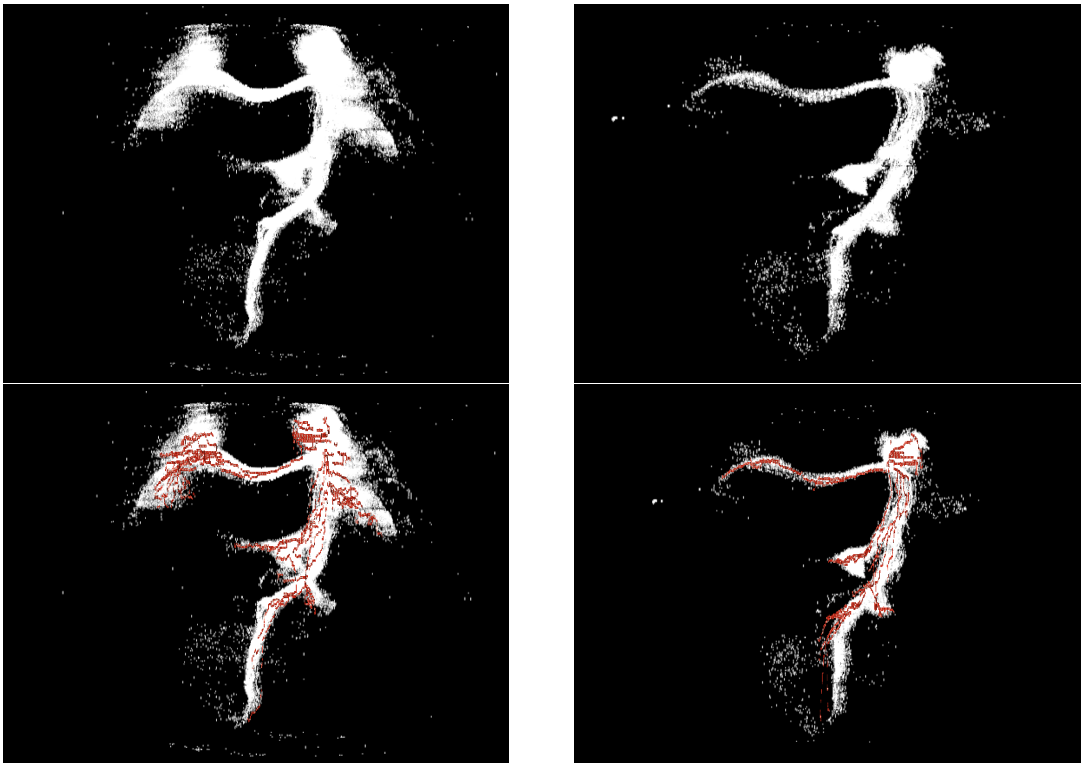


Figure 17: Motor Cortex injection and summarization

trajectories of bundles of axons presents a significant challenge.

The specific pre-processing procedure described in section 2.2 is applied to remove the background. See Fig 14 for fluorescence injection region and preprocessing result.

Our pipeline allows for the MBAP data to be processed in full resolution and also at a downsampled resolution. When processed at full resolution, the images are partitioned into tiles of 512*512 pixels with 5-pixels of overlapping area between adjacent tiles. The 1-stable manifolds are extracted from the tiles and the extraction results are merged using the stitching strategy described above. (Diffusing range 5 pixels) We also report the results on the downsampled image (with a resolution of 2850*3000*250) where there is no need for the divide-and-conquer approach to optimize the computing requirements. The results on this particular example shows that down-sampled data appears to be sufficient for summarization purposes. Figure 15 shows the results of the example LSc injection, the injection summarized tree (after simplification to show only the main trend) is shown superposed on the preprocessed input image. We have also added the specific brain areas (blue arrows) to denote the targets of the injection. The input is a 3D point cloud generated from 2D pre-processing detection.

This summarization is annotated and aligned with BAMS database [34], which is considered as a trust worthy standard of connectivity in rodent brains. Specifically in Figure 16, each color in BAMS data represents a region connected to the injection site and our neuron summary is consistent with the BAMS data.

We have applied our tree summarization framework to a collection of MBAP datasets; two examples of brains with injections in the motor cortex are shown in Figure 17.

In Figure 18, we compare our output with that of APP2 (computed via software platform Vaa3D): We note that APP2 also captures the four major branches from the injection site. However,

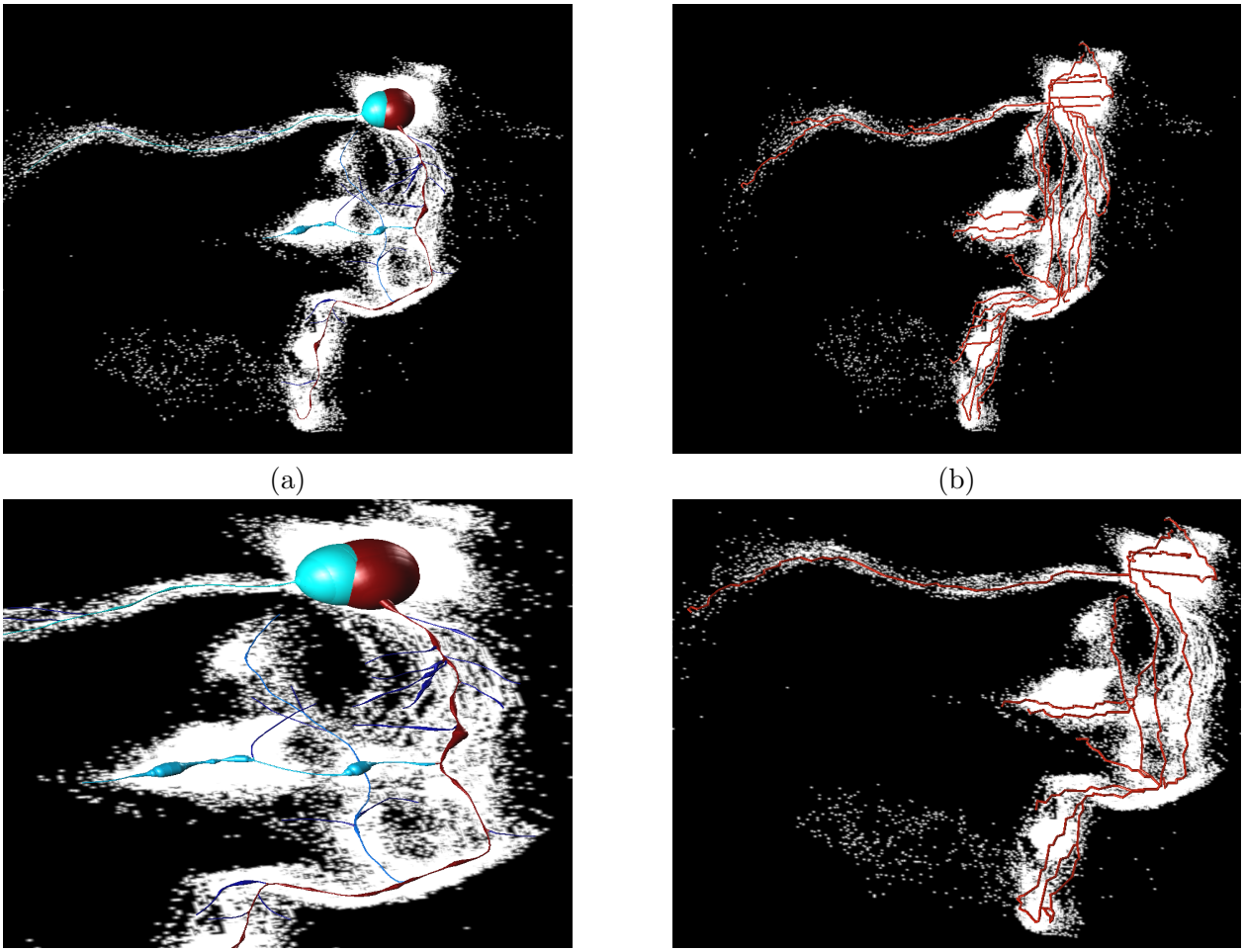


Figure 18: Comparison of the result of (a) APP2, and (b) our output. We note that APP2 captures the four major branches. However, it does not capture the details: horizontal small branches are traced instead of tracing the vertical signals which are visible in the input; we show a zoomed-in view in (c). In contrast, as we increase the number of branches, our output trace those different bundles of the downwards signals better. A more simplified output by our algorithm is shown in (d).

the additional branches tend to be local (semi-horizontal small branches in 18 (a) and (c)), and miss significant axonal bundles (instead, the small semi-horizontal branches cut across those bundles). We suspect that this could be partially due to the gaps in signal. In contrast, as our algorithm uses a global view of data, our reconstruction traces the axonal bundles better; see Figure 18 (b) and (d) for two different levels of simplification. (We note that our output are guaranteed to be trees, although due to occlusion and 2D projection in the pictures it may appear that there are loops.)

References

[1] G. A. Phd thesis. *Univ. California Berkeley*, 2008.

[2] L. Acciai, P. Soda, and G. Iannello. Automated neuron tracing methods: An updated account. *Neuroinformatics*, 14(4):353–367, Oct 2016.

[3] P. K. Agarwal, H. Edelsbrunner, J. Harer, and Y. Wang. Extreme elevation on a 2-manifold. *Discrete and Computational Geometry (DCG)*, 36(4):553–572, 2006.

[4] K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, and B. Roysam. Rapid automated three-dimensional tracing of neurons from confocal image stacks. *Trans. Info. Tech. Biomed.*, 6(2):171–187, June 2002.

[5] B. Arenkiel. *Neural Tracing Methods: Tracing Neurons and Their Connections*. Neuromethods. Springer New York, 2014.

[6] E. Bas and D. Erdogmus. Principal curves as skeletons of tubular objects. *Neuroinformatics*, 9(2):181–191, Sep 2011.

[7] W. Baschong, R. Suetterlin, and R. H. Laeng. Control of autofluorescence of archival formaldehyde-fixed, paraffin-embedded tissue in confocal laser scanning microscopy (clsm). *Journal of Histochemistry & Cytochemistry*, 49(12):1565–1571, 2001. PMID: 11724904.

[8] S. Basu, W. T. Ooi, and D. Racoceanu. Improved marked point process priors for single neurite tracing. pages 1–4, 06 2014.

[9] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78(Supplement C):76 – 90, 2017. Algorithms and Software for Computational Topology.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov 2001.

[11] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 124–135, New York, NY, USA, 2004. ACM.

[12] H. Chen, H. Xiao, T. Liu, and H. Peng. Smarttracing: self-learning-based neuron reconstruction. *Brain Informatics*, 2(3):135–144, 2015.

[13] A. Choromanska, S.-F. Chang, and R. Yuste. Automatic reconstruction of neural morphologies with multi-scale tracking. *Frontiers in Neural Circuits*, 6:25, 2012.

[14] P. Chothani, V. Mehta, and A. Stepanyants. Automated tracing of neurites from light microscopy stacks of images. *Neuroinformatics*, 9(2):263–278, Sep 2011.

[15] M. K. Chung, P. Bubenik, and P. T. Kim. Persistence diagrams of cortical surface data. In J. L. Prince, D. L. Pham, and K. J. Myers, editors, *Information Processing in Medical Imaging*, pages 386–397, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[16] O. Delgado-Friedrichs, V. Robins, and A. Sheppard. Skeletonization and partitioning of digital images using discrete morse theory. *IEEE Trans. Pattern Anal. Machine Intelligence*, 37(3):654–666, March 2015.

[17] T. K. dey, J. Wang, and Y. Wang. Improved road network reconstruction using discrete morse theory. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, page 58, 2017.

[18] T. K. Dey, J. Wang, and Y. Wang. Graph reconstruction by discrete Morse theory. In *Sympos. Comput. Geometry (SoCG), to appear*, 2018.

[19] D. E. Donohue and G. A. Ascoli. Automated reconstruction of neuronal morphology: An overview. *Brain Research Reviews*, 67(1):94 – 102, 2011.

[20] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002.

[21] H. Edelsbrunner and J. Harer. Persistent homology – a survey.

[22] M. Ferri. Persistent topology for natural data analysis - A survey. *ArXiv e-prints*, June 2017.

[23] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. *Multiscale vessel enhancement filtering*, pages 130–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[24] R. Gala, J. Chapeton, J. Jitesh, C. Bhavsar, and A. Stepanyants. Active learning of neuron morphology for accurate automated tracing of neurites. *Frontiers in Neuroanatomy*, 8:37, 2014.

[25] T. A. Gillette, K. M. Brown, and G. A. Ascoli. The diadem metric: Comparing multiple reconstructions of the same neuron. *Neuroinformatics*, 9(2):233, Apr 2011.

[26] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Trans. Visualization Computer Graphics*, 13(6):1432–1439, Nov 2007.

[27] J. Lamar-León, E. B. García-Reyes, and R. Gonzalez-Diaz. Human gait identification using persistent homology. In L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 244–251, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] P.-C. Lee, Y.-T. Ching, H. M. Chang, and A.-S. Chiang. A semi-automatic method for neuron centerline extraction in confocal microscopic image stack. In *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 959–962, May 2008.

[29] P.-C. Lee, C.-C. Chuang, A.-S. Chiang, and Y.-T. Ching. High-throughput computer method for 3d neuronal structure reconstruction from the image stack of the drosophila brain and its applications. *PLOS Computational Biology*, 8(9):1–12, 09 2012.

[30] J.-Y. Liu, S.-K. Jeng, and Y.-H. Yang. Applying topological persistence in convolutional neural network for music audio signals. *CoRR*, abs/1608.07373, 2016.

[31] E. Meijering. Neuron tracing in perspective. *Cytometry Part A*, 77A(7):693–704, 2010.

[32] J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

[33] Brain architecture project. http://mouse.brainarchitecture.org/homepage/.

[34] Bams dataset. https://bams1.org/.

[35] D. Myatt, T. Hadlington, G. Ascoli, and S. Nasuto. Neuromantic from semi-manual to semi-automatic reconstruction of neuron morphology. *Frontiers in Neuroinformatics*, 6:4, 2012.

[36] S. W. Oh, J. A. Harris, L. Ng, B. Winslow, N. Cain, S. Mihalas, Q. Wang, C. Lau, L. Kuan, A. M. Henry, et al. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

[37] H. Peng, F. Long, and G. Myers. Automatic 3d neuron tracing using all-path pruning. *Bioinformatics*, 27(13):i239, 2011.

[38] D. Platt, S. Basu, P. Zalloua, and L. Parida. Characterizing redescriptions using persistent homology to isolate genetic pathways contributing to pathogenesis. 10, 01 2016.

[39] Diadem challenge. http://diademchallenge.org.

[40] Dimorsc. http://github.com/SuyiWang/DiMorSC.

[41] Vaa3d. http://vaa3d.org.

[42] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Trans. Pattern Anal. Machine Intelligence*, 33(8):1646–1658, Aug 2011.

[43] S. Schmitt, J. F. Evers, C. Duch, M. Scholz, and K. Obermayer. New methods for the computer-assisted 3-d reconstruction of neurons from confocal image stacks. *NeuroImage*, 23(4):1283 – 1298, 2004.

[44] A. Sironi, V. Lepetit, and P. Fua. Projection onto the manifold of elongated structures for accurate extraction. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 316–324, Dec 2015.

[45] T. Sousbie. The persistent cosmic web and its filamentary structure - I. Theory and implementation. 414:350–383, June 2011.

[46] R. Srinivasan, X. Zhou, E. Miller, J. Lu, J. Litchman, and S. T. C. Wong. Automated axon tracking of 3d confocal laser scanning microscopy images using guided probabilistic region merging. *Neuroinformatics*, 5(3):189–203, Sep 2007.

[47] D. Sui, K. Wang, J. Chae, Y. Zhang, and H. Zhang. A pipeline for neuron reconstruction based on spatial sliding volume filter seeding. 2014:386974, 07 2014.

[48] E. Türetken, G. González, C. Blum, and P. Fua. Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9(2):279–302, Sep 2011.

[49] E. Tretken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing loopy curvilinear structures using integer programming. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1822–1829, June 2013.

[50] Z. Vasilkoski and A. Stepanyants. Detection of the optimal neuron traces in confocal microscopy images. *Journal of Neuroscience Methods*, 178(1):197 – 204, 2009.

[51] S. Wang. *Analyzing data with 1D non-linear shapes using topological methods*. PhD thesis, The Ohio State University, Computer Science and Engineering Department, 2018.

[52] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 25:1–25:10, New York, NY, USA, 2015. ACM.

[53] Y. Wang, A. Narayanaswamy, C.-L. Tsai, and B. Roysam. A broadly applicable 3-d neuron tracing method based on open-curve snake. *Neuroinformatics*, 9(2):193–217, 2011.

[54] H. Xiao and H. Peng. App2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics*, 29(11):1448, 2013.

[55] J. Yang, P. T. Gonzalez-Bellido, and H. Peng. A distance-field based automatic neuron tracing method. *BMC Bioinformatics*, 14(1):93, Mar 2013.

[56] X. Yuan, J. T. Trachtenberg, S. M. Potter, and B. Roysam. Mdl constrained 3-d grayscale skeletonization algorithm for automated extraction of dendrites and spines from fluorescence confocal images. *Neuroinformatics*, 7(4):213, 2009.

[57] Y. Zhang, X. Zhou, A. Degterev, M. Lipinski, D. Adjeroh, J. Yuan, and S. T. Wong. A novel tracing algorithm for high throughput imaging: Screening of neuron-based assays. *Journal of Neuroscience Methods*, 160(1):149 – 162, 2007.

[58] T. Zhao, J. Xie, F. Amat, N. Clack, P. Ahammad, H. Peng, F. Long, and E. Myers. Automated reconstruction of neuronal morphology based on local geometrical and global structural models. *Neuroinformatics*, 9(2):247–261, Sep 2011.

[59] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, Jul 1999.

[60] Z. Zhou, X. Liu, B. Long, and H. Peng. Tremap: Automatic 3d neuron reconstruction based on tracing, reverse mapping and assembling of 2d projections. *Neuroinformatics*, 14(1):41–50, Jan 2016.

[61] Z. Zhou, S. A. Sorensen, and H. Peng. Neuron crawler: An automatic tracing algorithm for very large neuron images. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 870–874, April 2015.

[62] A. J. Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.