Neuroinformatics (2011) 9:279-302
DOI 10.1007/s12021-011-9122-1

Automated Reconstruction of Dendritic and Axonal Trees by Global Optimization with Geometric Priors

Engin Türetken · Germán González · Christian Blum · Pascal Fua

Received: 14 October 2010 / Accepted: 15 May 2011

Abstract We present a novel probabilistic approach to fully automated delineation of tree structures in noisy 2D images and 3D image stacks. Unlike earlier methods that rely mostly on local evidence, ours builds a set of candidate trees over many different subsets of points likely to belong to the optimal tree and then chooses the best one according to a global objective function that combines image evidence with geometric priors. Since the best tree does not necessarily span all the points, the algorithm is able to eliminate false detections while retaining the correct tree topology.

Manually annotated brightfield micrographs, retinal scans and the DIADEM challenge datasets are used to evaluate the performance of our method. We used the DIADEM metric to quantitatively evaluate the topological accuracy of the reconstructions and showed that the use of the geometric regularization yields a substantial improvement.

Keywords DIADEM · Tree Reconstruction · Global Optimization · Minimum Arborescence · k-MST · Ant Colony Optimization

1 Introduction

Tree-like structures, such as dendritic, vascular, or bronchial networks, are pervasive in biological systems. With the advent of modern acquisition techniques that produce endless streams of imagery, there has been renewed interest in automated delineation to exploit this

E. Türetken · G. González · P. Fua
Computer Vision Laboratory,
Faculté Informatique et Communications,
École Polytechnique Fédérale de Lausanne,
CH-1015 Lausanne, Switzerland
Tel.: (+41 21) 693 75 19
Fax: (+41 21) 693 75 20
E-mail: engin.turetken@epfl.ch

C. Blum
ALBCOM, Dept. Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya,
Jordi Girona 1-3, Omega 112 Campus Nord
E-08034 Barcelona, Spain

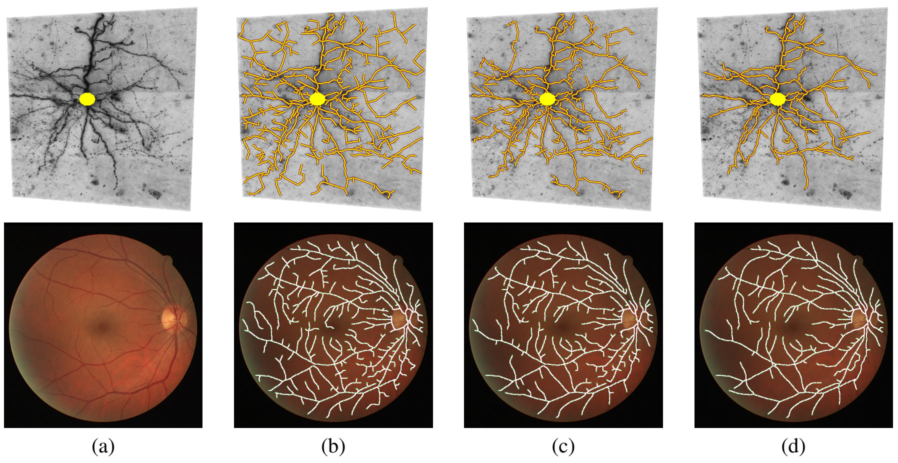


Fig. 1: Delineation results in 3D (top row) and 2D (bottom row). (a) A brightfield micrograph of a neuron and a scan of a retinal blood vessel network. (b) Minimum spanning trees. (c) Reconstructions obtained without geometric regularization. (d) Reconstructions obtained with geometric regularization. This figure, as most others in this paper, is best viewed in color because it contains overlays.

data. However, despite many years of sustained effort, automated techniques remain fragile and error-prone. In this paper, we use 3D optical micrographs of neurons and 2D retinal fundus images to demonstrate the importance of taking global tree structure and geometry into account to improve topological accuracy of the delineations.

More specifically, we rely on a machine learning approach to assign to image voxels probabilities of belonging to the centerline of a filament. We then select evenly spaced high-probability voxels that we treat as anchor points and connect them using maximum-probability paths. This turns our set of N anchor points into a weighted graph in which we look for minimum-weight trees that span $k < N$ of its edges, which is known as the k-Minimum Spanning Tree (k-MST) problem (Garg, 1996). Finding the optimal k-MST is NP-hard but an Ant Colony Optimization scheme we developed in earlier work (Blum and Blesa, 2005) has proved effective at generating good approximations. Such an approach is in contrast to more traditional ones (Fischler and Heller, 1998; Gonzalez et al, 2008) that use a minimum spanning tree to link all vertices and then prune the branches that do not conform to a shape or image appearance criterion. Such methods are faster and can eliminate spurious branches, but cannot recover from incorrect connectivity in the initial spanning tree. By contrast, when k is in the right range, the tree spanning only k edges does not suffer from this problem. We present an automated way to estimate the optimal k value by assigning probabilistic costs to trees and selecting the one that minimizes this cost. As a result, we obtain improved reconstructions such as those depicted in the third column of Fig. 1.

As shown in the last column of Fig. 1, these results can be further improved by incorporating into the tree reconstruction algorithm a regularization prior that rewards geometric consistency between consecutive edges and nodes. Unfortunately, this improvement comes at the cost of an increased computational complexity because the algorithm has to deal with pairs of edges as opposed to single ones. Nevertheless, we were able to extend our earlier optimization algorithm (Blum and Blesa, 2005) to handle the pairwise terms and compute near-optimal trees for all possible k values at little extra cost.

This results in a generic and fully automated technique. Our contribution is therefore both a global optimization approach to finding optimal trees and a practical algorithm to computing good approximations in an acceptably short time, even though the underlying problem is NP-hard.

2 Related Work

Most automated delineation techniques rely on a local *tubularity* measure that can be postulated *a priori* (Frangi et al, 1998; Law and Chung, 2008), optimized to find specific patterns (Jacob and Unser, 2004; Meijering et al, 2004), or learned (Santamaría-Pang et al, 2007; Gonzalez et al, 2009) from training data. Given an image stack, they compute a tubularity image in which this measure is computed at each voxel. The algorithms that use such tubularity scores can be roughly categorized into two classes.

The first class includes methods that use segmentation of the tubularity image such as thinning-based methods, which perform skeletonization of this segmentation (Weaver et al, 2004; Xu et al, 2009), and active contour-based methods, which are initialized from it (Cai et al, 2006; Vasilkoski and Stepanyants, 2009). Such methods are shown to be effective and efficient, when a very good segmentation is given or can be reliably obtained. In practice, however, such segmentations are hard to obtain. In particular, thinning-based methods often produce disconnected components and artifacts on noisy data, which then require considerable post-processing and analysis to merge into a meaningful tree.

The second class involves explicitly delineating the tree in the tubularity image. It includes tracking methods that start from a set of seed points and recursively trace high-tubularity paths (Can et al, 1999; Al-Kofahi et al, 2002; Yedidya and Hartley, 2008). These techniques are computationally efficient because the tubularity measure only needs to be evaluated for a small subset of the image volume. However, due to their incremental nature, local tracing errors may result in large topological perturbations, and hence they lack robustness.

Global methods avoid this problem by exploiting more of the image evidence and optimizing a global objective function, for example by using Markov Chain Monte Carlo (MCMC) algorithms (Fan, 2006; Sun et al, 2007). However, while such methods produce smooth tree components, they do not necessarily guarantee their spatial connectedness. Furthermore, they are computationally intensive, which limits their applicability to large datasets.

By contrast, methods that sample local maxima of the tubularity image and then connect these samples into a minimum spanning tree (MST) (Fischler and Heller, 1998; Gonzalez et al, 2008; Xie et al, 2010), guarantee connectivity. However, they usually result in many spurious branches and gaps. While pruning during post-processing can eliminate some of the erroneous branches, it does not allow for recovery from the other mistakes. To resolve some of these mistakes, in (Xie et al, 2010), a gap indicator function is incorporated in the edge weights. However, this approach can easily fail in the presence of noisy data where the branches appear as disconnected segments.

Furthermore, MST-based approaches usually do not take into account global tree geometry, such as smoothness along the edges or branching factors, which can play an important role in improving topological accuracy, avoiding over-fitting, and speeding up convergence. Finally, they do not explicitly account for junction points such as bifurcations and crossovers, which can easily lead to erroneous connections, as will be shown later.

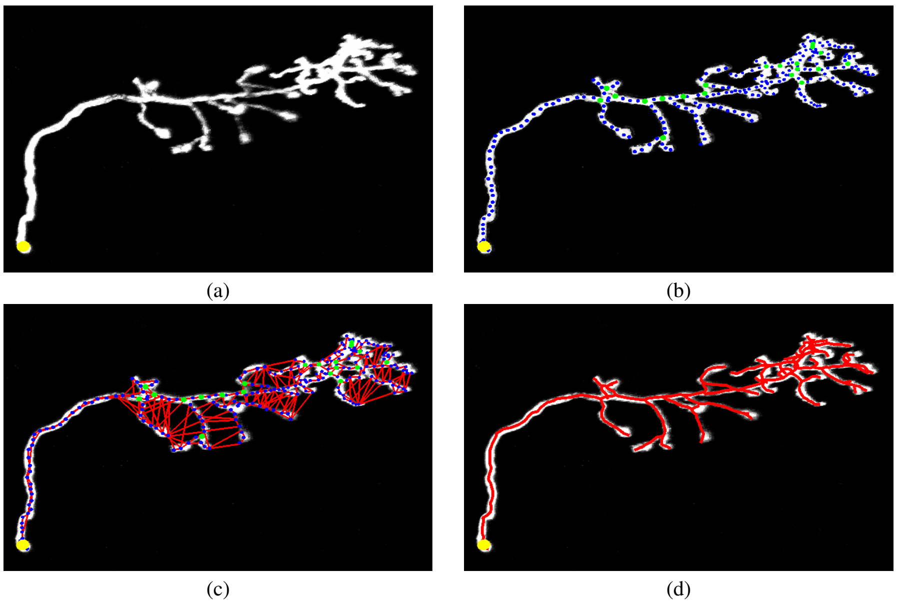


Fig. 2: The tree construction process. (a) Maximum intensity projection of an image stack representing olfactory projection fibers. The root node of the tree is manually supplied and is depicted by the yellow sphere. (b) Maximum intensity projection of the tubularity values computed for individual voxels. Anchor points obtained as local maxima of this measure are overlaid in green and blue. The green dots denote bifurcation points and the blue dots denote sampled points on the fibers' centerline. (c) The graph obtained by linking all anchor points to their neighbors. The edges depicted in red are illustrative and do not represent the actual paths. (d) The k-MST that minimizes our global objective function.

We address these problems by optimizing a global objective function that explicitly models spurious branches and incorporates geometric priors that capture geometric relationships between pairs of tree vertices and edges. These priors are of the same nature as those that are used in the object detection literature (Felzenszwalb and Huttenlocher, 2005; Leordeanu et al, 2007) to express dependencies between object parts.

3 Approach

In this section, we briefly outline our approach to delineation, which is depicted by Fig. 2. Our algorithm goes through the following steps:

1. We compute a *tubularity* value at each voxel, as shown in Fig. 2(b). It encodes how likely it is to be on the centerline of a tubular structure.
2. We select high-probability voxels that are as evenly spaced as possible and that we treat as anchor points, as shown in Fig. 2(b).
3. We compute the most probable paths between pairs of nearby anchor points and assign them probabilistic costs that are lowest when all voxels along them are likely to lie in the middle of a filament. This results in the graph of Fig. 2(c), where vertices are represented by the anchor points and edges by the paths linking them.

4. We compute the lowest-cost tree in this graph among those that span k of the edges for a wide range of $k < N$. This is known as the *k-Minimum Spanning Tree* (k-MST) or *k-Cardinality Tree* (KCT) problem. Even though it is NP-hard, approximate solutions can nevertheless be computed efficiently and fast (Blum and Blesa, 2005).
5. We select the tree depicted by Fig. 2(d) that maximizes a global objective function.

Steps 4 and 5 are those that most distinguish our approach from more traditional ones that either build trees spanning all the anchor points and then attempt to eliminate spurious branches, or grow the tree incrementally at the risk of propagating errors. We avoid these problems by minimizing a well-defined global objective function, as the minimum spanning tree approaches do, but with the possibility to explore different topologies because we do not force the tree to systematically connect all vertices.

For clarity's sake, we describe our approach in terms of finding dendritic and axonal trees in 3D image stacks. Note, however, that it also applies to linear structures in regular 2D images, such as the retinal scans of Fig. 1. One only has to replace the voxels by pixels and 26-connectivity by 8-connectivity.

In the remainder of this paper, we first describe in more details our approach to building graphs such as the one of Fig. 2(c). We then formulate the objective function we use to assess the quality of a tree within this graph and show how it can be optimized to produce a tree reconstruction such as the one of Fig. 2(d). Finally, we present results on real 2D and 3D images acquired using different modalities and discuss them.

4 Graph Construction

As discussed above, we begin by computing a graph, whose nodes are anchor points that are likely to lie at the center of filaments, and whose edges represent paths connecting them. In this section, we discuss the three steps involved in building this graph. We first introduce the tubularity measure we use to assess the probability that a voxel lies on a filament. We then discuss the sampling procedure we implemented to select regularly spaced anchor points by using the voxel probabilities. Finally, we describe our approach to linking them to create the edges of the graph.

4.1 Tubularity Measure

Elongated structures such as those of Fig. 1 can be found at many different scales and their appearance is often severely affected by the point spread function of the microscope, acquisition noise, and irregularities in the staining process. As a result, they only rarely appear as well-defined tubes, especially when they are very thin.

To achieve robustness to such distortions, we rely on statistical machine learning to learn the appearance of processes given the specific acquisition modality we are dealing with. More specifically, for each voxel, we form a set of feature vectors each of which is made up of steerable filter responses (Gonzalez et al, 2009) and Hessian eigenvalues at a particular scale and an orientation. We then train a Support Vector Machine classifier (SVM) (Schoelkopf et al, 1999) with Gaussian kernel on training data that consists of hand-supplied delineations. To this end, we collect voxels along these delineations as positive samples and randomly sample ones away from them as negative samples. At run-time, for a voxel x_i, we run the SVM for several possible widths and orientations. For each pair of

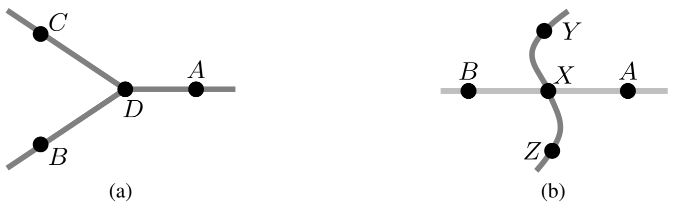


Fig. 3: Two cases that can lead to reconstruction errors. (a) If there is no sample point at bifurcation D, the tree reconstruction algorithm is likely to incorporate both the $\overline{AB}$ and $\overline{AC}$ paths into the final tree, which will result in counting twice the pixels in $\overline{AD}$ when scoring it. This is avoided by introducing an anchor point at D. (b) At a crossover, an anchor point can be used to build either the horizontal $\overline{AXB}$ branch or the vertical $\overline{YXZ}$ one, but not both. This is why we choose to create two anchor points at all potential crossovers.

width w and orientation ϕ, the SVM returns a score $f(x_i, w, \phi)$. We take our tubularity measure to be

$$f_i = \max_{w,\phi} f(x_i, w, \phi) \tag{1}$$

and retain the corresponding maximizing values w_i and ϕ_i as local width and orientation estimates. We use a sigmoid to map this measure to the posterior probability p_i of x_i being on the centerline of a linear structure given the score f_i. We write

$$p_i = \frac{1}{1 + e^{-(af_i + b)}} \ , \tag{2}$$

where the parameters a and b are estimated by cross-validation over a validation set. Using a sigmoid to convert an SVM output into a probability is valid because it preserves the sparseness of the SVM while returning probabilities comparable to those produced by regularized likelihood kernel methods (Platt, 2000).

4.2 Sampling the Image

To extract a representative set of anchor points that approximate well the underlying tree structure, samples should be taken from both junction points and tubular segments of the tree. We have therefore developed a two-step approach to finding anchor points introduced at the beginning of Section 3. We first attempt to detect junctions such as the green dots of Fig. 2(b) and then to find regularly spaced anchor points such as the blue dots away from them. In this way, if a junction is missed, it can be recovered when the algorithm tries to link blue dots. Furthermore, even if all the junctions were found, the blue dots would still be required to properly link distant ones.

Recall that we have assigned the tubularity probability p_i of Eq. 2 to each voxel. It is the voxel's probability of being on the centerline of a tubular structure. To detect junctions, we use a 0.5 threshold to binarize the resulting probability image and compute its skeleton (Lee et al, 1994). We label as potential junction voxels from which more than two skeletal branches emanate. Note that this can result either from a legitimate bifurcation or from a *crossover*, which can occur whenever two independent filaments approach each other at a distance smaller than the spatial resolution of the microscope. Because we cannot know at this stage whether a detected point is a bifurcation or a crossover, we generate two co-located anchor points for each such point to avoid problems such as those depicted by Fig. 3 at tree reconstruction time.

This being done, we remove from further consideration spatial neighborhoods of the detected junction points and sort the remaining voxels according to their tubularity probability. We then iteratively select the most probable one and eliminate all those within the neighborhood until the whole image has been explored. The neighborhood of voxel x_i is taken to be a cylindrical box whose axis is aligned with the orientation estimate ϕ_i. While the cylinder height is fixed to favor even sampling along filaments, the radius is taken to be linearly proportional to the width estimate w_i.

This results in a set of anchor points that are relatively regularly spaced along filaments. Inevitably, some of these points are false positives that do not lie on filaments and will have to be ignored when building the final tree.

4.3 Linking the Anchor Points

Let V be the set of anchor points, which is a subset of $V^I = \{x_i\}$, the set of all voxels in the image volume. We represent this volume by a directed graph $G^I = (V^I, E^I)$ whose vertex set is V^I and whose edges $E^I = \{e^I_{ij}\}$ connect each voxel to its 26 neighbors. Hereafter, G^I will be referred to as the *image graph*.

We construct a *reduced graph* $G = (V, E)$ of G^I over the set of anchor points V by linking all pairs of them, except the co-located ones, that are within a specified distance of each other. The edge set E corresponds to paths formed by successive edges of E^I connecting nearby anchor points. In the remainder of the paper, we will use the terms voxels and vertices of G as well as edges and paths interchangeably.

Formally, the path $e_{mn} \in E$ linking anchor points x_m , $x_n \in V$ can be represented by the set of edges $\{e^I_{mi}, e^I_{ij}, \ldots, e^I_{kn}\}$ it includes. In practice, we choose these edges by running Dijkstra's algorithm to minimize

$$d_{mn} = \sum_{e^I_{ij} \in e_{mn}} - \log p_{ij} \ , \tag{3}$$

where p_{ij} is an image-based probability that edge e^I_{ij} belongs to the centerline of a filament. In other words, we take e_{mn} to be the maximum likelihood path based on image evidence only and assuming that, in the absence of any such evidence, all paths are *a priori* equally likely.

We derive p_{ij} from the p_i values of Eq. 2 by considering the d_{mn} cost of Eq. 3. We require that d_{mn} should be roughly equal to the integral of $-\log(p_i)$ along the corresponding continuous path. In other words, we have

$$\begin{aligned} d_{mn} &\approx \int - \log p(s) ds \\ &\approx \sum_{e^I_{ij} \in e_{mn}} \int_0^{l_{ij}} - \log p(\frac{l_{ij} - s}{l_{ij}} x_i + \frac{s}{l_{ij}} x_j) ds \ , \end{aligned} \tag{4}$$

where s represents the curvilinear abscissa along the path, $p(s)$ the probability that the point at abscissa s is on a centerline, and l_{ij} the distance between neighboring points x_i and x_j. Since we work on a voxel grid, to compute the integral of Eq. 4, we only have values p_i and

p_j of $p((1-s/l_{ij})x_i+(s/l_{ij})x_j)$ for $s=0$ and $s=l_{ij}$ respectively. Assuming that p varies linearly between x_i and x_j, we write

$$
\begin{aligned}
d_{mn} &\approx \sum_{e^I_{ij}\in e_{mn}} \int_0^{l_{ij}} -\log\left(\frac{l_{ij}-s}{l_{ij}}p_i+\frac{s}{l_{ij}}p_j\right)ds \\
&\approx \sum_{e^I_{ij}\in e_{mn}} -l_{ij}\frac{p_i(\log(p_i)-1)+p_j(1-\log(p_j))}{p_i-p_j}.
\end{aligned} \tag{5}
$$

In practice, to avoid divisions by zero, we therefore take p_{ij} to be equal to $p_i^{l_{ij}}$ if $|p_i-p_j|\leq \epsilon$, and so that

$$
\log(p_{ij}) = l_{ij}\frac{p_i(\log(p_i)-1)+p_j(1-\log(p_j))}{p_i-p_j} \quad , \tag{6}
$$

otherwise. Note that this is consistent because when p_j-p_i tends towards zero, $\log(p_{ij})$ defined in this manner tends towards $l_{ij}\log(p_i)=l_{ij}\log(p_j)$.

5 Tree Reconstruction

Given the graph $G=(V,E)$ introduced in the previous section, we take trees to be directed subgraphs of G with a distinguished vertex, called the root, with in-degree 0 and such that there is a unique directed path from it to every other vertex in the tree.

Let $\mathcal{T}(G)$ be the set of all such trees in G. Our task now is to find the tree $\mathbf{t}^*\in\mathcal{T}(G)$ that best represents the underlying tree structure. To this end, we first introduce a Bayesian framework that lets us define an objective function to assess the quality of a tree $\mathbf{t}\in\mathcal{T}(G)$, using both image- and geometry-based evidence. Then, since $\mathcal{T}(G)$ is exceedingly large, we introduce two different algorithms designed to find acceptable approximations of the true optimum sufficiently fast to be of practical use.

5.1 Bayesian Formulation

A subgraph $\mathbf{t}$ of G can be represented by a set of indicator variables $\{t_{mn}\}$, one for each edge of G, such that

$$
\forall e_{mn}\in E,\quad t_{mn}=\begin{cases}1, & \text{if}\quad e_{mn}\in\mathbf{t}\\ 0, & \text{otherwise.}\end{cases} \tag{7}
$$

Let $\mathbf{T}=\{T_{mn}\}$ be the set of binary random variables such that T_{mn} stands for edge e_{mn} truly being on the centerline of a tree structure. Similarly, let T^I_{ij} stand for the random variable representing the presence or absence of a centerline along the edge e^I_{ij} of the image graph.

Recall from Section 4.1 that we associate to each voxel x_i the tubularity measure f_i of Eq. 1, which can be used to evaluate how likely x_i is to be on the centerline of a tubular structure. We also estimate the width w_i and the orientation ϕ_i of the tube, assuming there is one. Let $\mathbf{f}$ denote the set of tubularity measures and $\mathbf{\Phi}$ the set of all widths and orientations.

We take the optimal subgraph $\mathbf{t}^*$ to be the tree whose likelihood is greatest, given the image features $\mathbf{f}$ and the orientation and width estimates $\mathbf{\Phi}$. This can be written as

$$\begin{aligned}
\mathbf{t}^* &= \underset{\mathbf{t}\in\mathcal{T}(G)}{\operatorname{argmax}} P(\mathbf{T}=\mathbf{t}|\mathbf{f},\mathbf{\Phi},\mathbf{\Theta}) && (8)\\
&= \underset{\mathbf{t}\in\mathcal{T}(G)}{\operatorname{argmax}} P(\mathbf{f}|\mathbf{T}=\mathbf{t})P(\mathbf{T}=\mathbf{t}|\mathbf{\Phi},\mathbf{\Theta}) && (9)\\
&= \underset{\mathbf{t}\in\mathcal{T}(G)}{\operatorname{argmin}} -\log P(\mathbf{f}|\mathbf{T}=\mathbf{t}) - \log P(\mathbf{T}=\mathbf{t}|\mathbf{\Phi},\mathbf{\Theta}), && (10)
\end{aligned}$$

where $\mathbf{\Theta}$ denotes a set of learned meta-parameters encoding prior knowledge about plausible tree shapes. Eq. 9 follows from Eq. 8 because the tubularity measures $\mathbf{f}$ of Eq. 1 only denote the presence or the absence of centerlines and are therefore conditionally independent of the widths and orientations $\mathbf{\Phi}$, given the presence or absence of a tubular structure. We now turn to defining more precisely the two terms of Eq. 10, which we will refer to as the Image-Based and the Geometry-Based terms respectively.

5.1.1 Image-Based Term

$P(\mathbf{f}|\mathbf{T}=\mathbf{t})$ represents the probability of observing the image features we extract, knowing where the tree is. We assume conditional independence of these features along neighboring edges given that we actually know whether these edges belong to the tree or not, and hence, represent the likelihood as a product of individual edge likelihoods. Since our eventual goal is to maximize the first term of Eq. 9 with respect to the t_{mn} indicator variables of Eq. 7, in the following derivation, we drop all terms that are independent from the t_{mn} and have therefore no bearing on the outcome of the computation.

Recall from Section 4.3 that each edge $e_{mn} \in E$ is composed by a set of edges $\{e^I_{ij}\} \in E^I$ belonging to the image graph and linking anchor points $x_m, x_n \in V$. We therefore write

$$\begin{aligned}
P(\mathbf{f}|\mathbf{T}=\mathbf{t}) &= \prod_{e_{mn}\in E} P(\mathbf{f}_{mn}|T_{mn}=t_{mn}) && (11)\\
&= \prod_{e_{mn}\in E} P(\mathbf{f}_{mn}|T_{mn}=1)^{t_{mn}} \times P(\mathbf{f}_{mn}|T_{mn}=0)^{(1-t_{mn})} && (12)\\
&\propto \prod_{e_{mn}\in E} \left[\frac{P(\mathbf{f}_{mn}|T_{mn}=1)}{P(\mathbf{f}_{mn}|T_{mn}=0)}\right]^{t_{mn}} && (13)\\
&= \prod_{e_{mn}\in E} \left[\prod_{e^I_{ij}\in e_{mn}} \frac{P(f_{ij}|T^I_{ij}=1)}{P(f_{ij}|T^I_{ij}=0)}\right]^{t_{mn}} && (14)\\
&= \prod_{e_{mn}\in E} \left[\prod_{e^I_{ij}\in e_{mn}} \frac{P(T^I_{ij}=1|f_{ij})P(T^I_{ij}=0)}{P(T^I_{ij}=0|f_{ij})P(T^I_{ij}=1)}\right]^{t_{mn}} && (15)\\
&\propto \prod_{e_{mn}\in E} \left[\prod_{e^I_{ij}\in e_{mn}} \frac{P(T^I_{ij}=1|f_{ij})}{P(T^I_{ij}=0|f_{ij})}\right]^{t_{mn}} && (16)\\
&= \prod_{e_{mn}\in E} \left[\prod_{e^I_{ij}\in e_{mn}} \frac{p_{ij}}{1-p_{ij}}\right]^{t_{mn}}, && (17)
\end{aligned}$$

where $\mathbf{f}_{mn}$ is the set of tubularity values along the path e_{mn} and f_{ij} stands for the tubularity values f_i and f_j for the vertices of the edge $e^I_{ij} \in E^I$ of the image graph.

Eqs. 12 and 13 are respectively obtained by a simple algebraic manipulation and dropping a constant term. In Eq. 14, we use the conditional independence assumption of the tubularity measures along neighboring edges of the image graph given the true state of the edges. In Eqs. 15 and 16, we use Bayes' rule and assume that edges are *a priori* equally likely to belong to a tree, which lets us also drop the $P(T^I_{ij} = 0)$ and $P(T^I_{ij} = 1)$ terms. Finally, in Eq. 17, we replaced the probability $P(T^I_{ij} = 1|f_{ij})$, the likelihood that the edge e^I_{ij} is on a filament centerline, by its short notation p_{ij} introduced in Eq. 3 and computed according to the formula given in Eq. 6. Likewise, we replaced $P(T^I_{ij} = 0|f_{ij})$ by $1 - p_{ij}$.

For a tree $\mathbf{t}$ in G, the image-based term of Eq. 17 can therefore be written as

$$F_i(\mathbf{t}) = -\log P(\mathbf{f}|\mathbf{T} = \mathbf{t}) = \sum_{e_{mn} \in E} c_{mn}\, t_{mn}, \qquad (18)$$
$$\text{where } c_{mn} = \sum_{e^I_{ij} \in e_{mn}} -\log \frac{p_{ij}}{1 - p_{ij}}.$$

In essence, c_{mn} can be considered as the cost of edge e_{mn} and our algorithm will try to minimize the sum of these costs over the whole tree, while also enforcing the geometric constraints discussed in the following section.

5.1.2 Introducing the Geometry-Based Term

The simplest possible approach to estimating the geometry-based term, $-\log P(\mathbf{T} = \mathbf{t}|\boldsymbol{\Phi}, \boldsymbol{\Theta})$, is to assume that all trees have the same prior probability and that subgraphs that are not trees have probability zero, which makes it constant for all choices of t_{mn} values that result in a tree. Under this assumption, the optimal tree $\mathbf{t}^*$ of Eq. 10 can be obtained by finding the set of indicator variables $\{t^*_{mn}\}$ such that the corresponding subgraph is a tree and the image-based linear objective function of Eq. 18 is minimized.

However, as shown in Fig. 1, this can result in erroneous topologies in ambiguous cases, mostly because the image data is very noisy. To remedy this, we incorporate geometric priors into our objective function to penalize trees whose geometric properties make them unlikely candidates.

More specifically, we model the prior term as a tree structured Bayesian network that captures geometric relationships between consecutive edge and vertex pairs. In this work, we assume that the root vertex, $x_r \in V$, of the tree structure is either given or can be reliably estimated. Let $E_r \subset E$ be the set of edges emanating from x_r. We take the prior probability $P(\mathbf{T} = \mathbf{t}|\boldsymbol{\Phi}, \boldsymbol{\Theta})$ to be

$$\prod_{e_{ri} \in E_r} P(T_{ri} = 1|\boldsymbol{\Phi}_r, \boldsymbol{\Phi}_i, \boldsymbol{\Theta})^{t_{ri}} \times \prod_{\substack{e_{om} \in E \\ e_{mn} \in E \setminus E_r}} P(T_{mn} = 1|T_{om} = 1, \boldsymbol{\Phi}_{omn}, \boldsymbol{\Theta})^{t_{mn} t_{om}},$$

where $\boldsymbol{\Phi}_i$ denotes the width and orientation estimates for vertex x_i, and $\boldsymbol{\Phi}_{omn}$ denotes width and orientation estimates for vertex triplets. Under this model, the geometry-based term becomes the quadratic function of the indicator variables

$$-\log P(\mathbf{T} = \mathbf{t}|\boldsymbol{\Phi}, \boldsymbol{\Theta}) = \sum_{e_{ri} \in E_r} b_{ri}\; t_{ri} \; + \sum_{\substack{e_{om} \in E \\ e_{mn} \in E \setminus E_r}} a_{omn} t_{mn} t_{om} \;\; , \qquad (19)$$

where

$$\begin{aligned} b_{ri} &= -\log P(T_{ri} = 1 | \mathbf{\Phi}_r, \mathbf{\Phi}_i, \mathbf{\Theta}) \ , \\ a_{omn} &= -\log P(T_{mn} = 1 | T_{om} = 1, \mathbf{\Phi}_{omn}, \mathbf{\Theta}). \end{aligned} \tag{20}$$

These terms encode the fact that successive edges and vertices must have consistent widths and orientations. Specific choices for modeling them will be given in the next section.

The objective function $F_g(\mathbf{t})$ that must be minimized to find the optimal tree $\mathbf{t}^*$ of Eq. 10 can now be obtained by summing the geometry-based term of Eq. 19 to the image-based one of Eq. 18. This yields

$$F_g(\mathbf{t}) = \sum_{e_{mn} \in E} c_{mn} t_{mn} + \sum_{e_{ri} \in E_r} b_{ri} t_{ri} + \sum_{\substack{e_{om} \in E, \\ e_{mn} \in E \setminus E_r}} a_{omn} t_{mn} t_{om} \tag{21}$$

$$= \sum_{e_{ri} \in E_r} (c_{ri} + b_{ri}) t_{ri} + \sum_{\substack{e_{om} \in E, \\ e_{mn} \in E \setminus E_r}} (c_{mn} + a_{omn}) t_{mn} t_{om} \ . \tag{22}$$

Eq. 22 follows from Eq. 21 by rewriting the image-based term as a sum of unary and binary terms involving the indicator variables under the assumption that $\mathbf{t}$ is a tree, and thus has no cycles, and then grouping them with those that appear in the geometry-based term. Note that the unary term represents the total cost of the edges emanating from the root vertex and the binary term represents the cost for all edge pairs in $\mathbf{t}$. Note also that the c_{mn} and b_{ri} terms are edge weights while the a_{omn} terms are pairwise edge weights.

The above optimization problem generalizes the *Minimum Arborescence Problem* (Duhamel et al, 2008) with a quadratic cost and additional constraints, which is NP-hard. The presence of the quadratic terms makes the minimization of $F_g(\mathbf{t})$ more difficult than that of the linear objective function $F_i(\mathbf{t})$ of Eq. 18, which only involves unary terms. While it turned out to be possible to optimize the latter by directly using our earlier algorithm (Blum and Blesa, 2005), we had to extend it substantially to handle the former. This will be discussed in more details in Section 5.2.

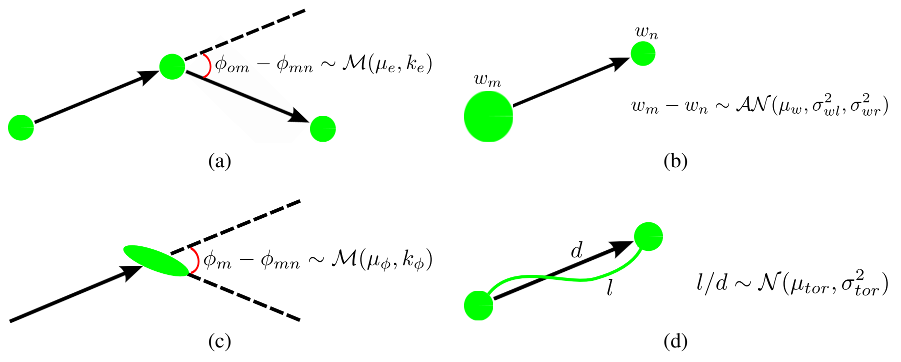
Even though the above formulation assumes a single tree with a well defined root, it is easily extended to the multiple tree reconstruction case. This is done by adding a virtual vertex to the graph and connecting it to the individual root vertices of the trees to be reconstructed by minimal cost outgoing edges. Likewise, the pairwise costs of edge pairs directly connected to the root vertices are set to minimal values. This transforms the multiple tree reconstruction problem into a single-tree reconstruction one with no loss of generality.

For multiple trees, as in the single tree case, we assume that the root vertices of each tree is either given *a priori* or can be reliably estimated. This is required because the global objective function of Eq. 22 does not score one single tree at a time, as is done in most tracking-based methods. Instead, the tree reconstruction algorithm discussed below, which is designed to minimize it, seeks to explain the whole image volume at once.

5.1.3 Modeling the Geometry-Based Term

In this work, we exploit four geometric properties to capture the underlying relations between parts of the tree structure, as illustrated by Fig. 4. They are:

(a) **Edge Direction Similarity.** To obtain smooth reconstructions, we encode direction similarity of pairs of consecutive edges. The angular difference between the directions is modeled by a von Mises distribution, that is a circular normal distribution of mean μ_e and concentration k_e.



$\phi_{om} - \phi_{mn} \sim \mathcal{M}(\mu_e, k_e)$
(a)
w_n
w_m
$w_m - w_n \sim \mathcal{AN}(\mu_w, \sigma_{wl}^2, \sigma_{wr}^2)$
(b)
$\phi_m - \phi_{mn} \sim \mathcal{M}(\mu_\phi, k_\phi)$
(c)
d
l
$l/d \sim \mathcal{N}(\mu_{tor}, \sigma_{tor}^2)$
(d)

Fig. 4: The four components of the geometric regularization term. The green circles are the vertices of the reduced graph and the solid lines illustrate the edges between them. (a) Edge direction similarity. (b) Width consistency. (c) Orientation consistency. (d) Tortuosity.

(b) **Width Consistency.** We model the width differences at pairs of consecutive vertices by an asymmetric Gaussian distribution of mean μ_w and variances σ_{wl}^2 and σ_{wr}^2. This model accommodates the fact that the width may tend to decrease with distance from the root in some datasets but not others.
(c) **Orientation Consistency.** We measure the angular deviation of the estimated orientations of two consecutive vertices from the direction of the line between them. This deviation is again modeled by a von Mises distribution of mean μ_ϕ and concentration k_ϕ.
(d) **Tortuosity,** We use the ratio of the path length to the linear distance between the endpoints as measure of tortuosity, which we then represent by a Gaussian distribution of mean μ_{tor} and variance σ_{tor}^2.

Given these geometric terms, the vector $\mathbf{\Theta}$ of meta-parameters introduced in Eq. 10 is composed of the means, variances, and concentrations of these four distributions, which we estimate using a maximum likelihood approach on a training data set. We therefore write the geometry-based probability of a pair of consecutive edges $e_{om} \in E$ and $e_{mn} \in E$ belonging to the tree as

$$\begin{aligned} P(T_{mn} = 1 | T_{om} = 1, \mathbf{\Phi}_{omn}, \mathbf{\Theta}) &= \mathcal{M}(\phi_{om} - \phi_{mn}, \mu_e, k_e) \\ &\times \mathcal{AN}(w_m - w_n, \mu_w, \sigma_{wl}^2, \sigma_{wr}^2) \\ &\times \mathcal{M}(\phi_m - \phi_{mn}, \mu_\phi, k_\phi) \\ &\times \mathcal{M}(\phi_n - \phi_{mn}, \mu_\phi, k_\phi) \\ &\times \mathcal{N}(tor(e_{mn}), \mu_{tor}, \sigma_{tor}^2) \ , \end{aligned} \tag{23}$$

where ϕ_{om} denotes the direction angle of the line between vertices x_o and x_m, and $tor(e_{mn})$ denotes the tortuosity value assigned to the path corresponding to edge e_{mn}. The prior probability $P(T_{ri} = 1 | \mathbf{\Phi}_r, \mathbf{\Phi}_i, \mathbf{\Theta})$ of Eq. 20 for edges emanating from the root vertex includes the same set of geometric terms except the edge direction similarity term.

5.2 Optimization

Let us assume that the graph G has N vertices. In the first part of this section, we present a simple approach that aims at building the best possible tree, $\mathbf{t}_k$, among all those that span exactly k edges. We run it for all k from 1 to $N-1$ and pick the $\mathbf{t}_k$ that yields the best overall score. This is a simple but practical way to optimize the criterion of Eq. 18 because, for each k, we can take advantage of our earlier k-MST algorithm (Blum and Blesa, 2005) to build a near-optimal k cardinality tree. However, this algorithm was not designed to handle pairwise edge-relationships, such as those that appear in the objective function of Eq. 22. Furthermore, this approach is computationally inefficient because it involves restarting the whole computation from scratch for each successive cardinality. In the second part of this section, we therefore extend the original k-MST algorithm both to handle geometric relationships between edges and to compute not one single tree but many, one for each cardinality, at once.

5.2.1 Optimizing the Image-Based Term

To minimize the image-based objective function F_i of Eq. 18, we begin by using the k-MST algorithm (Blum and Blesa, 2005) for each value of k ($0 < k < N$) to build the k-cardinality tree $\mathbf{t}_k$ that minimizes

$$\sum_{e_{mn} \in E} d_{mn} t_{mn} \ , \tag{24}$$

where the d_{mn} are the sum of negative log likelihoods of Eq. 3. We then simply select the k and corresponding $\mathbf{t}_k$ that yields the lowest value of $F_i(\mathbf{t}_k)$.

Note that it might have seemed more logical to use the sums of log likelihood ratios c_{mn} of Eq. 18 in Eq. 24 so that the k-MST algorithm could directly build k-cardinality trees that are optimal for the true objective function we seek to minimize. However, consider that the k-MST algorithm makes local decisions to build candidate trees, which involves selecting a new edge at each iteration to decrease the overall tree cost. If the edges were weighted using the log-likelihood ratios of Eq. 18, the preferred edge would always be the one accounting for the greatest amount of image evidence, thus ignoring the fact that there are more edges to be added. This would bias the optimization towards long edges and would produce undesirable effects, such as those depicted by Fig. 5(a). The same argument is made very convincingly, albeit in a different context, in Section 5 of (Felzenszwalb and McAllester, 2006). In contrast, using the log-likelihood of Eq. 24 as edge weights results in the highest density of probability tree being built. Since we assume that there exists only one tree per image, once the tree $\mathbf{t}_k$ is constructed we consider all remaining edges as background, and assign to the tree the score of Eq. 18.

The k-MST algorithm (Blum and Blesa, 2005) relies on an ant colony optimization (ACO) scheme (Dorigo and Stütale, 2004) inspired by the foraging behavior of real ants. While walking from food sources to the nest and vice versa, ants deposit chemicals known as pheromones on the ground. Paths marked by strong pheromone concentrations are more likely to be chosen when deciding in what direction to go. This group behavior is the basis for a cooperative interaction which lets the ants find shortest paths between their nest and food sources. In ACO algorithms, an artificial ant incrementally constructs a complete solution by iteratively adding appropriately chosen components to a partial one. The solution components to be added are chosen probabilistically according to a parameterized probabilistic model, the so-called *pheromone model* $\mathcal{P}$, which is a finite set of numerical values.

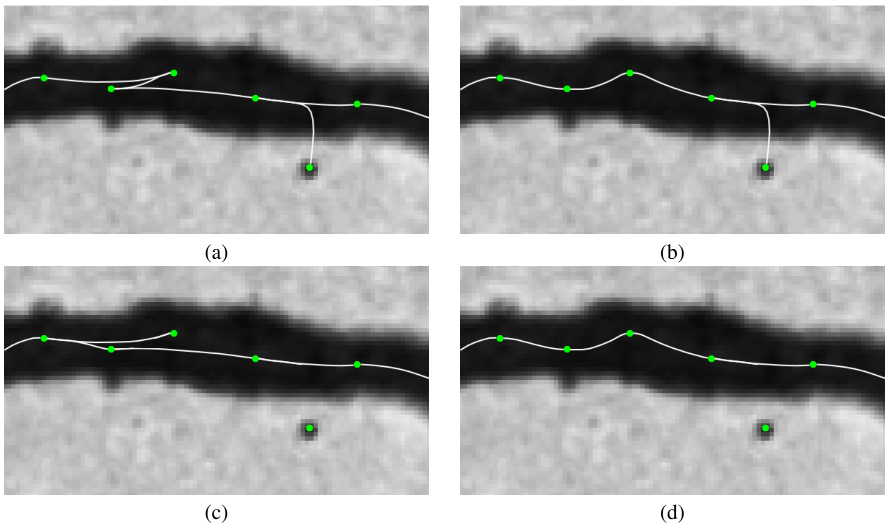


Fig. 5: Different approaches to linking a set of anchor points. (a) Minimizing the log-likelihood ratio cost function of Eq. 18 using the ACO algorithm of Section 5.2.1 favors long paths between anchor points, which results in zig-zags and some pixels being used twice. (b) Replacing the log-likelihood ratios by the log likelihoods of Eq. 24 removes the zig-zags but does not prevent double-counting, resulting in the linking of a false-positive and a spurious branch. (c) Introducing the geometric priors to the log-likelihood ratios of Eq. 25 also removes the zig-zags but still does not prevent double-counting and the creation of a different spurious branch. (d) Using the log likelihoods and the geometric priors as in Eq. 26 produces the right answer.

Each *pheromone value* $\tau_i \in \mathcal{P}$ is associated to an element from a set of potential solution components. The pheromone model is used to probabilistically generate complete solutions by assembling them from a set of solution components. In general, an ACO algorithm repeatedly goes through the two following steps:

1. Candidate solutions are constructed using a pheromone model, that is, a parameterized probability distribution over the solution space;
2. The candidate solutions are used to modify the pheromone values so as to bias future sampling toward high quality solutions.

The second step, often referred to as the pheromone update, aims at focusing the search towards promising parts of the search space and is a critical component of all ACO algorithms. It implicitly assumes that good solutions consist of good solution components.

More specifically, the ACO algorithm for the k-MST problem in undirected graphs that we use here works roughly as follows. At each iteration, a number n_a of l-cardinality trees—where $l > k$—are probabilistically constructed based on the pheromone model and edge weights. The pheromone model consists of a pheromone value τ_e for each edge e of the given undirected graph. Then, a dynamic programming algorithm (Blum, 2007) is used to extract from each of these l-cardinality trees the best k-cardinality tree they contain. The last step of each iteration consists in the pheromone update, which—depending on the so-called convergence factor—uses a weighted average of good solutions found previously. When reaching the imposed computation time limit, the algorithm returns the best solution it has found up to this point.

5.2.2 Accounting for the Pairwise Terms

The approach discussed above cannot be used to optimize the full objective function of Eq. 22 because our original k-MST algorithm cannot handle quadratic terms. Furthermore, it is computationally inefficient because it involves running the algorithm many times over, once for each cardinality. We have therefore extended the k-MST algorithm in the following ways:

1. Since the root node is given and the pairwise terms a_{omn} of Eq. 20 are asymmetric, we replace the undirected graph model used by the original k-MST algorithm by a directed one and explicitly introduce the root vertex x_r.
2. We take into account the pairwise terms introduced in Section 5.1.2 when computing effective edge weights.
3. To improve the convergence of the algorithm, pheromone values are assigned to pairs of consecutive edges instead of to individual ones. In other words, the set $\mathcal{P}$ contains a pheromone value for each pair of consecutive edges.
4. We introduce an additional tree neighborhood structure and search over this neighborhood to avoid incorrect connections at crossovers and to improve convergence.
5. A branching factor limit constraint is imposed on the reconstructions.

In our implementation, not only is the quality of the trees subject to optimization, but also their cardinality, which is not given beforehand. This is achieved through an optimization scheme that iterates two complementary steps, one for tree construction and the other for cardinality selection. Recall from Section 5.1.2, that we seek to minimize the objective function

$$F_g(\mathbf{t}) = \sum_{e_{ri} \in E_r} (c_{ri} + b_{ri}) t_{ri} + \sum_{\substack{e_{om} \in E, \\ e_{mn} \in E \setminus E_r}} (c_{mn} + a_{omn}) t_{mn} t_{om} \ , \tag{25}$$

which we will refer to as the *primary objective function*. We also introduce an *auxiliary objective function*

$$F'_g(\mathbf{t}) = \sum_{e_{ri} \in E_r} (d_{ri} + b_{ri}) t_{ri} + \sum_{\substack{e_{om} \in E, \\ e_{mn} \in E \setminus E_r}} (d_{mn} + a_{omn}) t_{mn} t_{om} \tag{26}$$

where we replace the log likelihood ratios c_{mn} of Eq. 18 by the log likelihoods d_{mn} of Eq. 3.

In our optimization procedure, we use the d_{mn} costs in the auxiliary objective function as a search heuristic to construct the trees and the c_{mn} ones in the primary objective function to score them. We do this for the same reason as we did in the simpler version of the algorithm outlined in Section 5.2.1. This remains necessary because, even though introducing the geometric priors tends to fix the zigzagging behavior depicted by Fig. 5(a), as shown in Fig. 5(c), it does not prevent some pixels from being used twice, resulting in spurious branches such as the one depicted by Fig. 5(c). This behavior results from the fact that when we compute the edges of the graph, that is, the paths linking the anchor points, nothing prevents the same voxels from being used in more than one path. Preventing this would require adding many additional anchor points to guarantee that there is one for every potential junction and would be very computationally expensive. Furthermore, this would result in irregularly spaced anchor points and paths of arbitrary length, which would give additional and unwarranted influence to path length when selecting edges.

The first reconstruction step of the optimization involves first building a number n_a of trees using an ACO scheme similar to the one discussed in Section 5.2.1 that adds edges so as to probabilistically minimize the auxiliary objective function with the quadratic term. For each one of these trees, the algorithm then uses the same dynamic programming technique (Blum, 2007) as before to extract the best k-cardinality tree for $k = 1, \ldots, k_{\text{limit}}$. Finally, for each cardinality, among the extracted n_a trees, we keep the one that minimizes the auxiliary objective function. This results in k_{limit} trees each with a different cardinality.

In the second reconstruction step, among all the resulting trees, the one that minimizes the primary objective function is selected and the pheromone values are updated accordingly.

Henceforth, we will refer to this algorithm as the ACO-RTS method, which stands for Ant Colony Optimization for the Reconstruction of Tree-like Structures. In the appendix, we describe it in more details and provide a pseudo-code for it.

6 Results

In this section, we first use eight brightfield micrographs, such as those depicted in Figs. 1 and 6, to evaluate the various components of our approach and to demonstrate the importance of using both the image-based and the geometry-based terms of our objective function. To demonstrate the generality of our approach, we show a similar improvement on a very different dataset, the DRIVE database of retinal scans (Staal et al, 2004), such as those in Figs. 1 and 8. Finally, we present our results on the DIADEM challenge datasets (Ascoli et al, 2010). All images in this section contain overlays that are best viewed in color.

In all cases, we used the same algorithm without customization, besides retraining using ground truth data. This illustrates the ability of our statistical learning-based approach to adapt to a wide range of modalities.

Running the full algorithm requires setting 13 parameters in total. The a and b parameters of the sigmoid function of Eq. 2 are learned from the training datasets and associated ground truth tracings using the Levenberg-Marquardt algorithm of (Platt, 2000). The radius and height of the cylindrical box used for sampling and described in Section 4.2 are estimated from the training datasets using a two dimensional grid search procedure to maximize the DIADEM score. The remaining parameters correspond to the distributions of the geometric priors introduced in Section 5.1.3 and are estimated using a maximum likelihood approach.

The algorithm is implemented in C++ and parallelized across multiple CPU cores. The training-times are in the order of a few tens of minutes. The run-times range between few seconds for relatively small datasets such as the Olfactory Projection Fibers to few tens of minutes for the largest ones such as the Hippocampal CA3 Neurons on an eight core 3 GHz PC. The software is available online at http://cvlab.epfl.ch/research/medical/lm.

6.1 Brightfield Micrographs

We evaluated our algorithm on eight brightfield micrographs such as those of Figs. 1 and 6. They were acquired by our colleagues from EPFL's Brain Mind Institute, who also gave us hand-traced ground-truth data. The images were obtained from biocityne-dyed rat brains.

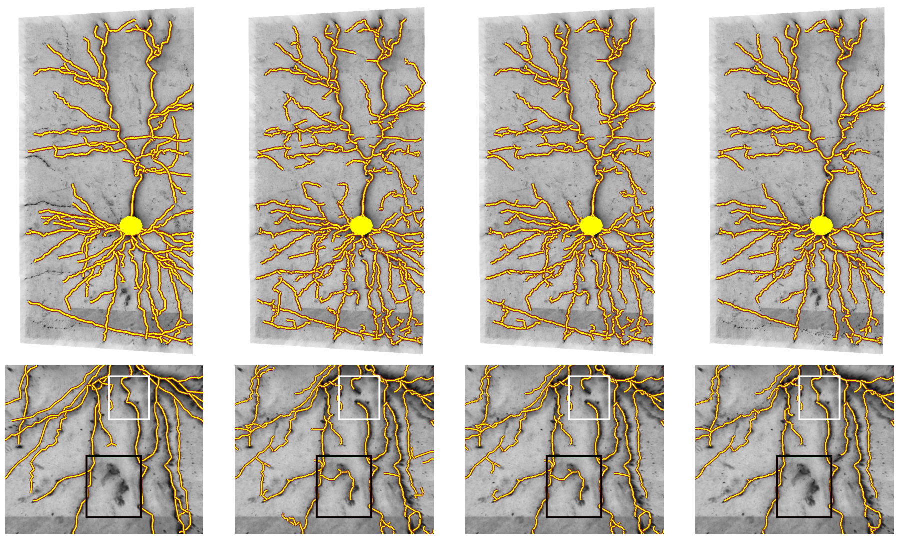
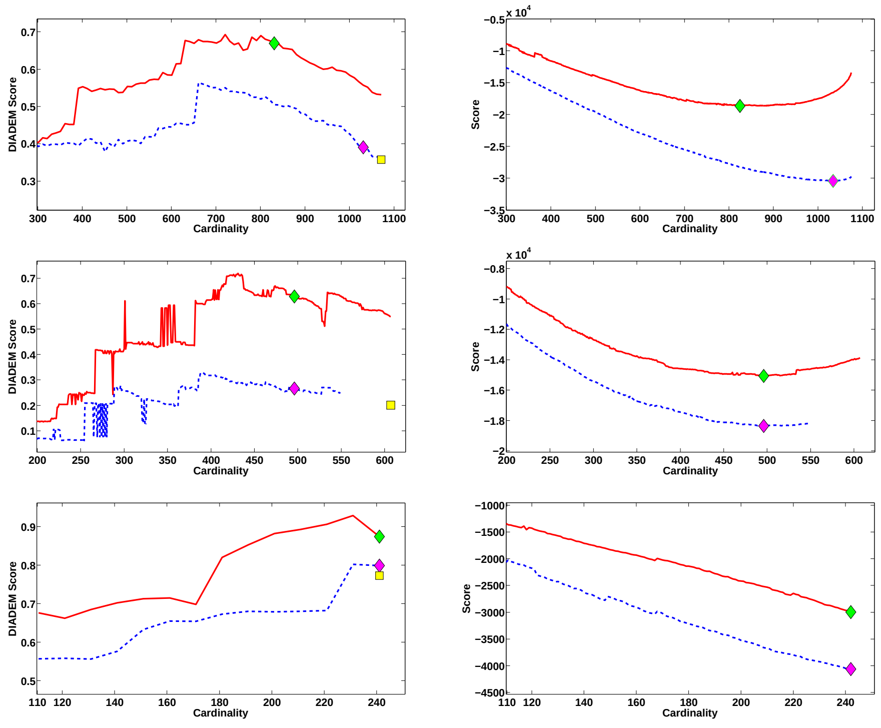


Fig. 6: Reconstructions in a brightfield micrograph. The top row depicts the whole image stack and the bottom one an enlarged portion of it. The yellow sphere denotes the soma and serves as the root node. (a) Manually delineated ground truth overlaid over the original stack. (b) Minimum spanning tree. (c) Reconstruction obtained without geometric regularization. (d) Reconstruction obtained with geometric regularization. The boxes overlaid on the bottom row images highlight locations where geometric regularization clearly brings about an improvement. In the area within the uppermost box, there seems to be a gap in the data. As a result, both the MST and the k-MST without regularization fail to connect the top and bottom part of the dendrite, which is wrong as can be seen in the ground truth data. By contrast, the regularized k-MST depicted by the last column exhibits the right topology. Moreover, in the area enclosed by the lowest of the boxes, there is structured noise, which is correctly ignored by the regularized k-MST but not by the others.

The numerous artifacts produced by irregularities of the staining process and the non-Gaussian blur introduced by the microscope make their automated analysis challenging. Many significant processes appear as faint structures, present abrupt intensity changes, or are severely blurred. Furthermore, the stain can dye irrelevant structures, such as blood vessels that are close to the neuron under analysis. This produces both structured and unstructured noise that is difficult to distinguish from true processes, even for a human expert.

As discussed in the implementation section, we use manually annotated trees to train the SVM classifiers of Section 4.1, which we use to evaluate tubularity, and to learn the geometric probability distributions of Section 5.1.3. Since the data is noisy, many spurious anchor points are generated during the sampling step of Section 4.2. Thus, the minimum spanning tree of Fig. 1(b) is over-complete. Applying our k-MST scheme results in the much cleaner trees of Fig. 1(c), when not using our geometric regularization term, and of Fig. 1(d), when using it.

In Fig. 6, we show similar results on a second micrograph. As shown in the areas highlighted by the boxes overlaid on the bottom row images, the geometric regularization term can bring very significant improvements both when there is a gap in the image data, by



(a) (b)

Fig. 7: Quantitative evaluation of our results for the brightfield stack of Fig. 6 (top row), the retinal scan of Fig. 8 (middle row) and the Olfactory Projection Fibers of Fig. 9 (bottom row). (a) DIADEM scores as a function of the tree cardinality. The red solid curve represents the scores obtained with the proposed geometric regularization, which means minimizing the objective function of Eq. 26 for each fixed cardinality, and the blue dotted curve represents those obtained without regularization. The yellow square indicates the score of the standard MST. (b) Corresponding values of the objective functions F_g of Eq. 22 and F_i of Eq. 18, also drawn as red solid and blue dotted lines respectively. The diamonds represent the selected cardinalities that minimize them.

bridging it, or when there is structured noise, by eliminating it. To quantify this improvement, we scored the reconstructions using the DIADEM metric (Ascoli et al, 2010), which is specifically designed to compare the topology of a reconstructed tree against ground truth. It returns a number between 0 and 1 that measures the topological distance between trees by matching their branching and end points, and analyzing the connecting paths. Each connection is weighted by the size of the subtree it is connected to. Therefore, errors close to the tree root are usually penalized more heavily than those further away because they result in more severe topological changes. The results shown in the top row of Fig. 7 confirm that trees reconstructed using our full approach score better than both the minimum spanning tree and those reconstructed without geometric regularization.

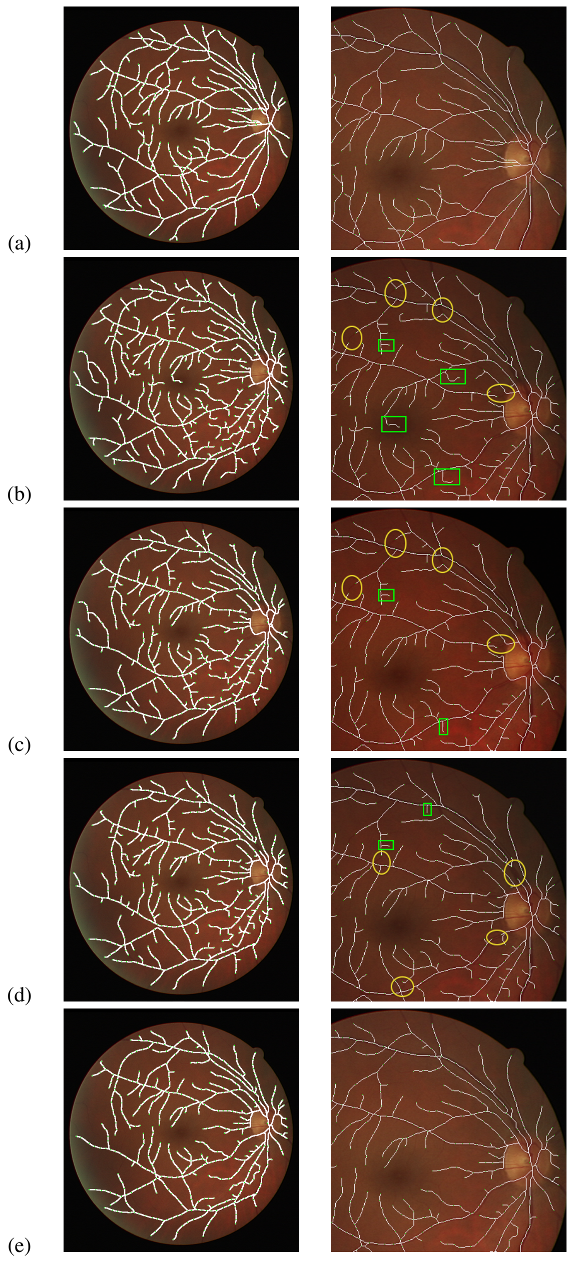


Fig. 8: Example result for a retinal image from the DRIVE database. The first column depicts the whole image and the second an enlarged portion of it. (a) Manually outlined blood vessels overlaid in white. (b) Minimum spanning tree. (c) Minimum spanning tree with the spurious branches trimmed by minimizing Eq. 18. (d) Reconstruction without geometric regularization. (e) Reconstruction with geometric regularization. The green boxes and the yellow ellipses denote spurious branches and gaps, all of which disappear in row (e).

6.2 Retinal Scans

To demonstrate the generality of our algorithm, we evaluated it on the 2D retinal images of the DRIVE database (Staal et al, 2004). The dataset contains 20 test images with manual segmentations of blood vessels performed by trained human observers. Since segmentations can not be used for quantitative topological comparison, we manually traced the vascular trees and treat them as ground truth. At detection time, the root vertex is automatically estimated to be the anchor point nearest to the center of the optic disk region, which is detected by using a variant of the method of (Huang and Yan, 2006).

Fig. 8 illustrates the behavior of our algorithm on one of these images[1]. Again, our approach eliminates many outliers and yields much cleaner trees than a standard minimum spanning tree, with far fewer mistakes such as those highlighted by the green rectangles and the yellow ellipses in the second column of the figure, especially when using the full objective function. Note that the final tree is not a subset of the minimum spanning tree. Its topology is actually different, which could not have been achieved by simply pruning the minimum spanning tree as shown in the third row of the figure. To quantify the corresponding improvement, we again computed the DIADEM scores. As shown in the middle row of Fig. 7, the results are similar to those we obtained for the brightfield micrographs. In terms of the metric, using the geometric regularizer improves the results by about 40% over not using it and by slightly more when comparing against the minimum spanning tree or the pruned version of it. Arguably, using the k-MST algorithm without geometric regularization eliminates many spurious branches but does not significantly improve the resulting topology. However, this is achieved using the full objective function.

6.3 DIADEM Data

We ran our algorithm on the five datasets provided by the DIADEM challenge (Ascoli et al, 2010). Each one includes training image stacks with ground-truth traces generated by experts and test stacks. As before, the ground truth is used to train the SVM classifiers of Section 4.1 and to learn the geometric probability distributions of Section 5.1.3.

6.3.1 Olfactory Projection Fibers

These image stacks are acquired from neurons of the olfactory bulb of the Drosophila fly. Each stack depicts a single neuron labeled with a Green Fluorescent Protein (GFP) and is imaged using a confocal microscope. The stacks are relatively clean, with a small point spread function. As can be seen in Fig. 9, the trees appear either as simple structures consisting of few branches or a path with almost no bifurcations that ends in a complex and a highly branched structure.

The reconstructions we produce are generally good as also shown by the DIADEM scores of the bottom row of Fig. 7. Note that the objective function F_g of Eq. 25 does not seem to reach to a minimum as shown in the bottom row of Fig. 7(b). Since this dataset is fairly clean with relatively little noise on the background, it turns out that the sampling does not produce any outliers outside the processes and all the anchor points are within the fibers. As a result, the objective function F_g always decreases with increasing tree cardinality, up to the point where all vertices are spanned by the tree.

[1] Additional results are available online at http://cvlab.epfl.ch/research/medical/lm/retinal/.

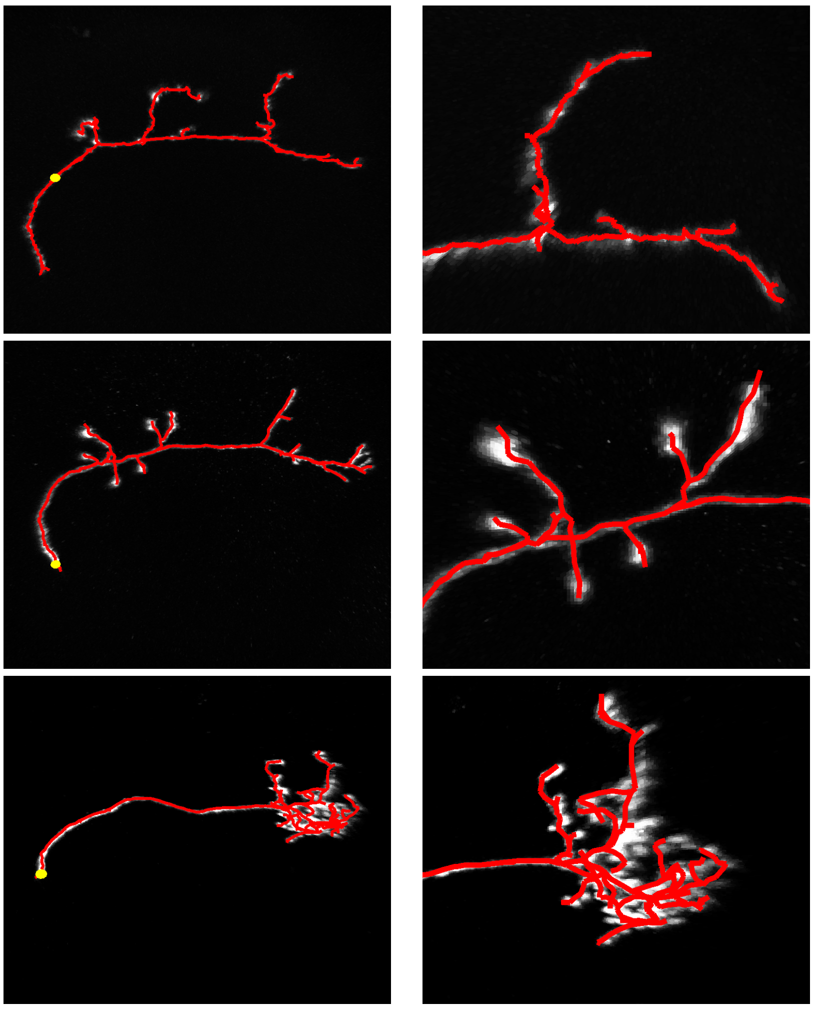


Fig. 9: Olfactory Projection Fibers. (a) Maximum intensity projections of three image stacks with the reconstructions overlaid. (b) Enlarged versions of the same images.

However, the reconstructions contain some spurious branches that are perpendicular to the main processes and that reside within them, which is attributable to the sampling methodology of Section 4.2. Because the processes occasionally have complex shapes, the sampling step generates two anchor points, one on the centerline and the other close to the surface of the fiber, corresponding to the same cross-section of the tube. This could be solved by a more sophisticated sampling strategy or by taking into account voxel width and orientation estimates when computing the paths of Eq. 3.

6.3.2 Cerebellar Climbing Fibers

These stacks are acquired from the cerebellar cortex of rats. The axons' terminals are dyed with Biotinylated Dextran Amine (Anterograde), which results in very dark processes and

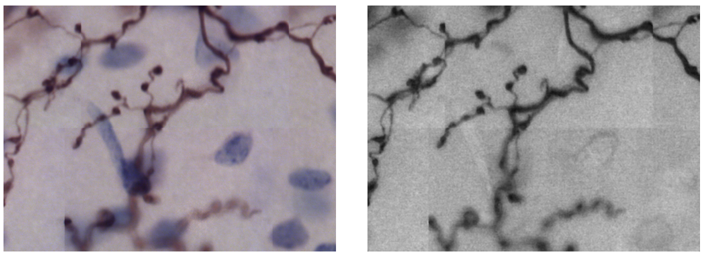

(a) (b)

Fig. 10: Conversion from color to grayscale of Cerebellar Climbing Fibers images. (a) Original image patch. (b) Grayscale version.

blob-like blue nuclei. Before processing the stacks, we converted them to grayscale by linearly combining the RGB components so as to suppress the blue blobs and maximize the absolute grayscale intensity difference between the axon terminals and the nuclei in the training stacks, as shown in Fig. 10.

As shown in Fig. 11, each stack contains several tiles that are stitched together and empty regions of space are assigned a neutral gray level. The resulting reconstructions are fairly accurate but contain some topological mistakes, such as erroneous connections at crossovers and shortcuts where the curvature is very high. Solving this would require more global geometrical constraints than the relatively local ones we impose.

Furthermore, in our implementation as discussed in Section 4.3, paths between anchor points only take into account the probabilities that their voxels are on the centerline of a filament. A natural extension that would generate more accurate and smoother paths would be to incorporate the width and the orientation estimates of the voxels in the shortest path computation.

6.3.3 Neocortical Layer 6 Axons

This dataset is obtained by labeling axons using GFP and imaging them with a two-photon microscope. The resulting image stack is depicted by Fig. 12. The axons appear as clean structures and the point spread function of the microscope is relatively small. There is some shot noise that is easily eliminated by our tubularity measure.

This dataset is nevertheless very challenging because there can be more than 30 different trees per stack. As discussed at the end of Section 5.1.2, we handle this by introducing a virtual node connected to the root node of each individual tree. These trees are then reconstructed simultaneously. Because their branches are closely spaced, it can easily happen that one tree “steals” the branches of another one, as depicted in the bottom row of Fig. 12. As a result, the DIADEM scores we obtain for this dataset are much lower than those of Fig. 7. They are in the order of 0.05 without the geometric priors and 0.1 with them. In other words, while imposing geometric regularity and explicitly sampling from junctions helps to some extent, a more global analysis of the interaction between distant anchor points and edges would be required to automatically correct such mistakes.

6.3.4 Hippocampal CA3 Neurons

These stacks are relatively similar to those of Figs. 1 and 6. They are acquired from the rat brains, dyed with biocityne, and imaged using brightfield microscopes. However, their im-

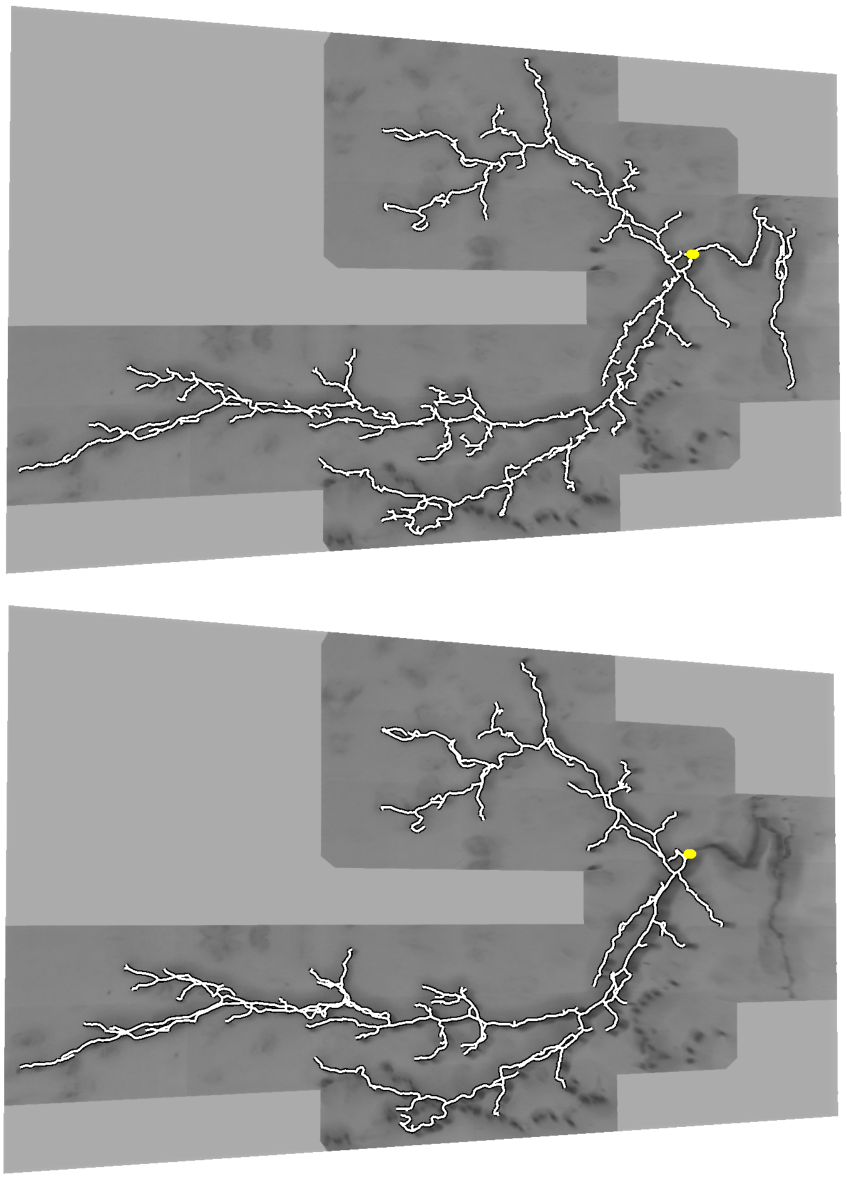


Fig. 11: Cerebellar Climbing Fibers. Top row: Image stack with reconstruction overlaid in white. The yellow dot denotes the tree root. Bottom Row: Ground truth delineation. Note that the branch on the right of the root node is correctly delineated, even though it was not traced in the ground truth.

age characteristics differ in three important ways. First, the point spread function of the microscope is very elongated in the z axis. As a result the dendrites look more like thin planes than tubular structures. Second, there are irregularities in the staining process, which make several dendrites degenerate into sequences of blobs. Finally, the dye spread around the

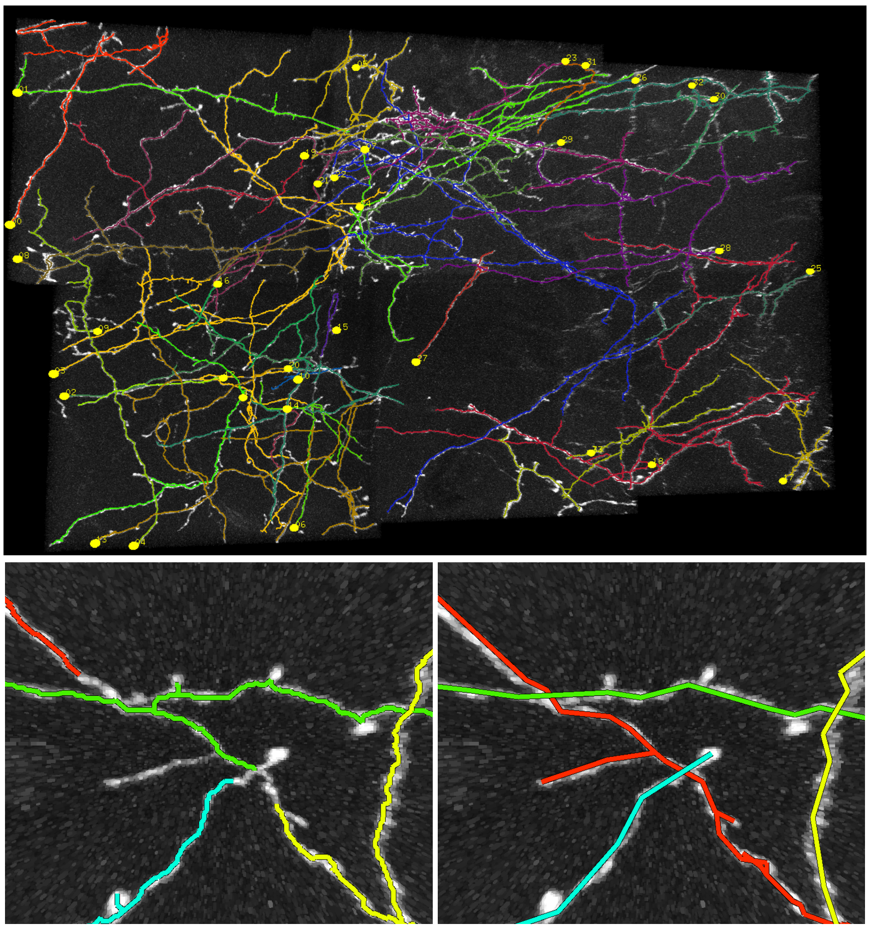


Fig. 12: Neocortical Layer 6 dataset. Top row: Image stack with reconstructions overlaid. Each colored tree corresponds to a different root node. Bottom row: An example of the tree stealing problem. The red, green and yellow trees (left) compete for the same part of the image data. In the ground truth reconstruction (right), the red tree explains all the diagonal process.

soma, generating blob-like structures. These artifacts make automated tracing much more difficult.

As in the Cerebellar Climbing Fiber dataset, the stacks consist of several tiles that we stitched together. As shown in Fig. 13, in spite of all the problems mentioned above, most of the salient processes are successfully reconstructed and the spurious ones ignored.

6.3.5 Neuromuscular Projection Fibers

This dataset consists of 152 image stack tiles of mouse neuromuscular axonal projection fibers acquired with a 2-channel confocal microscope. Starting from a set of nearby root nodes, the fibers follow approximately parallel paths while twisting around each other, as shown in Fig. 14(a).

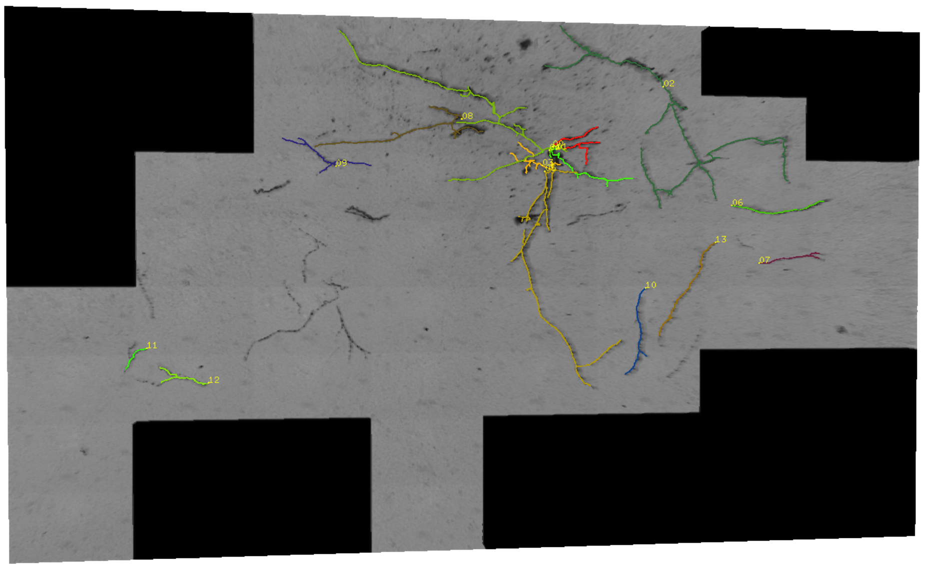


Fig. 13: Hippocampal CA3 Neurons stack with overlaid reconstructions. As in the case of Fig. 12, there are several trees overlaid using different colors.

Because the fibers are so close to each other, the reduced graph G, introduced in Section 4, inevitably contains low-cost edges that connect anchor points from different branches. At linking time, this often results in trees with branches erroneously jumping from one fiber to another. As shown in Fig. 14(b), this problem can be mitigated by increasing the sampling rate and generating more anchor points than we did for the other datasets.

However, if we were to process the whole dataset, this would result in hundreds of thousands of anchor points to be linked. This would be computationally infeasible using our current algorithm, which is why we only show results on a single tile. A feasible strategy could then be to trace the tiles sequentially and use the leaves obtained in one tile as the roots for the next one, possibly with some manual intervention to prevent the propagation of errors.

7 Conclusion

We have presented an approach to automatically inferring tree structures in 2D images and 3D image stacks by optimizing a global objective function that combines image data and geometric regularization terms. It also explicitly accounts for potential problems at bifurcations and crossovers. This allows us to recover faint and disconnected filaments at an acceptable computational cost while rejecting structured noise.

We evaluated our method on brightfield micrographs, retinal scans of the DRIVE database and the DIADEM challenge datasets. The final reconstructions are compared against manually annotated ground truths by experts. Despite the differences in the imaging modalities, our method produces visually pleasing results on all datasets, which is also supported by quantitative results. In particular, we showed that the use of the geometric regularization yields a substantial improvement in the DIADEM metric scores.

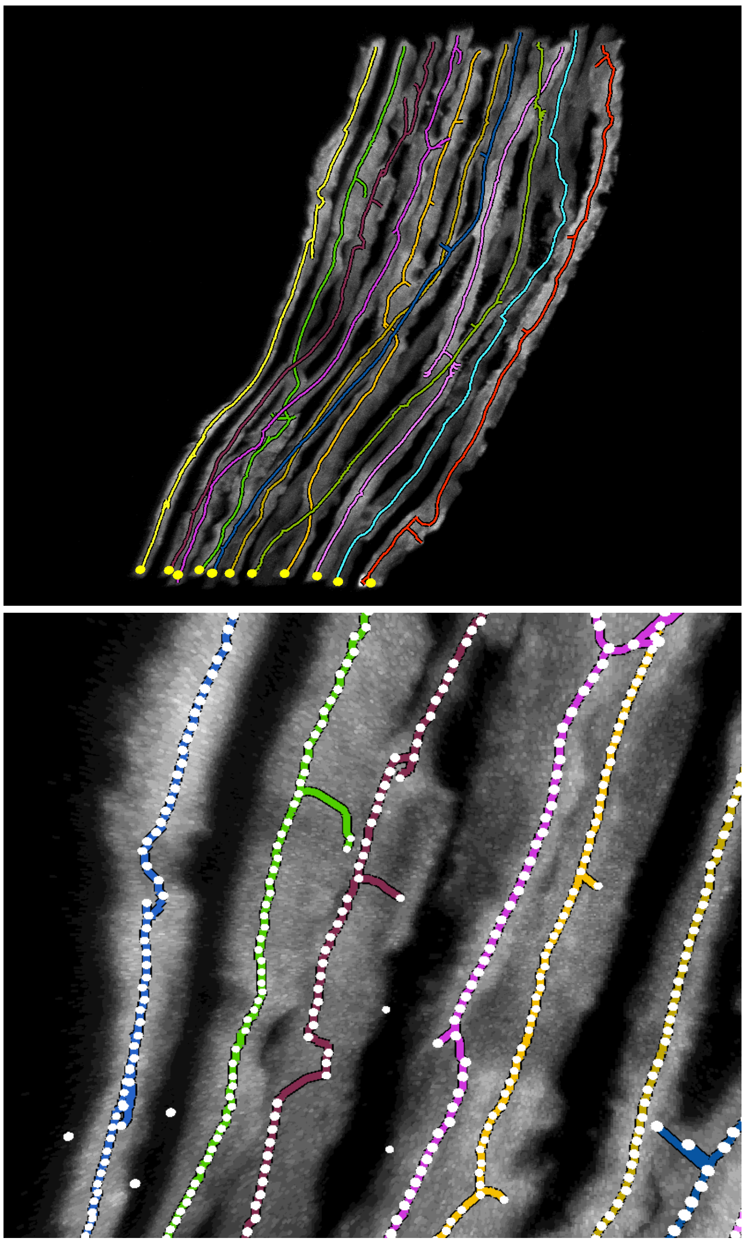


Fig. 14: The tile of the Neuromuscular Projection Fibers dataset that contains the root nodes. Top Row: Image stack with reconstructions overlaid. Each colored tree corresponds to a different root node. Bottom row: Enlarged version. The anchor points are depicted by the white circles.

The limitations of our current approach mainly come from the fact that the geometric properties we have incorporated in our objective function are still relatively local ones. As a result, we cannot account for very global properties, such as overall smoothness of a branch or expected branching factors as a given distance from the root of the tree. Our focus in future work will therefore be to extend our approach to such properties and thereby further increase its robustness.

Information and Sharing Agreement

The software described in this work will be publicly accessible online at http://cvlab.epfl.ch/research/medical/lm .

Acknowledgements This work was supported in part by the Swiss National Science Foundation and in part by the MicroNano ERC project. Christian Blum also acknowledges support from the *Ramón y Cajal* program of the Spanish Government.

We would like to thank Felix Schürmann for his advice and for giving us access to the brightfield dataset and corresponding ground-truth.

Appendix: Pseudo-code for the ACO-RTS Algorithm

In this appendix, we describe in more details our ACO-RTS algorithm and supply pseudo-code for it.

Notations and Overview

ACO-RTS is designed to minimize the primary cost function of Eq. 25 and uses the auxiliary cost function of Eq. 26 during tree construction. To this end, it maintains three sets of trees, namely the best-so-far solutions T_{bs}, the restart-best solutions T_{rb}, and the iteration-best solutions T_{ib}. Each one of these sets contains a group of trees with cardinalities $\{1, \ldots, k_{\text{limit}}\}$, where k_{limit} is the maximum cardinality of the trees to be constructed. Its value is updated at each iteration of the algorithm so as to focus the search on the regions of most likely cardinalities and to discard very high ones.

Given a directed graph $G = (V, E)$, let H be the set of all directed paths (e_{us}, e_{st}) containing exactly two consecutive edges. The pheromone model $\mathcal{P}$, first introduced in Section 5.2.1, is constructed by assigning a pheromone value τ_{ust} to each one of these *edge pairs* and τ_{rv} to each edge emanating from the root vertex x_r. More formally,

$$\mathcal{P} = \{\tau_{ust} \in [\tau_{min},\ \tau_{max}] \mid e_{st} \in E, e_{us} \in E\} \ \cup\ \{\tau_{rv} \in [\tau_{min},\ \tau_{max}] \mid e_{rv} \in E_r\}, \tag{27}$$

where $\tau_{min} \in \mathbb{R}$ and $\tau_{max} \in \mathbb{R}$ are lower and upper limits for the pheromone values and E_r is the set of edges emanating from x_r. In this work, τ_{min} and τ_{max} are set to 0.01 and 0.99, respectively. In effect, this model amounts to introducing a likelihood distribution over edge pairs, which improves convergence by resolving conflicts at crossovers.

Fig. 15 shows a simplified flowchart of the algorithm. At each iteration, n_a directed trees are probabilistically constructed by combining the pheromone information and the auxiliary cost of Eq. 26. The tree cardinality is bounded by the variable k_{limit}, whose value is dynamically updated at run-time. Tree construction always starts at the root vertex x_r, with edges being probabilistically selected and iteratively added to the tree. Edge pairs whose contribution to the auxiliary cost is small are given higher priority and care is taken not to violate the branching factor limit.

We rely on dynamic programming (Blum, 2007) to find the best l-cardinality tree for all l from 1 to k_{limit} in each one of the n_a trees that have been built. For each cardinality, we select from the resulting n_a trees the one that minimizes the auxiliary objective function and store it in T_{ib}. Each l-cardinality tree in T_{rb} or T_{bs} is then updated by the l-cardinality tree in T_{ib}, if the latter has a lower auxiliary cost. The best cardinality k^* for the next iteration is then taken to be the one that minimizes the primary objective function of Eq. 25 among the trees in T_{bs}. The cardinality limit k_{limit} is taken to be the average of k^* and $|V| - 1$, the maximum possible cardinality of a tree. The pheromone values are updated by using the k^*-cardinality trees in T_{ib}, T_{rb} and T_{bs} so that values corresponding to edge pairs belonging to these k^*-cardinality trees are increased while others are decreased. Finally, when the algorithm has run for a preset amount of time, it returns the k^*-cardinality tree in T_{bs}.

Pseudo Code

Fig. 16 depicts the pseudo-code that implements the approach outlined above. It includes the following functions:

– ConstructDirectedTree($\mathcal{P}$, k_{limit}, x_r): This function constructs a k_{limit}-cardinality tree $T = (V_T, E_T)$ with vertices $V_T \subseteq V$, edges $E_T \subseteq E$, and pairs of consecutive edges $H_T \subseteq H$. The construction starts from the given root vertex x_r, that is, initially $T = (\{x_r\}, \emptyset)$. At each step, a tree is stochastically selected from a neighborhood of the current solution. This neighborhood is generated by adding an edge and its target vertex to the tree under construction such that the tree property is maintained (i.e., no cycles are formed) and no bifurcation with a branching factor greater than a predefined threshold b_{th} is introduced. The set of candidate edges is generated as follows.

As a convention, let the source vertex of an edge $e_{st} \in E$ be denoted by x_s and the target vertex by x_t. Let also $N \subseteq V$ be the set of vertices such that $N \cap V_T = \emptyset$ and for each $x_t \in N$, there exists at least one edge $e_{st} \in E$ with $x_s \in V_T$. Moreover, let $E^t \subseteq E$ denote the set of edges that can connect the vertex x_t to T such that the following conditions hold:

1. For each $e_{st} \in E^t$, $deg^+(x_s) < b_{th}$, where $deg^+(x_s)$ denotes the out-degree of vertex x_s in tree T.

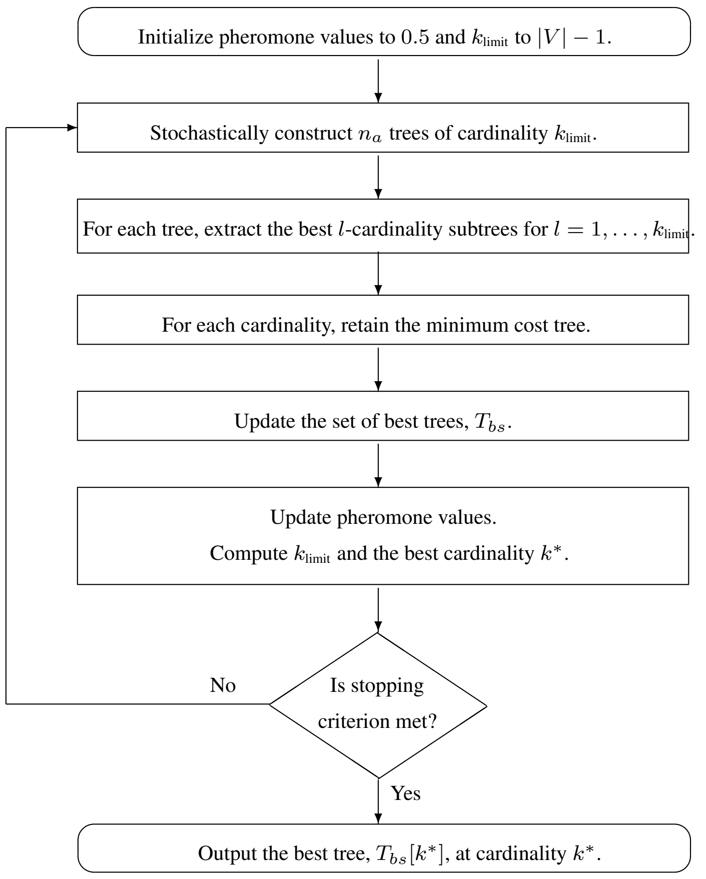

Initialize pheromone values to 0.5 and k_{limit} to $|V| - 1$.
Stochastically construct n_a trees of cardinality k_{limit}.
For each tree, extract the best l-cardinality subtrees for $l = 1, \ldots, k_{\text{limit}}$.
For each cardinality, retain the minimum cost tree.
Update the set of best trees, T_{bs}.
Update pheromone values.
Compute k_{limit} and the best cardinality k^*.
Is stopping criterion met?
No
Yes
Output the best tree, $T_{bs}[k^*]$, at cardinality k^*.

Fig. 15: ACO-RTS algorithm flowchart.

2. There does not exist an edge $e_{sl} \in E_T$ such that x_s and x_l are co-located vertices as defined in Section 4.2.

Then, the set of candidate edges is taken to be

$$\mathcal{C} = \{\underset{e_{st} \in E^t}{\text{argmin}}\{w(e_{st})\} \mid x_t \in N\}, \tag{28}$$

where $w(e_{st})$ is the effective weight of an edge e_{st}, which can be expressed in terms of the cost terms of Eq. 26.

$$w(e_{st}) = \begin{cases} d_{st} + b_{st} & \text{, if } x_s = x_r \ , \\ d_{st} + a_{ust} & \text{, if } x_s \neq x_r, e_{us} \in E_T \ . \end{cases} \tag{29}$$

At each construction step, with a constant probability $q \in [0, 1]$, the algorithm chooses an edge $e_{st} \in \mathcal{C}$ that deterministically minimizes $\tau_{ust}/w(e_{st})$, where $\tau_{ust} \in \mathcal{P}$ denotes the pheromone value assigned to the edge pair e_{st} and $e_{us} \in E_T$. Otherwise, with probability $1 - q$, $e_{st} \in \mathcal{C}$ is chosen probabilistically in proportion to $\tau_{ust}/w(e_{st})$.

The resulting trees may contain incorrect connections at crossovers. We observed that, while growing the tree from the root vertex, one of the crossing branches usually grows more rapidly than the other and dominates the crossover region by taking the ownership of the continuation edges. This is mainly caused by contrast differences in the original data and local classifier errors, which lead to high-cost edges not being selected to bridge gaps along the branches.

```
INPUT: A directed graph $G = (V, E)$ with both edge and pairwise edge weights and a root vertex $x_r$.
$n_a$ := Number of ants.
$T_{bs} := \emptyset, T_{rb} := \emptyset, k_{\text{limit}} := |V| - 1,$
$cf := 0, bs\_update :=$ FALSE.
forall $h \in \mathcal{P}$ do $\tau_h := 0.5$ end forall
while a preset time limit is not reached do
    for $j = 1$ to $n_a$ do
        $T :=$ ConstructDirectedTree($\mathcal{P}, k_{\text{limit}}, x_r$)
        $\mathcal{S} :=$ DynamicTree($T$,$k_{\text{limit}}$)
        Update($T_{ib}, \mathcal{S}, k_{\text{limit}}$)
    end for
    Update($T_{rb}, T_{ib}, k_{\text{limit}}$)
    Update($T_{bs}, T_{ib}, k_{\text{limit}}$)
    $\hat{T}^*_{bs} :=$ argmin$\{F_g(T) \mid T \in T_{bs}\}$
    $k^* := |V_{\hat{T}^*_{bs}}| - 1$, (i.e., $\hat{T}^*_{bs} = T_{bs}[k^*]$)
    $k_{\text{limit}} := k^* + \frac{|V|-k^*-1}{2}$
    PheromoneUpdate( $cf, bs\_update, \mathcal{P},$
                     $T_{ib}[k^*], T_{rb}[k^*], T_{bs}[k^*]$)
    $cf :=$ ComputeConvergenceFactor($\mathcal{P}, T_{rb}[k^*], k^*$)
    if $cf \geq 0.99$ then
        if $bs\_update =$ TRUE then
            forall $h \in \mathcal{P}$ do $\tau_h := 0.5$ end forall
            $T_{rb} := \emptyset$
            $bs\_update :=$ FALSE
        else
            $bs\_update :=$ TRUE
        end if
    end if
end while
OUTPUT: $\hat{T}^*_{bs}$
```

Fig. 16: ACO-RTS: Pseudo-code for Ant Colony Optimization for the Reconstruction of Tree-Like Structures.

To correct such errors, we introduce an additional neighborhood structure, which we name *crossover neighborhood*. Two trees are said to be crossover-neighbors if one can be obtained from the other by simply changing possession of continuation edges at two co-located crossover vertices. Fig. 17 gives an example of such a neighborhood for a crossover with three continuation edges. After each construction step, we search over this neighborhood of the constructed tree and minimize the primary cost function. Note that, this minimization reduces to the minimization of the sum of the pairwise edge weights in this case, since all the trees in the neighborhood have the same set of edges, and hence the unary edge costs are common.

– DynamicTree(T, k_{limit}): This runs a dynamic programming algorithm (Blum, 2007), which we extended to handle directed graphs. For each $l = 1, \ldots, k_{\text{limit}}$, it returns the best l-cardinality tree rooted at x_r. The quality of a tree is determined by the auxiliary cost function.

– Update($T_a, T_b, k_{\text{limit}}$): This function updates the set of trees T_a by the set T_b for each cardinality. That is, for $l = 1, \ldots, k_{\text{limit}}$, if $F'_g(T_a[l]) > F'_g(T_b[l])$, then $T_a[l] = T_b[l]$, where $T_a[l]$ and $T_b[l]$ denote the l-cardinality trees in T_a and T_b, respectively.

– PheromoneUpdate(cf, bs_update, $\mathcal{P}$, $T_{ib}[k^*]$, $T_{rb}[k^*]$, $T_{bs}[k^*]$): The pheromone values are updated using the three solutions $T_{ib}[k^*]$, $T_{rb}[k^*]$ and $T_{bs}[k^*]$ that have the same k^* cardinality. Their influence on the update depends on the state of convergence of the algorithm. This procedure is the same as the one described in (Blum and Blesa, 2005) for the k-cardinality tree problem, except that the pheromone values are assigned to edge pairs instead of individual edges. We refer the interested reader to our earlier publication for further details.

– ComputeConvergenceFactor($\mathcal{P}, T_{rb}[k^*], k^*$): In this procedure, the convergence factor of the algorithm is computed based on the convergence of the restart-best tree with cardinality k^*. We take it to

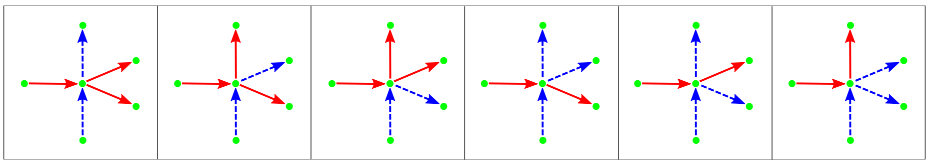


Fig. 17: Crossover neighborhood for a junction with two branches and three continuation edges. The green circles are vertices. The blue dotted lines represent one of the crossing branches and the red solid lines represent the other one.

be

$$cf = \frac{\sum_{h \in \mathcal{P}_{T_{rb}[k^*]}} \tau_h}{k^*.\tau_{max}} , \tag{30}$$

where $\mathcal{P}_{T_{rb}[k^*]}$ denotes the set of pheromone values of the edges and edge pairs included in $T_{rb}[k^*]$ and τ_{max} is the upper limit for the pheromone values. Note that, the convergence factor can only assume values in the interval $[0\ 1]$.

References

Al-Kofahi KA, Lasek S, Szarowski DH, Pace CJ, Nagy G, Turner JN, Roysam B (2002) Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. Transactions on Information Technology in Biomedicine 6(2):171–187

Ascoli GA, Svoboda K, Liu Y (2010) Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge. Http://diademchallenge.org/

Blum C (2007) Revisiting Dynamic Programming for Finding Optimal Subtrees in Trees. European Journal of Operational Research 177(1):102–115

Blum C, Blesa M (2005) Combining Ant Colony Optimization With Dynamic Programming for Solving the K-Cardinality Tree Problem. In: Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science, vol 3512, pp 25–33

Cai H, Xu X, Lu J, Lichtman J, Yung S, Wong S (2006) Repulsive force based snake model to segment and track neuronal axons in 3D microscopy image stacks. NeuroImage 32(4):1608–1620

Can A, Shen H, Turner J, Tanenbaum H, Roysam B (1999) Rapid Automated Tracing and Feature Extraction from Retinal Fundus Images Using Direct Exploratory Algorithms. Transactions on Information Technology in Biomedicine 3(2):125–138

Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT press

Duhamel C, Gouveia L, Moura P, Souza M (2008) Models and Heuristics for a Minimum Arborescence Problem. Networks 51(1):34–47

Fan D (2006) Bayesian Inference of Vascular Structure from Retinal Images. PhD thesis, Dept. of Computer Science, U. of Warwick, Coventry, UK

Felzenszwalb P, Huttenlocher D (2005) Pictorial Structures for Object Recognition. International Journal of Computer Vision 16(1):55–79

Felzenszwalb P, McAllester D (2006) A Min-Cover Approach for Finding Salient Curves. In: Conference on Computer Vision and Pattern Recognition, pp 61–74

Fischler M, Heller A (1998) Automated Techniques for Road Network Modeling. In: DARPA Image Understanding Workshop, pp 501–516

Frangi AF, Niessen WJ, Vincken KL, Viergever MA (1998) Multiscale Vessel Enhancement Filtering. Lecture Notes in Computer Science 1496:130–137

Garg N (1996) A 3-Approximation for the Minimum Tree Spanning K Vertices. In: IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, vol 27, pp 302–309

Gonzalez G, Fleuret F, Fua P (2008) Automated Delineation of Dendritic Networks in Noisy Image Stacks. In: European Conference on Computer Vision, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, vol 5305, pp 214–227

Gonzalez G, Aguet F, Fleuret F, Unser M, Fua P (2009) Steerable Features for Statistical 3D Dendrite Detection. In: Conference on Medical Image Computing and Computer Assisted Intervention, vol 12, pp 625–32

Huang K, Yan M (2006) Robust Optic Disk Detection in Retinal Images Using Vessel Structure and Radon Transform. In: SPIE, vol 6144

Jacob M, Unser M (2004) Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(8):1007–1019

Law M, Chung A (2008) Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In: European Conference on Computer Vision, pp 368–382

Lee T, Kashyap R, Chu C (1994) Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. CVGIP: Graphical Models and Image Processing 56(6):462–478

Leordeanu M, Hebert M, Sukthankar R (2007) Beyond Local Appearance: Category Recognition from Pairwise Interactions of Simple Features. In: Conference on Computer Vision and Pattern Recognition, pp 1 –8

Meijering E, Jacob M, Sarria JCF, Steiner P, Hirling H, Unser M (2004) Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. Cytometry Part A 58A(2):167–176

Platt J (2000) Advances in Large Margin Classifiers, MIT Press, chap Probabilistic Outputs for SVMs and Comparisons to Regularized Likelihood Methods

Santamaría-Pang A, Colbert CM, Saggau P, Kakadiaris I (2007) Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In: Conference on Medical Image Computing and Computer Assisted Intervention, pp 486–494

Schoelkopf B, Burges C, Smola A (1999) Advances in Kernel Methods. In: Support Vector Learning, MIT Press

Staal J, Abramoff M, Niemeijer M, Viergever M, van Ginneken B (2004) Ridge Based Vessel Segmentation in Color Images of the Retina. IEEE Transactions on Medical Imaging 23:501–509

Sun K, Sang N, Zhang T (2007) Marked Point Process for Vascular Tree Extraction on Angiogram. In: Conference on Computer Vision and Pattern Recognition, pp 467–478

Vasilkoski Z, Stepanyants A (2009) Detection of the Optimal Neuron Traces in Confocal Microscopy Images. Journal of Neuroscience Methods 178(1):197–204

Weaver C, Hof P, Wearne S, Brent L (2004) Automated Algorithms for Multiscale Morphometry of Neuronal Dendrites. Neural Computation 16(7):1353–1383

Xie J, Zhao T, Lee T, Myers G, Peng H (2010) Automatic Neuron Tracing in Volumetric Microscopy Images with Anisotropic Path Searching. In: Conference on Medical Image Computing and Computer Assisted Intervention

Xu J, Wu J, Feng D, Cui Z (2009) Dsa Image Blood Vessel Skeleton Extraction Based on Anti-Concentration Diffusion and Level Set Method. Computational Intelligence and Intelligent Systems 51:188–198

Yedidya T, Hartley R (2008) Tracking of Blood Vessels in Retinal Images Using Kalman Filter. In: Digital Image Computing: Techniques and Applications, pp 52–58