Reconstructing Curvilinear Networks Using Path Classifiers and Integer Programming

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · January 2015

DOI: 10.1109/TPAMI.2016.2519025

6 authors, including:

Engin Turetken
École Polytechnique Fédérale de Lausanne
25 PUBLICATIONS **1,614** CITATIONS

SEE PROFILE

Fethallah Benmansour
Roche
38 PUBLICATIONS **835** CITATIONS

SEE PROFILE

Hanspeter Pfister
Harvard University
341 PUBLICATIONS **15,071** CITATIONS

SEE PROFILE

Pascal Fua
École Polytechnique Fédérale de Lausanne
714 PUBLICATIONS **40,916** CITATIONS

SEE PROFILE

Reconstructing Curvilinear Networks using Path Classifiers and Integer Programming

Engin Türetken, Fethallah Benmansour, Bjoern Andres, Przemysław Głowacki, Hanspeter Pfister, *Senior Member, IEEE*, and Pascal Fua, *Fellow, IEEE*

Abstract—We propose a novel approach to automated delineation of curvilinear structures that form complex and potentially loopy networks. By representing the image data as a graph of potential paths, we first show how to weight these paths using discriminatively-trained classifiers that are both robust and generic enough to be applied to very different imaging modalities. We then present an Integer Programming approach to finding the optimal subset of paths, subject to structural and topological constraints that eliminate implausible solutions. Unlike earlier approaches that assume a tree topology for the networks, ours explicitly models the fact that the networks may contain loops, and can reconstruct both cyclic and acyclic ones. We demonstrate the effectiveness of our approach on a variety of challenging datasets including aerial images of road networks and micrographs of neural arbors, and show that it outperforms state-of-the-art techniques.

Index Terms—Curvilinear networks, tubular structures, curvilinear structures, automated reconstruction, integer programming, path classification, minimum arborescence.

1 Introduction

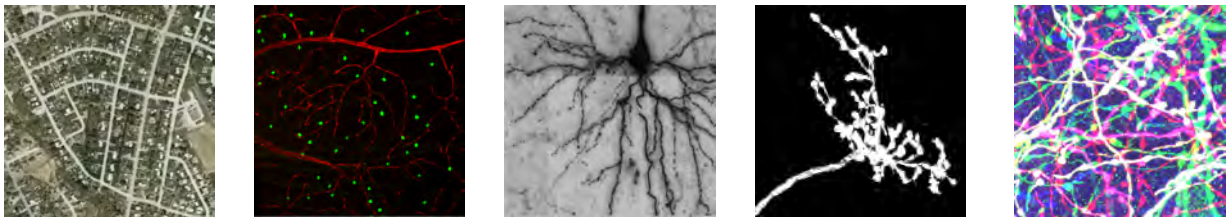
Networks of curvilinear structures are pervasive both in nature and man-made systems. They appear at all possible scales, ranging from nanometers in Electron Microscopy image stacks of neurons to petameters in dark-matter arbors binding massive galaxy clusters. Modern acquisition systems can produce vast amounts of such imagery in a short period of time. However, despite the abundance of available data and many years of sustained effort, full automation remains elusive when the image data is noisy and the structures exhibit complex morphology.

As a result, practical systems require extensive manual intervention that is both time-consuming and tedious. For example, in the DIADEM challenge to map nerve cells [2], the results of the finalist algorithms, which proved best at tracing axonal and dendritic networks, required substantial time and effort to proofread and correct [3]. Such editing dramatically slows down the analysis process and makes it impossible to exploit the large volumes of imagery being produced for neuroscience and medical research.

Part of the problem comes from the fact that scoring paths by integrating pixel values along their length, as is done in many automated methods, often fails to adequately penalize short-cuts and makes it difficult to compute commensurate scores for paths of different lengths. An additional difficulty comes from the fact that many interesting networks, such as those formed by roads and blood vessels depicted by Fig. 1 contain loops, which few existing methods attempt to model explicitly. Furthermore, even among structures that really are trees, such as the neurites of Fig. 1, the imaging resolution is often so low that the branches appear to cross, thus introducing several spurious cycles that can only be recognized once the whole structure has been recovered. In fact, this is widely reported as a major source of error [4], [5], [6], [7], [8], [9].

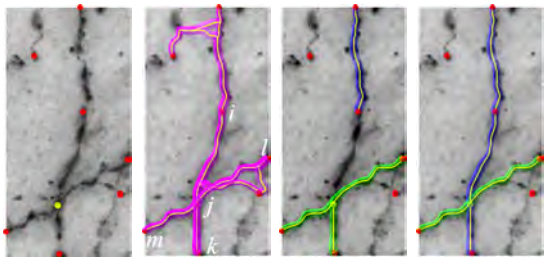
In this paper, we attempt to overcome both of these limitations simultaneously by formulating the reconstruction problem as one of solving an Integer Program (IP) on a graph of potential tubular paths. We further propose a novel approach to weighting these paths using discriminatively-trained path classifiers. More specifically, we first select evenly spaced voxels that are very likely to belong to the curvilinear structures of interest. We treat them as vertices of a graph and connect those that are within a certain distance of each other by geodesic paths [10], to which we assign informative scores based on an original classification approach. We then look for a subgraph that maximizes a global objective function that combines both image-based and geometry-based terms. Unlike earlier approaches that aim at building trees, we look for a potentially loopy subgraph while penalizing the formation of spurious junctions and early branch terminations. This is done by letting graph vertices be used by several branches, thus allowing cycles

- *Engin Türetken is with the Computer Vision Laboratory, EPFL, Lausanne, and the Swiss Center for Electronics and Microtechnology (CSEM), Neuchâtel, Switzerland.*
  E-mail: engin.tueretken@alumni.epfl.ch
- *Przemysław Głowacki and Pascal Fua are with the Computer Vision Laboratory, EPFL, Lausanne, Switzerland.*
  E-mail: przemyslaw.glowacki@epfl.ch, pascal.fua@epfl.ch
- *Fethallah Benmansour is with Roche Pharma Research and Early Development, Roche Innovation Center, Basel, Switzerland.*
  E-mail: fethallah.benmansour@roche.com
- *Bjoern Andres is with the School of Engineering and Applied Sciences, Harvard University, Cambridge, US.*
  E-mail: bjoern@andres.sc
- *Hanspeter Pfister is with the Visual Computing Group, Harvard University, Cambridge, US.*
  E-mail: pfister@seas.harvard.edu



Road Networks (a) Aerial | Blood Vessels (b) Confocal | Neurons (c) Brightfield | Axons (d) Confocal | Dendrites and Axons (e) Brainbow

Fig. 1. 2D and 3D datasets used to test our approach. (a) Aerial image of roads. (b) Maximum intensity projection (MIP) of a confocal image stack of blood vessels, which appear in red. It features cycles created by circulatory anastomoses or by capillaries connecting arteries to veins. (c) Minimum intensity projection of a brightfield stack of neurons. (d) MIP of a confocal stack of olfactory projection axons. (e) MIP of a brainbow [1] stack. As most images in this paper, they are best visualized in color.



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

Fig. 2. Enforcing tree topology results in creation of spurious junctions. (a) Image with two crossing branches. The dots depict sample points (seeds) on the centerlines found by maximizing a tubularity measure. In 3D, these branches may be disjoint but the z-resolution is insufficient to see it and only a single sample is found at their intersection, which we color yellow. (b) The samples are connected by geodesic paths to form a graph. (c,d) The final delineation is obtained by finding a subgraph that minimizes a global cost function. In (c), we prevent samples from being used more than once, resulting in an erroneous delineation. In (d), we allow the yellow point to be used twice, and penalize creation of early terminations and spurious junctions, which yields a better result.

as shown in Fig. 2, but introducing a regularization prior and structural constraints to limit the number of branchings and terminations.

Our approach was first introduced in [11] and [12], both of which produce connected reconstructions. We combine them here and propose new connectivity constraints along with a new optimization strategy to impose them, which result in significant speed-ups. Furthermore, we provide a more thorough validation with new experiments and comparisons to several other methods.

We demonstrate the effectiveness of our approach on a wide range of noisy 2D and 3D datasets, including aerial images of road networks, and micrographs of neural and vascular arbors depicted by Fig. 1. We show that it consistently outperforms state-of-the-art techniques.

2 Related Work

Most delineation techniques and curvilinear structure databases use a sequence of cylindrical compartments as a representation of curvilinear structures [13], [14], [15], [16], [17], [18], [11], [19]. Besides being compact, this geometric representation captures two important properties, namely connectedness and directedness, which are common among many structures such as blood vessels and neurons and are essential in studying their morphology. Current approaches can be roughly categorized as manual, semi-automated or automated.

Since we focus on automation, we discuss the first two categories in Appendix A and briefly review existing automated techniques below. Most of them require a seed point, the root, to be specified for each connected network in the image. Starting from the roots, they then grow branches that follow local maxima of a *tubularity* measure that captures the likelihood of a point being on the centerline of a curvilinear structure. Depending on the search mechanism employed, existing algorithms can be subdivided into local and global ones as discussed below.

2.1 Local Search

Local search algorithms make greedy decisions about which branches to keep in the solution based on local image evidence. They include methods that segment and skeletonize the tubularity image [20], [13], [21], [22], [23], [24], and active contour-based methods, which are initialized from it [25], [26], [5]. Such methods are shown to be effective and efficient when a very good segmentation can be reliably obtained. In practice, however, this is hard to do. 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 network.

Another important class of local approaches involves explicitly delineating the curvilinear networks. It includes greedy tracking methods that start from a set of seed points and incrementally grow branches by evaluating a tubularity measure [27], [28], [29], [30], [6], [31], [9]. High tubularity paths are then iteratively added to the solution and their end points are treated as the new seeds from which the process can be restarted. These techniques are computationally efficient because the tubularity measure only needs to be evaluated for a small subset of the image volume in the vicinity of the seeds. However, they typically

require separate procedures to detect branching points. Furthermore, due to their greedy nature, imaging artifacts and noise can produce local tracing errors that propagate. This often results in large morphological mistakes, especially when there are large gaps along the filaments.

2.2 Global Search

Global search methods aim to achieve greater robustness by computing the tubularity measure everywhere. Although this is more computationally demanding, it can still be done efficiently in Fourier space or using GPUs [32], [33], [34]. They then optimize a global objective function over a graph of high-tubularity seed points [8] or superpixels [35]. Markov Chain Monte Carlo algorithms, for instance, belong to this class [36], [37], [38]. These methods explore the search space efficiently by first sampling seed points and linking them, and then iteratively adjusting their positions and connections so as to minimize their objective function. However, while they produce smooth tree components in general, the algorithms presented in [36], [37] do not necessarily guarantee spatial connectedness of these components.

By contrast, many other graph-based methods use the specified roots to guarantee connectivity. They sample seed points and connect them by paths that follow local maxima of the tubularity measure. This defines a graph whose vertices are the seeds and edges are the paths linking them. The edges of the graph form an overcomplete representation of the underlying curvilinear structures and the final step is to build a solution by selecting an optimal subset of these candidate edges.

Many existing approaches weight the edges of this graph and pose the problem as a minimum-weight tree problem, which is then solved optimally. Algorithms that find a Minimum Spanning Tree (MST) [39], [14], [7], [18] or a Shortest Path Tree (SPT) [17] belong to this class. Although efficient polynomial-time algorithms exist for both SPT- and MST-based formulations, these approaches suffer from the fact that they must span all the seed points, including some that might be false positives. As a result, they produce spurious branches when seed points that are not part of the tree structure are mistakenly detected, which happens all too often in noisy data. Of course, some of the spurious branches can be eliminated after the fact by pruning the tree. In the results section, we will compare this pruning approach to our IP one and show that the latter usually gives better results.

Our earlier k-Minimum Spanning Tree (k-MST) formulation [8] addressed this issue by posing the problem as one of finding the minimum cost tree that spans only an *a priori* unknown subset of k seed points. However, the method relies on a heuristic search algorithm and two distinct objective functions, one for searching and the other for scoring, without guaranteeing the global optimality of the final reconstruction. Furthermore, it requires an explicit optimization over the auxiliary variable k, which is not relevant to the problem. By contrast, the integer programming formulation we advocate in this paper involves minimization of a single global objective function that allows us to link legitimate seed points while rejecting spurious ones by finding the optimum solution to within a small user-specified tolerance.

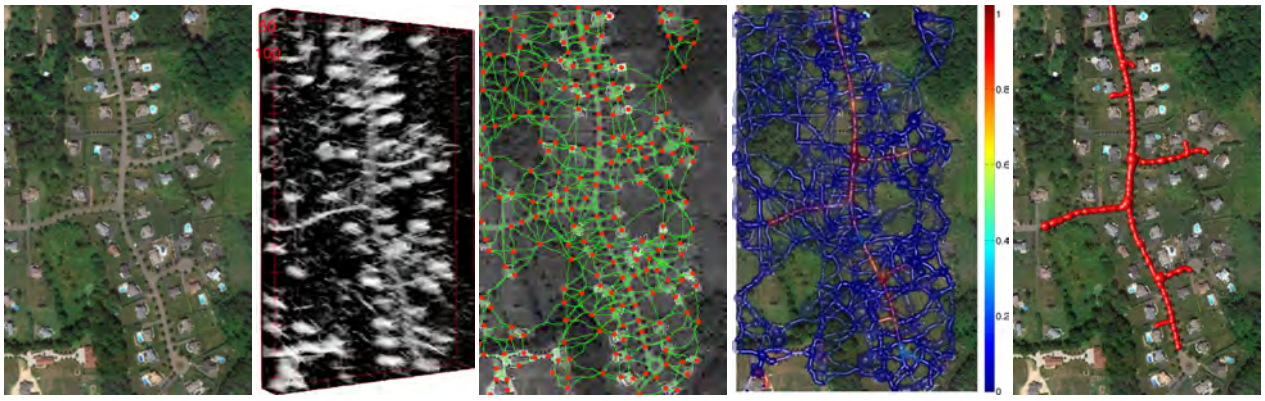
Furthermore, all the spanning tree approaches rely on the local tubularity scores to weight the graph edges. For example, global methods that rely on geodesic distances express this cost as an integral of a function of the tubularity values [40], [7]. Similarly, active contour-based methods typically define their energy terms as such integrals over the paths [18], [33]. In our own experience [8], because they involve averaging, such measures are not particularly effective at ignoring paths that are mostly on the curvilinear structures but also partially on the background. Moreover, because the scores are computed as sums of values along the path, normalizing them so that paths of different lengths can be appropriately compared is non-trivial. By contrast, the path classification approach we introduce in Section 5 returns comparable probabilistic costs for paths of arbitrary length. Furthermore, our path features capture global appearance, while being robust to noise and inhomogeneities.

Last but not least, none of the existing approaches addresses the delineation problem for loopy structures. For the cases where spurious loops seem to be present, for example due to insufficient imaging resolution in 3D stacks or due to projections of the structures on 2D [41], some of the above-mentioned methods [7], [5] attempt to distinguish spurious crossings from legitimate junctions by penalizing sharp turns and high tortuosity paths in the solutions and introducing heuristics to locally and greedily resolve the ambiguities. They do not guarantee global optimality and, as a result, can get trapped into local minima, as reported in several of these papers. This is what our approach seeks to address by looking for the global optimum of a well-defined objective function.

3 Approach

We first briefly outline our reconstruction algorithm, which goes through the following steps depicted by Fig. 3:

1) We first compute a *tubularity* value at each image location x_i and radius value r_i, which quantifies the likelihood that there exists a tubular structure of radius r_i, at location x_i. Given an N-D image, this creates an $(N+1)$-D scale-space tubularity volume such as the one shown in Fig. 3(b).
2) We select regularly spaced high-tubularity points as seed points and connect pairs of them that are within a given distance from each other. This results in a directed tubular graph, such as the one of Fig. 3(c), which serves as an overcomplete representation for the underlying curvilinear networks.
3) Having trained a path classifier using such graphs and ground-truth trees, we assign probabilistic weights to pairs of consecutive edges of a given graph at detection time as depicted by Fig. 3(d).
4) We use these weights and solve an integer program to compute the maximum-likelihood directed subgraph



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

Fig. 3. Algorithmic steps. (a) Aerial image of a suburban neighborhood. (b) 3-D scale-space tubularity image. (c) Graph obtained by linking the seed points. They are shown in red with the path centerlines overlaid in green. (d) The same graph with probabilities assigned to edges using our path classification approach. Blue and transparent denote low probabilities, red and opaque high ones. Note that only the paths lying on roads appear in red. (e) Reconstruction obtained by our approach.

of this graph to produce a final result such as the one of Fig. 3(e).

These four steps come in roughly the same sequence as those used in most algorithms that build trees from seed points, such as [39], [7], [18], [8], but with three key differences. First, whereas heuristic optimization algorithms such as MST followed by pruning or the k-MST algorithm of [8] offer no guarantee of optimality, our approach guarantees that the solution is within a small tolerance of the global optimum. Second, our approach to scoring individual paths using a classifier instead of integrating pixel values as usually done gives us more robustness to image noise and provides peaky probability distributions, such as the one shown in Fig. 3(d), which helps ensure that the global optimum is close to the ground truth. Finally, instead of constraining the subgraph to be a tree as in Fig. 2(c), we allow it to contain cycles, as in Fig. 2(d), and penalize spurious junctions and early branch terminations. In the results section, we show that this yields improved results over very diverse datasets.

4 Graph Construction

We build the graphs such as the one depicted by Fig. 3(c) in four steps. We first compute the Multi-Directional Oriented Flux tubularity measure introduced in [42]. This measure is used to assess if a voxel lies on a centerline of a filament at a given scale. Unlike the original Oriented Flux approach [33] that relies on a circular model of the cross-sections, this measure allows for arbitrarily-shaped ones that are prevalent in biological imagery. This is achieved by maximizing the image gradient flux along multiple directions and radii, instead of only two with a unique radius. We then suppress non-maxima responses around filament centerlines using the NMST algorithm [42], which helps remove some artifacts from further consideration. More specifically, the NMST algorithm suppresses voxels that are not local maxima in the plane that is perpendicular to a local orientation estimate and within the circular neighborhood defined by their scale. It then computes the MST of the tubularity measure and links pairs of disconnected components in the non-maxima suppressed volume with the MST paths.

Next, we select maximum tubularity points in the resulting image and treat them as graph vertices, or *seeds*, to be linked. They are selected greedily, at a minimal distance of d to every point that was selected previously. Finally, we compute paths linking every pair of seed points within a certain distance $l(d)$ from each other using the minimal path method in the scale space domain [40]. We take the *geodesic tubular path* connecting vertices i and j to be

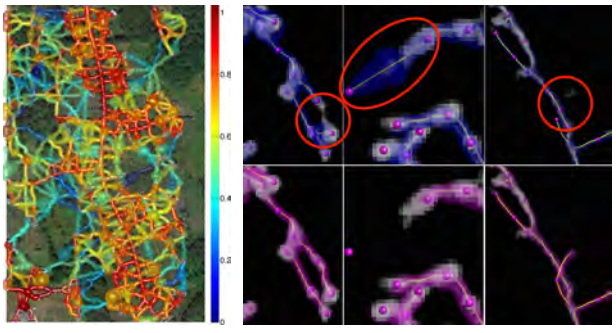
$$p_{ij} = \underset{\gamma}{\operatorname{argmin}} \int_0^1 \mathcal{P}\left(\gamma(s)\right) \mathrm{d}s, \tag{1}$$

where $\mathcal{P}$ is the negative exponential mapping of the tubularity measure, $s \in [0,1]$ is the arc-length parameter and γ is a parametrized curve mapping s to a location in $\mathbb{R}^{N+1}$ [10]. As mentioned earlier, the first N dimensions are spatial ones while the last one denotes the scale. The minimization relies on the Runge-Kutta gradient descent algorithm on the geodesic distance, which is computed using the Fast Marching algorithm [43].

5 Path Classification

Once the graph has been built, a key component of our algorithm, and one of the two main contributions of this paper, is our approach to assigning weights to its edges, or more specifically pairs of consecutive edges. As can be seen in Fig. 3(d), it is designed so that only relatively few edge pairs receive high scores and considerably simplifies the task of the IP solver used in step 4. In fact, we have checked that without this property, the solver often fails to converge. Furthermore, we will show in Section 7 that using these weights in conjunction with other approaches to finding desirable subgraphs does also bring about an improvement, albeit a smaller one than using the IP solver.

A standard approach to computing such weights is to integrate a function of the tubularity values along the paths, as in Eq. 1. However, as shown in Fig. 4(a), the resulting estimates are often unreliable because a few very



(a) (b)

Fig. 4. Scoring paths by classification vs Integration. (a) Tubular graph of Fig. 3(c) with edge weights computed by integrating tubularity values along the paths instead of using our path classification approach. We use the same color scheme as in Fig. 3(d) to demonstrate how much less informative these weights are. (b) In one of the microscopy stacks of the DIADEM data, scoring paths by summing tubularity values in, from left to right, shortcuts, spurious branches, and missing branches denoted by the red circles at the top. Using our classification approach to scoring paths yields the right answer in all three cases, as shown in the bottom row.

high values along the path might offset low values and, as a result, fail to adequately penalize spurious branches and short-cuts. Furthermore, it is often difficult to find an adequate balance between allowing paths to deviate from a straight line and preventing them from meandering too much, which results in errors such as those depicted by the top row of Fig. 4(b) when using them to find the optimal subgraph in Step 4 of our algorithm. In this section, we propose a path-classification approach to computing the probability estimates that we found to be more reliable. More specifically, given a tubular path computed as discussed in Section 4, we break it down into several segments and compute one feature vector for each based on gradient histograms. We then use an embedding approach [44] to compute fixed-size descriptors from the potentially arbitrary number of feature vectors we obtain. Finally, we feed these descriptors to a classifier and turn its output into a probability estimate.

As shown in the bottom row of Fig. 4(b), this approach penalizes paths that mostly follow the tree structure but cross the background. Thus, it discourages shortcuts and spurious branches, which the integration approach along the path fails to do.

In the remainder of this section, we describe our path features, embedding scheme, and training data collection mechanism in more detail.

5.1 Histogram of Gradient Deviation Descriptors

Gradient orientation histograms have been successfully applied to detecting objects in images and recognizing actions in videos [45], [46], [44]. In a typical setup, the image is first divided into a grid of fixed-size blocks, called cells, and then for each cell, a 1-D histogram of oriented gradients (HOG) is formed from the pixel votes within it. Histograms from neighboring cells are then combined

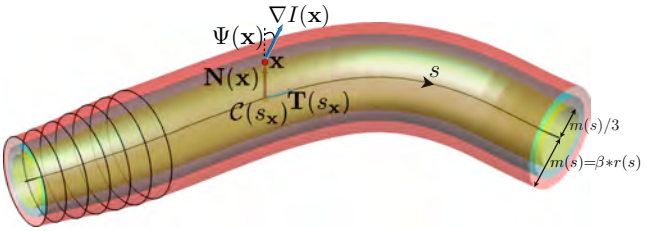


Fig. 5. Three aspects of our feature extraction process. An extended neighborhood of points around the path centerline $\mathcal{C}(s)$ is defined as the envelope of cross-sectional circles shown in black. This neighborhood is divided into R radius intervals highlighted by the yellow, green and red tubes (here $R = 3$) and a histogram is created for each such interval. A point $\mathbf{x}$ contributes a weighted vote to an angular bin according to the angle between the normal $\mathbf{N}(\mathbf{x})$ and the image gradient $\nabla I(\mathbf{x})$ at that point.

and normalized to form features invariant to local contrast changes. Finally, these features are fed into a classifier to detect objects of interest. We adapt this strategy for tubular paths by defining *Histogram of Gradient Deviation* (HGD) descriptors as follows.

Given a tubular path $\gamma(s)$ such as the one depicted by Fig. 5, with s being the curvilinear abscissa, let $\mathcal{C}(s)$ be the centerline and $r(s)$ the corresponding radius mappings. We partition the path into equal-length overlapping segments $\gamma_i(s)$ and, for each, we compute histograms of angles between image gradients $\nabla I(\mathbf{x})$ and normal vectors $\mathbf{N}(\mathbf{x})$ obtained from the points $\mathbf{x} \in \gamma_i(s)$ within the tubular segment. To ensure that all the gradient information surrounding the path is captured, we enlarge the segments by a margin $m(s) = \beta * r(s)$ proportional to the radius values. A fixed size margin is not preferred to avoid biased interference of background gradients for thin paths.

Furthermore, to obtain a description of paths' cross-sectional appearance profile, we split its tubular segments into R equally spaced radius intervals as shown in Fig. 5 and create two histograms for each such interval. While the first histogram captures the *strength* of the gradient field inside an interval, the second one captures its *symmetry* about the segment centerline.

Given a point $\mathbf{x} \in \gamma_i(s)$, let $\mathcal{C}(s_\mathbf{x})$ be the closest centerline point, and $\mathbf{N}(\mathbf{x})$ be the *normal ray vector* from $\mathcal{C}(s_\mathbf{x})$ to $\mathbf{x}$, as illustrated by Fig. 5. Each such point contributes a vote to a bin of the *gradient-strength* and *gradient-symmetry* histograms. The bin is determined by the following equation:

$$\Psi(\mathbf{x}) = \begin{cases} \text{angle}(\nabla I(\mathbf{x}), \mathbf{N}(\mathbf{x})) \text{, if } \|\mathbf{x} - \mathcal{C}(s_\mathbf{x})\| > \varepsilon \\ \text{angle}(\nabla I(\mathbf{x}), \mathbf{\Pi}(\mathbf{x})) \text{, otherwise,} \end{cases} \quad (2)$$

where $\mathbf{\Pi}(\mathbf{x})$ is the cross-sectional plane, which we use to compute the deviation angle in the special case when $\mathbf{x}$ belongs to the centerline and the normal ray vector is not defined. The votes are weighted by $\|\nabla I(\mathbf{x})\|$ and $\sqrt{< -\nabla I(\mathbf{x}), \nabla I(\mathcal{C}(s_\mathbf{x}) - \mathbf{N}(\mathbf{x})) >}$ for the *gradient-strength* and *gradient-symmetry* histograms respectively.

We compute the radius interval and the angular bin indices for a point $\mathbf{x}$ respectively as $\min(R - 1, \lfloor R\|\mathbf{N}(\mathbf{x})\|/(r(s_\mathbf{x}) + m(s_\mathbf{x}))\rfloor)$ and

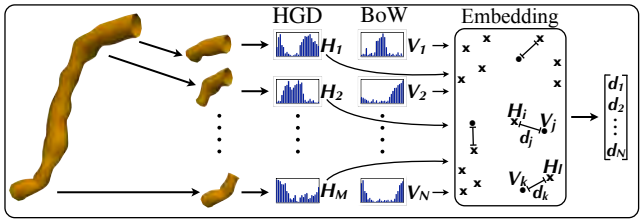


Fig. 6. Flowchart of our approach to extracting appearance features from tubular paths of arbitrary length.

$\min(B-1, \lfloor B\Psi(\mathbf{x})/\pi \rfloor)$, where B is the number of histogram bins. For each segment, this produces R pairs of histograms, each one corresponding to a radius interval. Finally, we normalize each histogram by the number of points that voted for it.

This yields a set of histograms for each segment, which we combine into a single HGD descriptor.

5.2 Embedding

The above procedure produces an arbitrary number of HGD descriptors per path. To derive a fixed-size descriptor from them, we first use a Bag-of-Words (BoW) approach to compactly represent their feature space. The words of the BoW model are generated during training by randomly sampling a predefined number of descriptors from the training data. For a given path of arbitrary length, we then compute an embedding of the path's HGD descriptors into the *codewords* of the model. As illustrated in Fig. 6, computing the embedding amounts to finding the minimum Euclidean distance from the descriptors to each word in the model. The resulting feature vector of distances has the same length as the number of elements in the BoW model.

To account for geometry and characterize paths that share a common section, such as the one shown in Fig. 7(a), we incorporate into these descriptors four geometry features, the maximum curvature along the centerline curve $\mathcal{C}(s)$, its tortuosity, and normalized length along the z and the scale axes defined in Section 4. For a path of length L in world coordinates and distance d between its start and end points, these features are defined respectively as $\text{argmax}\|\mathbf{T}'(s)\|$, d/L, Δ_z/L and Δ_r/L, where Δ_z and Δ_r are the path lengths along the image axial dimension z and the scale dimension, and $\mathbf{T}(s)$ is the unit tangent vector depicted in Fig. 5.

5.3 Training

Given a set of training images $\{I_i\}$, the associated ground truth tracings $\{H_i\}$ annotated manually and tubular graphs $\{G_i\}$ obtained using the method of Section 4, we sample an equal number of positive and negative paths from each pair $\{H_i, G_i\}$. To train the path classifier, we obtain positive samples by simply sampling the ground truth delineations $\{H_i\}$ associated with our training images. We obtain negative samples by first randomly selecting *candidate* paths from a tubular graph G_i, and then attempting to find *matching* paths in the corresponding ground truth H_i. The *candidate* paths are considered as negatives if they satisfy certain incompatibility conditions, such as having a small overlap with their respective *matching* ones.

More specifically, given a randomly selected *candidate* path p_g of a graph G_i, we find its *matching* p_h by first finding the two centerline points in H_i that are closest to the start and end points of p_g in the world coordinates of the image I_i and then extracting the shortest directed path p_h connecting these points in H_i.

Let $l(p)$ denote the centerline length of a path p, and $l_{p_1}(p_2)$ denote the length of p_2's longest segment that is outside the volume enclosed by p_1. We consider p_g as a negative example if it satisfies at least one of the following criteria:

1) $max(l_{p_g}(p_h), l_{p_h}(p_g))$ is larger than a length threshold t_l, which is taken to be the minimum image spacing in our experiments.
2) The ratio $min(l(p_h), l(p_g))/max(l(p_h), l(p_g))$ is smaller than a threshold, which we set to 0.75. This is to detect if p_g is meandering too much with respect to p_h.
3) Volumetric intersection of p_h and p_g over their union is smaller than a threshold, taken to be 0.5 in our experiments.

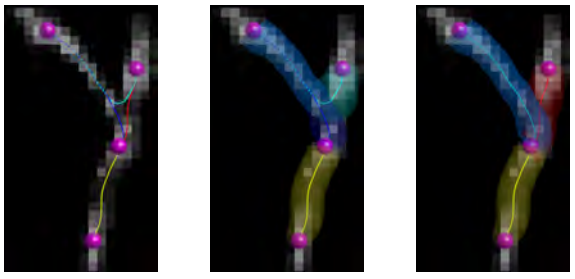
We label those negatives that partially overlap with the *matching* paths as hard-to-classify examples, and those that do not overlap as easy-to-classify ones. In our experiments, hard-to-classify examples constitute 99 percent of all the negative examples.

We choose the path lengths randomly from a probability distribution built from the consecutive edge pair lengths of the graphs $\{G_i\}$ in the training dataset. This is achieved by first labeling the edge pairs as positive or negative using the above procedure and then constructing two separate distributions, one for each class, using the assigned labels.

Finally, at detection time, we run the path classifier on consecutive edge pairs and assign to them the resulting probabilities of belonging to the underlying curvilinear networks. In practice, as we show in Appendix B, we found a Gradient Boosted Decision Tree classifier to be better suited for this learning task than SVM classifiers with linear or RBF kernels, or random forest classifiers with oblique or orthogonal splits. It is therefore the one we use in our implementation.

6 Formulating the Integer Program

Formally, the graph-building procedure of Section 4, yields a graph $G = (V, E)$, whose vertices V represent the seed points and directed edges $E = \{e_{ij} \mid i \in V, j \in V\}$ geodesic tubular paths linking them. It is designed to be an overcomplete representation of the underlying network of tubular structures. In this section, we formulate the finding of an optimal subgraph of curvilinear structures as an integer program. We first constrain it to be a directed tree and then relax this constraint.



(a) (b) (c)

Fig. 7. Considering geometric relationships between edges helps at junctions. (a) A closeup of the graph built by our algorithm at a branching point. (b) Minimizing a sum of individual path costs yields these overlapping paths. (c) Accounting for edge pair geometry yields the correct connectivity.

6.1 Standard Formulation

Many earlier approaches to finding optimal subgraphs [39], [7], [18], [8] can be understood as maximizing an *a posteriori* probability given a tubularity image and optionally a set of meta parameters that encode geometric relations between vertices and edge pairs. For example, in [8], the tree reconstruction problem is addressed by solving

$$\min_{\mathbf{t}\in\mathcal{T}(G)} \left\{ \sum_{e_{ij}\in E} c^a_{ij} \;\; t_{ij} + \sum_{e_{ij},\; e_{jk}\in E} c^g_{ijk} \;\; t_{ij} \;\; t_{jk} \right\} , \quad (3)$$

where $\mathcal{T}(G)$ denotes the set of all trees in G, t_{ij} is a binary variable indicating the presence or absence of e_{ij} in tree $\mathbf{t}$. c^a_{ij} represents the cost of an edge, which can be either negative or positive and is computed by integrating pixelwise negative log-likelihood ratio values along the path connecting the vertices, while c^g_{ijk} encodes the geometric compatibility of consecutive edges. As shown in Fig. 7, these geometric terms are key to eliminating edge sequences that backtrack or curve unnaturally. This approach, however, has three severe shortcomings. First, because the c^a_{ij} and c^g_{ijk} are computed independently, they are not necessarily commensurate or consistent with each other. As a consequence, careful weighting of the two terms is required for optimal performance. Second, it disallows cycles by preventing vertices from being shared by separate branches, as is required for successful reconstruction cases such as the one of Fig. 2. Finally, optimizing using a heuristic algorithm [47] does not guarantee a global optimum.

Nevertheless, we will show in the results section that the probabilities returned by the path classifier of Section 5 are sufficiently discriminative to produce competitive results using this kind of formulation. Furthermore, even better ones can be obtained by explicitly addressing the issues discussed above. We handle the first one by computing probability estimates, not on single edges, but on edge pairs so that both appearance and geometry can be accounted for simultaneously. Given such estimates, we solve the remaining two problems by reformulating the reconstruction problem not as the heuristic minimization of an energy such as the one of Eq. 3 but as a solution of an IP that allows loops while penalizing the formation of spurious junctions. In the following, we first introduce our probabilistic model and consider the simpler acyclic case. We then extend our approach to networks that contain loops.

6.2 Probabilistic Model

Let $F = \{e_{ijk} = (e_{ij}, e_{jk})\}$ be the set of pairs of consecutive edges in G. By analogy to the binary variable t_{ij} of Eq. 3, let t_{ijk} denote the presence or absence of e_{ijk} in the curvilinear networks and T_{ijk} be the corresponding hidden variable. Let also $\mathbf{t}$ and $\mathbf{T}$ be the set of all t_{ijk} and T_{ijk} variables respectively. Given the graph G and the image evidence I, we look for the optimal delineation $\mathbf{t}^*$ as the solution of

$$\begin{aligned} \mathbf{t}^* &= \operatorname*{argmax}_{\mathbf{t}\in\mathcal{P}_c} P(\mathbf{T}=\mathbf{t}|I,G) \;\; , \\ &= \operatorname*{argmax}_{\mathbf{t}\in\mathcal{P}_c} P(I,G|\mathbf{T}=\mathbf{t})P(\mathbf{T}=\mathbf{t}) \;\; , \\ &= \operatorname*{argmin}_{\mathbf{t}\in\mathcal{P}_c} -\log(P(I,G|\mathbf{T}=\mathbf{t})) - \log(P(\mathbf{T}=\mathbf{t})), \end{aligned} \quad (4)$$

where we used the Bayes' rule and assumed a uniform distribution over the image I and the graph G. The binary vector $\mathbf{t}$ belongs to the set $\mathcal{P}_c$ of binary vectors that define feasible solutions. In the following, we present two different ways of defining $\mathcal{P}_c$ and the two likelihood terms of Eq. 4.

6.3 Tree Reconstruction

In this section, we take $\mathcal{P}_c$ to be the set of all directed trees $\mathcal{T}(G)$ of G, and use a uniform prior, which amounts to assuming that all tree shapes are equally likely. We therefore drop the prior term $\log(P(\mathbf{T}=\mathbf{t}))$ from Eq. 4. Furthermore, we assume conditional independence of image evidence along the edge pairs $\{e_{ijk}\}$, given that we know whether or not they belong to the tree structure. We therefore represent the likelihood term $P(I,G|\mathbf{T}=\mathbf{t})$ as a product of individual edge pair likelihoods. As shown in Appendix C, this leads to

$$\mathbf{t}^* = \operatorname*{argmin}_{\mathbf{t}\in\mathcal{T}(G)} \sum_{e_{ijk}\in F} -\log\left(\frac{P(T_{ijk}=1|I_{ijk},e_{ijk})}{P(T_{ijk}=0|I_{ijk},e_{ijk})}\right) t_{ijk} \quad (5)$$

$$= \operatorname*{argmin}_{\mathbf{t}\in\mathcal{T}(G)} \sum_{e_{ijk}\in F} c_{ijk} \;\; t_{ijk}, \quad (6)$$

where I_{ijk} represents image data around the tubular path corresponding to e_{ijk}. The probability $P(T_{ijk}=1|I_{ijk},e_{ijk})$ denotes the likelihood of edge pair e_{ijk} belonging to the tree structure, which we compute based on global appearance and geometry of the paths as described in Section 5. The c_{ijk} variables represent the probabilistic likelihood ratios we assign to the edge pairs. As discussed in Section 2, we found this more effective at distinguishing legitimate paths from spurious ones than more standard methods, such as those that integrate a tubularity measure along the path. In the following, we show that optimizing the objective function of Eq. 6 with respect to the constraints $\mathbf{t} \in \mathcal{T}(G)$ amounts to solving a minimum arborescence problem [48] with a quadratic cost.

By decomposing the indicator variable t_{ijk} introduced above as the product of the two variables t_{ij} and t_{jk}, the minimization of Eq. 6 can be reformulated as the quadratic program

$$\operatorname*{argmin}_{\mathbf{t}\in\mathcal{T}(G)} \sum_{e_{ij},e_{jk}\in E} c_{ijk} \;\; t_{ij} \;\; t_{jk} \quad (7)$$

However, not all choices of binary values for the indicator variables give rise to a plausible delineation. The above minimization must therefore be carried out subject to the constraints $\mathbf{t} \in \mathcal{T}(G)$ to ensure that the solution is a connected tree. We define these constraints by adapting the network flow formulation presented in [48], which provides a compact system with a polynomial number of variables and constraints. Assuming that the root vertex $r \in V$ of the optimal tree is given, we express these constraints as

$$\sum_{j \in V \setminus \{r\}} y^l_{rj} \leq 1, \qquad \forall l \in V \setminus \{r\}, \tag{8}$$

$$\sum_{j \in V \setminus \{l\}} y^l_{jl} \leq 1, \qquad \forall l \in V \setminus \{r\}, \tag{9}$$

$$\sum_{j \in V \setminus \{i,r\}} y^l_{ij} - \sum_{j \in V \setminus \{i,l\}} y^l_{ji} = 0, \quad \begin{array}{l} \forall l \in V \setminus \{r\}, \\ \forall i \in V \setminus \{r,l\}, \end{array} \tag{10}$$

$$y^l_{ij} \leq t_{ij}, \qquad \forall e_{ij} \in E,\ l \in V \setminus \{r,i,j\}, \tag{11}$$

$$y^l_{il} = t_{il}, \qquad \forall e_{il} \in E, \tag{12}$$

$$y^l_{ij} \geq 0, \qquad \forall e_{ij} \in E,\ l \in V \setminus \{r,i\}, \tag{13}$$

$$t_{ij} \in \{0,1\}, \qquad \forall e_{ij} \in E, \tag{14}$$

where the y^l_{ij} are auxiliary continuous variables that denote the flow from the root vertex to all others. More specifically, y^l_{ij} indicates whether the unique directed path from the root r to vertex l traverses the edge e_{ij}. If the optimal tree $\mathbf{t}^*$ does not contain l and hence such a path does not exist, then $y^l_{ij} = 0$. The first two constraints ensure that there can be at most one path in $\mathbf{t}^*$ from the root to each vertex in the graph. The third one enforces conservation of flow at intermediate vertices $i \in V$. The remaining constraints guarantee that $\mathbf{t}^*$ includes a path from the root to the vertex l passing through edge e_{il} if $\mathbf{t}^*$ contains e_{il}.

Some of our images contain several disconnected curvilinear structures. To avoid having to process them sequentially in a greedy manner, which may result in some branches being "stolen" by the first structure we reconstruct and therefore a suboptimal solution, we connect them all. Assuming we are given a set $V_r \subset V$ of root vertices, one for each underlying curvilinear structure of interest, we create a virtual vertex v and connect it to each $r \in V_r$ by zero cost edge pairs containing all other vertices to which r is connected. To reconstruct multiple disconnected structures at once, we therefore replace r in the constraints of Eqs. 8-13 with v and add the following constraints to the problem

$$t_{vr} = 1, \ \ \forall e_{vr} \in E \ : \ r \in V_r \ , \tag{15}$$

which ensures that all the root vertices will be in the solution.

6.4 Loopy Reconstruction

As discussed above, allowing branches to cross produces cyclic graphs. However, in some cases, such as when delineating the neural structures of the brightfield and brainbow images of Fig. 1, we know that the underlying structure truly is a tree whose topology we will eventually want to recover. This means that we need to be able to distinguish one branch from the other. One approach would be to first recover the subgraph defined by the active edges and then attempt to assess its topology. However, to consistently enforce geometric constraints on branches even at junctions, we do both simultaneously by reasoning in terms of whether consecutive pairs of edges belong to the final delineation or not.

Fig. 8(a) depicts this approach in a specific case. Similarly, consider the vertices labeled i,j,k,l, and m in the graph of Fig. 2(b). In the delineation of Fig. 2(d), edge pairs e_{ijk} and e_{mjl} are both active and vertex j belongs to both branches.

In practice, we seek to minimize the sum of the two negative log-likelihood terms of Eq. 4 as before but modify them to penalize the formation of spurious junctions. In the following, we first describe these two components and then introduce the constraints that allow graph vertices to be shared among separate branches.

6.4.1 Image and Geometry Likelihood Term

Assuming conditional independence of the image evidence given the true values of the random variables T_{ijk}, the first term of Eq. 4 can be rewritten as

$$-\log(P(I,G|\mathbf{T}=\mathbf{t})) = \sum_{e_{ijk} \in F} (c_{ijk} + w_{ijk})\ t_{ijk}, \tag{16}$$

where c_{ijk} is the likelihood ratio of Eq. 6 and w_{ijk} is a learned compatibility function of edges e_{ij} and e_{jk}.

To see this, let us consider two sets $\mathbf{Y}$ and $\mathbf{Z}$ of auxiliary random variables denoting respectively the presence of edges in the optimal solution and compatibility of consecutive edge pairs. More specifically, let $\mathbf{Y} = \{Y_{jk}\}$ be the vector of binary random variables indicating whether edges $\{e_{jk}\}$ truly belong to the underlying curvilinear structures, and $\mathbf{y} = \{y_{jk}\}$ denote their realizations. As will be discussed in Section 6.4.3, we do not allow edges to have more than one active incoming edge pair in the solution. Hence, we have $y_{jk} = \sum_{e_{ij} \in E} t_{ijk} \leq 1$. As a result, for each edge e_{jk} in the solution, there can be at most one parent edge e_{ij} such that $t_{ijk} = 1$.

Let also Z_{jk} be the random variable standing for the parent of e_{jk} and $\mathbf{Z}$ be the vector of all such variables. That is, Z_{jk} can take values from the set $\{e_{ij} \mid e_{ij} \in E \setminus \{e_{kj}\}\}$. There is a one-to-one deterministic relation between $\mathbf{T}$ and $(\mathbf{Y}, \mathbf{Z})$, which we write as

$$T_{ijk} = Y_{jk}\ \ 1(Z_{jk} = e_{ij}), \qquad \forall e_{ijk} \in F \tag{17}$$

where $1(.)$ is an indicator function. We express the first term of Eq. 4 in terms of $\mathbf{Y}$ and $\mathbf{Z}$ as

$$P(I,G|\mathbf{T}=\mathbf{t}) = P(I,G|\mathbf{Y}=\mathbf{y},\mathbf{Z}=\mathbf{z}) \tag{18}$$

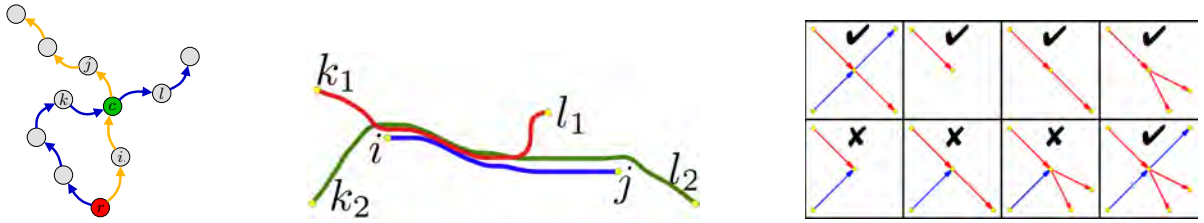
$$= \prod_{e_{jk} \in E} P(I_{jk}, E_{jk}|Y_{jk}=y_{jk}, Z_{jk}=z_{jk}) \tag{19}$$

$$\propto \prod_{e_{jk} \in E} P(Z_{jk}=z_{jk}|Y_{jk}=y_{jk}, I_{jk}, E_{jk}) P(Y_{jk}=y_{jk}|I_{jk}, E_{jk}) \tag{20}$$

$$\propto \prod_{e_{jk} \in E} [P(Z_{jk}=z_{jk}|Y_{jk}=1, I_{jk}, E_{jk}) P(Y_{jk}=1|I_{jk}, E_{jk})]^{y_{jk}} \times [P(Z_{jk}=z_{jk}|Y_{jk}=0, I_{jk}, E_{jk}) P(Y_{jk}=0|I_{jk}, E_{jk})]^{1-y_{jk}} \tag{21}$$

$$\propto \prod_{e_{jk} \in E} \left[\left[\prod_{e_{ij} \in E} (p^c_{ijk})^{t_{ijk}} \right] P(Y_{jk}=1|I_{jk}, E_{jk}) \right]^{y_{jk}} \times \left[\frac{1}{deg^*(v_j, v_k)} \quad P(Y_{jk}=0|I_{jk}, E_{jk}) \right]^{1-y_{jk}} \tag{22}$$

$$\propto \prod_{e_{jk} \in E} \left[\left[\prod_{e_{ij} \in E} (p^c_{ijk})^{t_{ijk}} \right] \left[\frac{p^q_{jk} deg^*(v_j, v_k)}{(1-p^q_{jk})} \right]^{\sum_{e_{ij} \in E} t_{ijk}} \right], \tag{23}$$



(a) (b) (c)

Fig. 8. Reasoning about edge-pairs. (a) A loopy graph with a root vertex r, in red. Allowing vertex c, in green, to be used by the two different branches, denoted by blue and yellow arrows, produces a loopy solution instead of a tree. However, describing the crossing in terms of edge pairs e_{icj} and e_{kcl} being active, and e_{kcj} and e_{icl} being inactive makes it possible to eventually recover the correct tree topology. (b) Handling overlapping edges. (c) Enforcing crossover consistency.

where E_{jk} denotes the set of edges containing e_{jk} and its incoming edges in the graph, and I_{jk} denotes the image evidence around these edges. Eq. 19 is obtained under the assumption that the image evidence around edge pairs are conditionally independent given that we know whether they belong to the curvilinear structures or not. In Eq. 20, we use Bayes' rule and remove the constant terms $P(I_{jk}, E_{jk})$ and $P(Y_{jk}=y_{jk}, Z_{jk}=z_{jk})$, assuming a uniform prior for both. We derive Eq. 21 and 22 by using the fact that $y_{jk}, t_{ijk} \in \{0,1\}$, and substituting $P(Z_{jk}=e_{ij}|Y_{jk}=1, I_{jk}, E_{jk})$ and $P(Z_{jk}=z_{jk}|Y_{jk}=0, I_{jk}, E_{jk})$ respectively with p^c_{ijk} and $\frac{1}{deg^*(v_j,v_k)}$. The term $deg^*(v_j, v_k)$ denotes the number of in-edges of vertex v_j excluding the edge e_{kj}, if it exists. Finally, in Eq. 23, we drop the constant terms, express y_{jk} in terms of t_{ijk}'s, and substitute $P(Y_{jk}=1|I_{jk}, E_{jk})$ with p^q_{jk}. Taking the negative logarithm of Eq. 23 yields

$$
\begin{aligned}
-\log(P(I,G|\mathbf{T}=\mathbf{t})) &= \sum_{e_{ijk}\in F} -\log\left(\frac{p^q_{jk}p^c_{ijk}deg^*(v_j,v_k)}{(1-p^q_{jk})}\right) t_{ijk} \\
&= \sum_{e_{ijk}\in F}\left(-\log\left(\frac{p^q_{jk}}{1-p^q_{jk}}\right) - \log\left(p^c_{ijk}deg^*(v_j,v_k)\right)\right) t_{ijk} \\
&= \sum_{e_{ijk}\in F}\left(-\log\left(\frac{p^q_{jk}}{1-p^q_{jk}}\right) + w_{ijk}\right)\ t_{ijk} \\
&\approx \sum_{e_{ijk}\in F} (c_{ijk} + w_{ijk})\ t_{ijk}\ , \qquad (24)
\end{aligned}
$$

which is the formulation of Eq. 16. The probability p^q_{jk} denotes the likelihood that edge e_{jk} belongs to the curvilinear structure given the associated geodesic path and corresponding image evidence. This is an image-based term that accounts for the quality of the paths associated with the edges. In practice, instead of relying only on the image evidence around the edge e_{jk}, we evaluate the path classifier on a larger neighbourhood including its in-edges $\{e_{ij}\}$ and use the corresponding probabilities in the above summation. Therefore, for each p^q_{jk} term in the above summation, we use the path probability $P(T_{ijk}=1|I_{ijk}, e_{ijk})$ corresponding to the edge pair e_{ijk}, which we obtain using the path classification approach of Section 5.

The term p^c_{ijk} denotes the probability that the edge pair e_{ijk} belongs to the structures given that its target edge e_{jk} belongs to them. Since there must be exactly one active parent for any active edge, these probabilities are normalized over the sum of the probabilities of all incoming edge-pairs so that

$$\sum_{e_{ij}\in E} p^c_{ijk} = 1. \qquad (25)$$

In our experiments, this probability is expressed as a sigmoid function of a distinctive feature, such as directional deviation along e_{ijk}, which helps recovering the right connectivity at crossovers such as the one of Fig. 2.

6.4.2 Prior Term

We take $-\log(P(\mathbf{T}=\mathbf{t}))$ the second likelihood term of Eq. 4 to be a junction prior that penalizes unwarranted bifurcations or terminations, which we write as

$$\sum_{e_{ij}\in E}\left[\sum_{e_{mi}\in E}\lambda_t\, t_{mij} + \sum_{e_{jn}\in E}\gamma_t\, t_{ijn} + \sum_{e_{jn}\in E}\sum_{\substack{e_{jk}\in E\\ k<n}}\delta_t\, t_{ijn}t_{ijk}\right], \qquad (26)$$

where λ_t,γ_t, and δ_t are constants we derive below.

To this end, we treat the graph G as a Bayesian network with latent variables $M_{ij} = \sum_{e_{mij}\in F} T_{mij}$ and $O_{ij} = \sum_{e_{ijn}\in F} T_{ijn}$, which denote the true number of incoming and outgoing edge pairs into and out of edge e_{ij}, respectively. Note that the M_{ij} are binary variables since we limit the number of active incoming edge pairs into an edge to one. Assuming that the O_{ij} variables take values from the set $\{0,1,2\}$ and using a Bayesian network to model the dependencies between the variables M_{ij} and O_{ij}, we get

$$
\begin{aligned}
P(\mathbf{T}=\mathbf{t}) &= \prod_{e_{ij}\in E} P(\mathbf{T}_{ij}=\mathbf{t}_{ij}|O_{ij}=o_{ij})P(O_{ij}=o_{ij}|M_{ij}=m_{ij}) \qquad (27)\\
&\propto \prod_{e_{ij}\in E} P(O_{ij}=o_{ij}|M_{ij}=m_{ij}) \qquad (28)\\
&\propto \prod_{e_{ij}\in E} P(O_{ij}=o_{ij}|M_{ij}=1)^{m_{ij}}\, P(O_{ij}=o_{ij}|M_{ij}=0)^{(1-m_{ij})} \qquad (29)\\
&\propto \prod_{e_{ij}\in E} P(O_{ij}=o_{ij}|M_{ij}=1)^{m_{ij}} \qquad (30)\\
&\propto \prod_{e_{ij}\in E} \Big[P(O_{ij}=0|M_{ij}=1)^{1(o_{ij}=0)} \times \\
&\quad P(O_{ij}=1|M_{ij}=1)^{1(o_{ij}=1)}\ P(O_{ij}=2|M_{ij}=1)^{1(o_{ij}=2)}\Big]^{m_{ij}}, \qquad (31)
\end{aligned}
$$

where $\mathbf{T}_{ij}$ denotes the vector of random variables $T_{ijn}, \quad \forall e_{ijn} \in F$. In this work, we assume that all configurations $\mathbf{T}_{ij}$ are equally likely for an edge e_{ij} given that we know the total number of outgoing edge pairs out of it. Under this assumption, we obtain Eq. 28, which we then decompose into two terms in Eq. 29 using the fact

that $m_{ij} \in \{0,1\}$. In Eq. 30, we remove the second term $P(O_{ij}=o_{ij}|M_{ij}=0)^{(1-m_{ij})}$ in the product because we have $o_{ij}=0$ when $m_{ij}=0$ due to the connectedness constraints we will introduce in the next section, and hence, the term is always equal to 1. Finally, we derive Eq. 31 by expressing the probability $P(O_{ij}=o_{ij}|M_{ij}=1)$ as a product of three admissible event probabilities, namely termination, continuation and bifurcation, only one of which contributes to the product for each edge e_{ij}. The indicator functions are defined as

$$1(o_{ij}=2) = \sum_{e_{jn}\in E} \sum_{\substack{e_{jk}\in E \\ k<n}} t_{ijn}\ t_{ijk}\ , \tag{32}$$

$$1(o_{ij}=1) = \sum_{e_{jn}\in E} t_{ijn} - 2 \sum_{e_{jn}\in E} \sum_{\substack{e_{jk}\in E \\ k<n}} t_{ijn}\ t_{ijk}\ , \tag{33}$$

$$1(o_{ij}=0) = \sum_{e_{mi}\in E} t_{mij} - \sum_{e_{jn}\in E} t_{ijn} + \sum_{e_{jn}\in E} \sum_{\substack{e_{jk}\in E \\ k<n}} t_{ijn}\ t_{ijk}\ . \tag{34}$$

Note that multiplying these functions by $m_{ij} = \sum_{e_{mi}\in E} t_{mij}$ results in themselves since they are all equal to zero when $m_{ij}=0$. Substituting them in Eq. 31 and taking the negative logarithm yields

$$-\sum_{e_{ij}\in E}\Bigg[\sum_{e_{mi}\in E}\log(p^t)t_{mij} + \sum_{e_{jn}\in E}\log\left(\frac{p^c}{p^t}\right)t_{ijn} + \sum_{e_{jn}\in E}\sum_{\substack{e_{jk}\in E \\ k<n}}\log\left(\frac{p^b p^t}{(p^c)^2}\right)t_{ijn}t_{ijk}\Bigg], \tag{35}$$

where $p^t = P(O_{ij}=0|M_{ij}=1)$, $p^c = P(O_{ij}=1|M_{ij}=1)$ and $p^b = P(O_{ij}=2|M_{ij}=1)$ denote respectively the prior probabilities for branch termination, continuation and bifurcation at edge e_{ij}. Note that Eq. 35 always results in positive values. This penalizes the creation of bifurcations and terminations but also continuations, that is, $O_{ij}=1 \wedge M_{ij}=1$, which truly belong to the underlying curvilinear networks. To prevent this, we set the continuation probability p^c to 1 and drop the corresponding terms from Eq. 35. This yields the desired expression

$$-\log(P(\mathbf{T}=\mathbf{t})) = -\sum_{e_{ij}\in E}\Bigg[\sum_{e_{mi}\in E}\log(p^t)t_{mij} + \sum_{e_{jn}\in E}\log\left(\frac{1}{p^t}\right)t_{ijn} + \sum_{e_{jn}\in E}\sum_{\substack{e_{jk}\in E \\ k<n}}\log\left(p^b p^t\right)t_{ijn}t_{ijk}\Bigg], \tag{36}$$

which is the one of Eq. 26.

Note that when the task is to reconstruct a tree structure, this regularization term helps penalize spurious bifurcations and early terminations at branch crossings. However, for loopy networks, it also penalizes legitimate bifurcations that are part of the loops. We therefore do not use it for inherently loopy networks such as blood vessels and roads.

6.4.3 Constraints

In short, minimizing the negative log-likelihood of Eq. 4 amounts to seeking

$$\mathbf{t}^* = \underset{\mathbf{t}\in\mathcal{P}_c}{\operatorname{argmin}} \sum_{e_{ijk}\in F} a_{ijk}t_{ijk} + \sum_{e_{ijk},e_{ijn}\in F} b_{ijkn}t_{ijk}t_{ijn}\ . \tag{37}$$

The criterion being minimized is the sum of the linear and quadratic terms of Eqs. 16 and 36 and the a_{ijk} and b_{ijkn} coefficients are obtained by summing the appropriate terms.

Using the same notation as in Section 6.3, we say that an edge pair e_{ijk} is active in the solution if $t_{ijk}=1$. Similarly, an edge e_{ij} is active if there exist an active incoming edge pair containing it, that is, $\exists e_{lij}\in F : t_{lij}=1$.

We take the feasible region $\mathcal{P}_c$ introduced in Eq. 4 to be the set of subgraphs that satisfy certain connectivity conditions. More specifically, we define four sets of constraints to ensure that the solutions to the minimization of Eq. 37 are such that root vertices are not isolated, branches are edge-disjoint, potential crossovers are consistently handled, and all active edge pairs are connected.

Non-isolated Roots. As described in Section 6.3, we assume that we are given a set $V_r \subset V$ of root vertices, one for each curvilinear structure of interest in the image. To handle multiple disconnected structures, we create a virtual vertex v and connect it to each given root $r\in V_r$ by zero cost edge pairs.

We require the root vertices to be connected to at least one vertex other than the virtual one v and to have no active incoming edge other than e_{vr}. We write

$$\sum_{e_{ri}\in E} t_{vri} \geq 1,\ \ \forall r\in V_r\ \ , \tag{38}$$

$$\sum_{e_{ijr}\in F} t_{ijr} + \sum_{e_{irj}\in F: i\neq v} t_{irj} = 0,\ \ \forall r\in V_r\ \ . \tag{39}$$

Disjoint Edges. For each edge $e_{ij}\in E$, such as the one depicted by Fig. 8(b), we let at most one edge pair be active among all those that are incoming into e_{ij} or e_{ji}. Second we prevent the number of active edge pairs that overlap more than a certain fraction of their mean radius to be more than one. Let p_{ijk} denote the geodesic path corresponding to edge pair e_{ijk}. We write

$$\sum_{\substack{e_{ni}\in E: \\ n\neq j}} t_{nij} + \sum_{\substack{e_{mj}\in E: \\ m\neq i}} t_{mji} \leq 1,\ \ \forall e_{ij}\in E : i\neq v, \tag{40}$$

$$t_{mnl} + t_{ijk} \leq 1, \tag{41}$$

$$\forall e_{mnl},\ e_{ijk}\in F : \left\{\begin{array}{c} (e_{mn}=e_{ji}) \vee (e_{mn}=e_{kj}) \vee (e_{ln}=e_{ij}) \vee \\ \Big((e_{nl}\neq e_{ij}) \wedge (e_{nl}\neq e_{ji}) \wedge (e_{nl}\neq e_{jk}) \wedge \\ (e_{nl}\neq e_{kj}) \wedge (e_{mn}\neq e_{ij}) \wedge (e_{mn}\neq e_{ji}) \wedge \\ (e_{mn}\neq e_{jk}) \wedge (e_{mn}\neq e_{kj}) \wedge \\ \Big(l(p_{mnl}\cap p_{ijk}) > \alpha\bar{r}(p_{mnl}\cap p_{ijk})\Big) \wedge \\ \Big(l(p_{mnl}) < l(p_{ijk})\Big)\Big) \end{array}\right\}, \tag{42}$$

where $l(.)$ and $\bar{r}(.)$ denote the length and mean radius of a path respectively. α is a constant value that determines the allowed extent of the overlap between the geodesic paths of the edges. It is set to 3 in all our experiments. In the example depicted by the figure above, among all the edge pairs incoming to the edges e_{ij}, $e_{k_1l_1}$ and $e_{k_2l_2}$, only one can be active in the final solution.

For those curvilinear structures that are inherently trees, these constraints make their recovery from the resulting subgraph possible by starting from the root vertices and following the active edge pairs along the paths that lead to the terminal vertices.

Crossover Consistency. A potential crossover in G is a vertex that is adjacent to at least four other vertices and whose in- and out-degrees are greater than one. A consistent solution containing such a vertex is then defined as a one whose branches do not terminate at the vertex if

its in-degree in the solution is greater than one. Fig. 8(c) illustrates consistent configurations denoted by a tick mark and inconsistent ones denoted by a cross. We express this as

$$\sum_{\substack{e_{ki}\in E:\\ k\neq j}} t_{kij} + \sum_{\substack{e_{lm}\in E:\\ l\neq q, l\neq j}} t_{lmq} - \sum_{\substack{e_{ju}\in E:\\ u\neq i}} t_{iju} \leq 1 \ , \tag{42}$$
$$\forall e_{ij} \in E \quad \forall e_{mq} \in C(j) : m \neq i$$
$$C(j) = \{e_{nk} \in E \mid k = j \vee (j \in p_{nk}, n \neq j, k \neq j)\}$$

These constraints are only active when dealing with structures that inherently are trees, such as dendritic and axonal arbors. For inherently loopy ones, such as road and vascular networks, we deactivate them to allow creation of junctions that are parts of legitimate cycles.

Connectedness via Network Flow. We require all the active edge pairs to be connected to the virtual vertex v. An edge pair e_{ijk} is said to be connected if there exists a path in G, starting at v and containing e_{ijk}, along which all the edge pairs are active.

Let y^l_{ij} $(i \neq l)$ be a non-negative continuous flow variable which denotes the number of distinct directed paths that connect v to vertex l, and traverse the edge e_{ij} in the solution. This gives rise to the following constraint set

$$\sum_{e_{vj}\in E} y^l_{vj} = \sum_{e_{il}\in E} y^l_{il}, \ \ \forall l \in V \setminus \{v\} \ , \tag{43}$$
$$\sum_{e_{vj}\in E} y^l_{vj} \leq deg^-(l), \ \ \forall l \in V \setminus \{v\} \ , \tag{44}$$
$$\sum_{e_{ij}\in E} y^l_{ij} = \sum_{e_{jk}\in E} y^l_{jk}, \ \ \forall j, l \in V \setminus \{v\} : j \neq l \ , \tag{45}$$
$$y^l_{il} \geq t_{ilk}, \ \ \forall e_{ilk} \in F \ , \tag{46}$$
$$y^l_{il} = \sum_{e_{ki}\in E} t_{kil}, \ \ \forall e_{il} \in E : i \neq v \ , \tag{47}$$
$$y^l_{ij} \leq deg^-(l) \sum_{e_{ki}\in E} t_{kij}, \ \ \substack{\forall e_{ij}\in E: i\neq v,\\ \forall l\in V\setminus\{v,i,j\}} \ , \tag{48}$$
$$y^l_{ij} \geq y^l_{jk} + deg^-(l)(t_{ijk} - 1), \ \ \substack{\forall e_{ijk}\in F,\\ \forall l\in V\setminus\{v\}: j\neq l, i\neq l} \ , \tag{49}$$

where $deg^-(.)$ is the in degree of a vertex. The first two constraints guarantee that the amount of flow outgoing from the virtual vertex v to a vertex l is equal to the incoming flow to l, which must be smaller than the in degree of l. The constraints in Eq. 45 impose conservation of flow at intermediate vertices of the directed paths from v to l. The following three constraints bind the flow variables to the binary ones, ensuring that a connected network formed by the non-zero flow variables is also connected in the active edge pair variables. Note that, since we are looking for possibly cyclic subgraphs, there can be multiple paths incoming to a vertex. Hence, unlike the flow variables of Eqs. 8-13, which are bounded by one, the ones defined here have no upper bounds.

Finally, the constraints introduced in Eq. 49 ensure that active edge pairs do not form a cycle disconnected from the virtual vertex. This is achieved by imposing conservation of flow on active edge pairs. These constraints are required to ensure that a tree topology can be recovered from a potentially loopy solution by tracing the active edge pairs from the roots $r \in V_r$ to the branch terminals.

6.5 Optimization

The two IP formulations introduced in Sections 6.3 and 6.4 are generalizations of the Minimum Arborescence Problem, which is known to be NP-Hard [48]. Nevertheless, we obtained globally optimal or near-optimal solutions for all the problem instances discussed in the results section using the branch-and-cut algorithm [49]. More precisely, the objective values of the solutions we found are guaranteed to be within a small distance $2e^{-16}$ of the true optimum.

However, the branch-and-cut procedure does not scale well to large graphs especially when we use the network flow formulation of Eqs. 43-49. To overcome this, we reduce the size of the graphs by pruning or merging some of the edge pairs based on their assigned costs, which we describe in Appendix D. Furthermore, we introduce a new set of connectivity constraints expressed only in the edge pair variables t_{ijk}, which we describe in the remainder of this section. This new approach eliminates the need for adding the large number, $|V|\cdot|E|$, of auxiliary flow variables y^l_{ij} and their constraints to the optimization.

More specifically, we require every subset of active edge pairs to be connected to the virtual vertex v. To this end, we extend the *enhanced connectivity constraints* for edges introduced in [48] to edge pairs by imposing

$$\sum_{\substack{e_{nl}\in H,\\ e_{mn}\in E\setminus(H\cup e_{ij})}} t_{mnl} \geq \sum_{e_{ki}\in H} t_{kij}, \ \ \substack{\forall H\subset E\setminus\{e_{vr}|r\in V_r\}, \ \forall e_{ij}\in E\setminus H :\\ (\forall e_{uw}\in H,\, e_{wu}\in H)} \tag{50}$$

Note that, since we consider every possible subset H of graph edges, the number of constraints grows exponentially with the number of edges and is too large in practice to be dealt with exhaustively. To make the problem tractable, we therefore take a lazy constraint approach, which requires only a small but sufficient fraction of the constraints to be used. This is done iteratively by identifying those constraints of Eq. 50 whose elimination gives rise to disconnections and adding them to the problem.

More specifically, we start by creating a reduced IP model with the cost function of Eq. 37 and a subset of the constraints given by Eqs. 38-42. We then run the branch-and-cut algorithm on this model and check for connectivity violations in the intermediate solutions it produces. These violations are found in linear time by identifying those connected components which are not connected to the virtual vertex v by a directed path on which all edge pairs are active.

Let C be the set of components and E_c be the set of edges for $c \in C$. We add a *lazy constraint* given by Eq. 50 for each such component by randomly selecting an edge $e_{ij} \in E_c$ and forming a new set of edges $H = \{e_{kl} \mid (e_{kl} \in E_c \setminus e_{ij}) \vee (e_{lk} \in E_c \setminus e_{ij})\}$. We then continue the branch-and-cut optimization with this new model.

We repeat this procedure for every intermediate solution until no more constraints are violated and hence the solution is globally optimal. As will be shown in Section 7.4, it

requires only a small fraction of the original constraints to be added in practice and yields significantly faster run times compared to the network flow formulation.

7 RESULTS

In this section, we first introduce the datasets, baselines, and metrics used to validate our approach. We then quantify the performance of our approach for both tree and loopy networks. Finally, we briefly discuss run-times on multi-core machines.

7.1 Datasets and Parameters

We evaluated our approach on the five datasets depicted by Fig. 1 and described in detail below:

- *Aerial*: Aerial images of road networks obtained from Google maps. We used 21 grayscale versions of these images to train our path classifier and 10 for testing.
- *Confocal-Vessels*: Two image stacks of blood vessel networks acquired with a confocal microscope. We used a portion of one for training and both for testing. We only considered the red channel of these stacks since it is the only one used to label the blood vessels.
- *Brightfield*: Six image stacks were acquired by bright-field microscopy from biocytin-stained rat brains. We used three for training and three for testing.
- *Confocal-Axons*: 3D confocal microscopy image stacks of Olfactory Projection Fibers (OPF) of the Drosophila fly taken from the DIADEM competition [2]. The dataset consists of 8 image stacks of axonal trees, which we split into a training and a testing set, leaving 4 stacks in the latter.
- *Brainbow*: Neurites of this dataset were acquired by targeting mice primary visual cortex using the brainbow technique [1] so that each has a distinct color. We used one image stack for training and three for testing.

For all five datasets, we used a semi-automated delineation tool [50] we developed to obtain the ground truth tracings. We built the tubular graphs using a seed selection distance of $d = 5\Delta_I$ for the *Confocal-Axons* dataset and $d = 20\Delta_I$ for the other four, and a seed linking distance of $l(d) = 5d$, where Δ_I denotes the minimum voxel spacing. In all our experiments, we used the same parameters to compute the HGD descriptors: segment length $2\Delta_I$; segment sampling step $0.5\Delta_I$ implying a 75% overlap; $B = 9$ histogram bins; $R = 3$ radius intervals; radius margin $\beta = 0.6$, and a BoW model of 300 codewords.

We estimated the branch bifurcation and termination probabilities p^b and p^t introduced in Section 6.4.2 from the training data by first counting the total number of graph edges that intersect with the ground truth tracings and then finding the ratio of the number of terminating and bifurcating edges to this number. For example, in the Brainbow dataset this yields $p^b = 0.07$ and $p^t = 0.20$, and in the Brightfield dataset $p^b = 0.13$ and $p^t = 0.16$.

7.2 Baselines and Metrics

We compared our tree and loopy reconstruction approaches, which we refer to as **Arbor-IP** and **Loopy-IP**, to several state-of-the-art algorithms. They are **APP2**, the pruning-based approach of [51], **Osnake** the active contour algorithm of [18], **NS** the NeuronStudio software [13], **Focus** the focus-based depth estimation method of [52], and our own earlier **kMST** technique [8], the last two of which were finalists in the DIADEM competition. For all these algorithms, we used the implementations provided by their respective authors with default parameters.

Our algorithm requires the tree roots to be given. There are ways to do this automatically but, since this is not the focus of this paper, we supplied them manually. Since the interfaces of **APP2**, **NS**, and **Osnake** allow the user to specify a single root node, we did so both for these methods and ours for the Confocal-Axons dataset.

For quantitative evaluation when the root nodes are specified, we used the DIADEM metric [2], which is designed to compare topological accuracy of a reconstructed tree against a ground truth tree. Misconnections in the reconstruction are weighted by the size of the subtree to which the associated ground truth connection leads. As a result, connections that are close to the given root vertices are given more weight than those that are close to the branch terminals.

However, because the available implementations of our baselines do not allow the user to provide the multiple root vertices required in the case of the Brightfield and Brainbow datasets, we also used the NetMets [53] measure. It does not depend heavily on roots at the cost of being more local than the DIADEM metric and not explicitly taking into account the network topology. As the DIADEM metric, it takes as input the reconstruction, the corresponding ground truth tracings, and a sensitivity parameter σ, which is set to $3\Delta_I$ in all our experiments.

To disentangle the respective contributions of the two novel components of our approach – the path classifier of Section 5 and the IP formalism of Section 6 – we implemented two additional algorithms that minimize the following simplified versions of Eq. 3.

$$\sum_{e_{ij} \in E} -\log \frac{p_{ij}}{1 - p_{ij}} \, t_{ij} \;, \tag{51}$$

$$\sum_{e_{ijk} \in F} -\log \frac{p_{ijk}}{1 - p_{ijk}} \, t_{ij} t_{jk} \;, \tag{52}$$

where the p_{ij} and p_{ijk} are the probabilities returned by the path classifier on edges and edge pairs respectively. The first one relies on first computing a minimum spanning tree under the cost of Eq. 51 and then pruning it in an optimal way. A detailed explanation of the pruning procedure can be found in Appendix E. The second algorithm relies on our earlier k-Minimum Spanning Tree algorithm [8], which minimizes a quadratic cost function. In other words, these two algorithms take advantage of the path classifier without relying on the IP framework. We will refer to them as **MST+PC** and **kMST+PC** respectively.

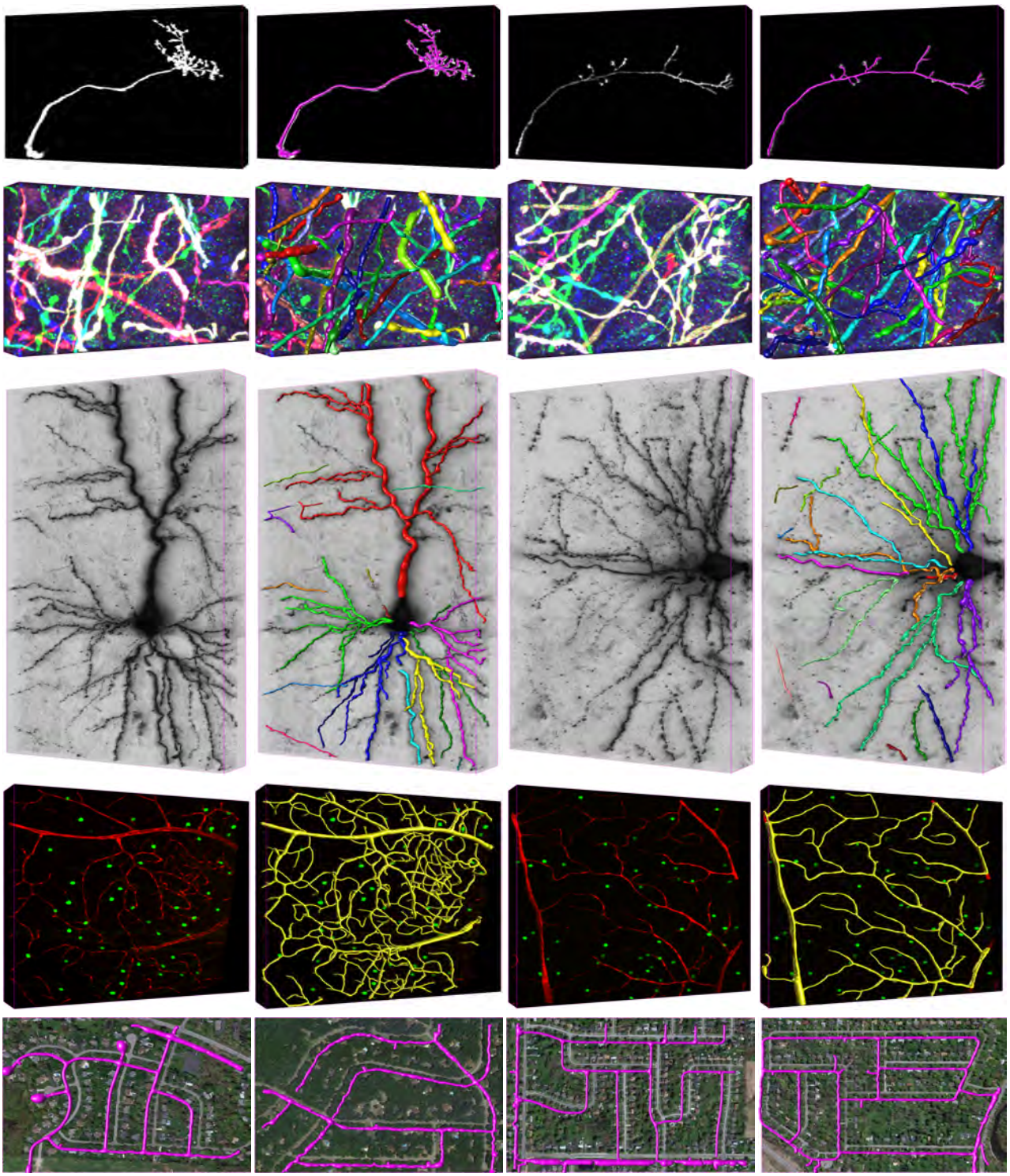


Fig. 9. Delineation results, best viewed in color. **Top Rows:** For each dataset, two minimal or maximal projections and overlaid delineation results. Each connected curvilinear network is shown in a distinct color. **Bottom Row:** Four road images with final delineations shifted and overlaid to allow comparisons. The renderings are created using the Vaa3D software [54].

7.3 Quantitative Evaluation

We now compare our results to those of the baselines discussed above on our four test datasets.

7.3.1 Neural Structures

The neurites of the *Confocal-Axons*, *Brightfield* and *Brainbow* datasets form tree structures without cycles. However, in the latter two datasets, because of the low z-resolution, disjoint branches appear to cross, introducing false loops. As described in Section 6.4, these loops can be successfully

	Loopy-IP	**Arbor-IP**	**kMST** [8]	**NS** [13]	**Osnake** [18]	**APP2** [51]
OPF4	**0.91**	0.89	0.87	0.58	0.00	0.67
OPF6	**0.91**	0.90	0.90	0.65	0.80	0.82
OPF7	**0.94**	0.93	0.91	0.42	0.68	0.76
OPF8	**0.90**	**0.90**	0.74	0.58	0.69	0.63

TABLE 1

Reconstruction accuracy in terms of the DIADEM [2] metric on four test stacks of the *Confocal-Axons* dataset. Each row corresponds to an image stack denoted by OPi. Higher scores are better.

	BRBW1	BRBW2	BRBW3	BRF1	BRF2	BRF3
Loopy-IP	**0.65**	**0.56**	**0.66**	**0.76**	**0.50**	**0.80**
Arbor-IP	0.44	0.32	0.45	0.59	0.32	0.71
kMST [8]	0.26	0.16	0.25	0.51	0.28	0.47
kMST+PC	0.51	0.21	0.30	0.49	0.36	0.59
MST+PC	0.46	0.40	0.49	0.44	0.37	0.55

TABLE 2

DIADEM scores for the *Brainbow* (BRBWi columns) and *Brightfield* (BRFi columns) datasets.

identified and a tree topology can be recovered from a loopy solution by tracing the active edge pairs from the root vertices to the branch terminals.

Fig. 9 shows two tree reconstructions we obtained for each of these datasets using the formulation of Section 6.4. Additional ones are supplied as supplementary material. In Tables 1, 2, and 3, we provide scores in terms of either the DIADEM [2] or the NetMets [53] score both for our own approach and those of [8], [13], [18], [51], [52], as discussed in Section 7.2.

Given that the DIADEM scores should be high and the NetMets rates low, one can see that our **Loopy-IP** approach consistently outperforms these baselines. As can be seen in Table 1, for the Confocal-Axons datasets **Arbor-IP** and **Loopy-IP** scores are quite similar because the z-resolution is sufficiently high so that the false loops that **Loopy-IP** is designed to avoid are rare. By contrast, for the other two datasets whose axial (z-) resolution is much lower, **Loopy-IP** clearly performs much better than **Arbor-IP**.

Interestingly, as can be seen in Tables 2 and 3, the **MST+PC** and **kMST+PC** algorithms are already competitive, while being less consistent than the **Arbor-IP** and **Loopy-IP** ones. In other words, the effectiveness and versatility of our algorithm largely comes from using the path classifier but the IP framework is required to get the most out of it.

7.3.2 Roads and Blood Vessels

The roads and blood vessels of the *Aerial* and *Confocal-Vessel* datasets form networks, in which there are many real cycles. Many road networks of the *Aerial* dataset are partially occluded by trees or shadows cast by them, making the delineation task challenging. However, as can be seen in the last row of Fig. 9, in spite of the occlusions, the road networks are recovered almost perfectly. The main errors are driveways that are treated as very short roads, thin side streets mistaken as high-ways and a few roads dead-ending because the connecting path to the closest junction is severely occluded. The first one could be addressed by introducing a semantic threshold on short overhanging segments while the latter two would require a much more sophisticated semantic understanding.

The quality of the blood vessel delineations depicted by the second row is much harder to assess on the printed page but becomes clear when looking at the rotating volumes available as supplementary material.

	BRF1				BRF2				BRF3			
Loopy-IP	**0.05**	0.29	0.71	**0.65**	0.11	**0.29**	**0.81**	**0.78**	**0.07**	**0.28**	0.77	**0.70**
kMST [8]	0.10	0.44	0.79	0.88	0.11	0.53	0.84	0.91	0.13	0.35	0.81	0.92
kMST+PC	0.06	0.47	0.83	0.84	**0.09**	0.47	0.90	0.91	**0.07**	0.60	0.89	0.89
MST+PC	0.11	**0.24**	**0.62**	0.83	0.13	0.31	0.86	0.93	0.11	0.29	**0.70**	0.87
Focus [52]	0.39	0.54	0.75	1.00	0.49	0.53	0.90	1.00	0.38	0.46	0.74	1.00
Osnake [18]	0.66	0.63	0.98	0.99	0.66	0.59	0.99	1.00	0.69	0.38	0.95	0.99
APP2 [51]	0.68	0.64	1.00	1.00	0.63	0.54	1.00	1.00	0.65	0.49	1.00	1.00

TABLE 3

NetMets [53] scores for the *Brightfield* dataset. The NetMets software outputs four values for each trial, which are geometric False Positive Rate (FPR), geometric False Negative Rate (FNR), connectivity FPR, and connectivity FNR, from left to right. Here, lower is better.

	BRBW1	BRBW2	BRBW3	BRF1	BRF2	BRF3	CONV1	CONV2
Network Flow	166.7	132.9	291.1	379.3	17.2	**4.9**	8.4	106.8
Subset + Lazy	**32.9**	**4.9**	**19.5**	**36.6**	**10.9**	7.7	**0.6**	**4.2**
# Lazy Constr.	1008	892	1092	1556	1165	2259	972	5175

TABLE 4

Run-times for the network flow and subset formulations of Sections 6.4.3 and 6.5 respectively. The run-times are averaged over 3 runs per image and measured in minutes on a multi-core machine. While the flow constraints are imposed all at once, the subset ones are added lazily as described in the Section 6.5. The last row shows the total number of lazy constraints added in the branch-and-cut search before the global optimum is reached. CONVi denotes an image stack from the *Confocal-Vessels* dataset.

7.4 Run Time Analysis

Table 4 provides running times in minutes per image for our network flow and subset approaches described in Section 6.4.3. We solve both problems to optimality, which results in exactly the same solution. As described in Section 6.5, the latter is solved iteratively by identifying violations of the connectivity constraints of Eq. 50 and adding them to the problem. As can be seen from the table, in practice, only a small fraction of these constraints are needed to ensure connectivity, and the approach results in significant speed-ups, especially on those problem instances that are hard-to-solve using the network flow formulation.

8 Conclusion

We have proposed a novel graph-based approach to delineating complex curvilinear structures that lets us find the

desired network as the global optimum of a well- designed objective function. Unlike most earlier techniques, it explicitly handles the fact that the structures may be cyclic and builds graphs in which vertices may belong to more than one branch. Another key ingredient is a classification-based approach to scoring the quality of paths. This combination overall allows us to outperform state-of-the-art methods without having to manually tune the algorithm parameters for each new dataset.

9 Acknowledgements

This work was supported in part by a grant from the Swiss National Science Foundation (SNSF) and in part by the Swiss Center for Electronics and Microtechnology (CSEM). The authors also wish to thank Luke Bogart, Maximilian Joesch, Karim Ali, Horesh Ben Shitrit, Takao Hensch, Felix Schürmann, Markus Meister and Jeff Lichtman for generously sharing their image data and insights with us.

References

[1] J. Livet, T. Weissman, H. Kang, R. Draft, J. Lu, R. Bennis, J. Sanes, and J. Lichtman, "Transgenic Strategies for Combinatorial Expression of Fluorescent Proteins in the Nervous System," *Nature*, vol. 450, no. 7166, pp. 56–62, 2007.

[2] G. A. Ascoli, K. Svoboda, and Y. Liu, "Digital Reconstruction of Axonal and Dendritic Morphology Diadem Challenge," 2010, http://diademchallenge.org/.

[3] H. Peng, F. Long, T. Zhao, and E. Myers, "Proof-Editing is the Bottleneck of 3D Neuron Reconstruction: the Problem and Solutions," *Neuroinformatics*, vol. 9, no. 2, pp. 103–105, 2011.

[4] Y. Wang, A. Narayanaswamy, C. Tsai, and B. Roysam, "A Broadly Applicable 3D Neuron Tracing Method Based on Open-Curve Snake," *Neuroinformatics*, vol. 9, no. 2-3, pp. 193–217, 2011.

[5] P. Chothani, V. Mehta, and A. Stepanyants, "Automated Tracing of Neurites from Light Microscopy Stacks of Images," *Neuroinformatics*, vol. 9, pp. 263–278, 2011.

[6] E. Bas and D. Erdogmus, "Principal Curves as Skeletons of Tubular Objects - Locally Characterizing the Structures of Axons," *Neuroinformatics*, vol. 9, no. 2-3, pp. 181–191, 2011.

[7] 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*, vol. 9, pp. 247–261, May 2011.

[8] E. Turetken, G. Gonzalez, C. Blum, and P. Fua, "Automated Reconstruction of Dendritic and Axonal Trees by Global Optimization with Geometric Priors," *Neuroinformatics*, vol. 9, no. 2-3, pp. 279–302, 2011.

[9] A. Choromanska, S. Chang, and R. Yuste, "Automatic Reconstruction of Neural Morphologies with Multi-Scale Graph-Based Tracking," *Frontiers in Neural Circuits*, vol. 6, no. 25, 2012.

[10] F. Benmansour and L. Cohen, "Tubular Structure Segmentation Based on Minimal Path Method and Anisotropic Enhancement," *International Journal of Computer Vision*, vol. 92, no. 2, pp. 192–210, 2011.

[11] E. Turetken, F. Benmansour, and P. Fua, "Automated Reconstruction of Tree Structures Using Path Classifiers and Mixed Integer Programming," in *Conference on Computer Vision and Pattern Recognition*, June 2012.

[12] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua, "Reconstructing Loopy Curvilinear Structures Using Integer Programming," in *Conference on Computer Vision and Pattern Recognition*, June 2013.

[13] S. Wearne, A. Rodriguez, D. Ehlenberger, A. Rocher, S. Henderson, and P. Hof, "New Techniques for Imaging, Digitization and Analysis of Three-Dimensional Neural Morphology on Multiple Scales," *Neuroscience*, vol. 136, no. 3, pp. 661–680, 2005.

[14] H. Cuntz, F. Forstner, A. Borst, and M. Häusser, "One Rule to Grow Them All: A General Theory of Neuronal Branching and Its Practical Application," *PLoS Computational Biology*, vol. 6, no. 8, p. 1000877, 2010.

[15] D. R. Myatt, T. Hadlington, G. A. Ascoli, and S. J. Nasuto, "Neuromantic - from Semi Manual to Semi Automatic Reconstruction of Neuron Morphology," *Frontiers in Neuroinformatics*, vol. 6, no. 4, 2012.

[16] M. Longair, D. A. Baker, and J. D. Armstrong, "Simple Neurite Tracer: Open Source Software for Reconstruction, Visualization and Analysis of Neuronal Processes," *Bioinformatics*, vol. 27, no. 17, pp. 2453–2454, 2011.

[17] H. Peng, F. Long, and G. Myers, "Automatic 3D Neuron Tracing Using All-Path Pruning," *Bioinformatics*, vol. 27, no. 13, pp. 239–247, 2011.

[18] Y. Wang, A. Narayanaswamy, and B. Roysam, "Novel 4D Open-Curve Active Contour and Curve Completion Approach for Automated Tree Structure Extraction," in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1105–1112.

[19] George Mason University, "NeuroMorpho.Org," http://neuromorpho.org.

[20] C. Weaver, P. Hof, S. Wearne, and L. Brent, "Automated Algorithms for Multiscale Morphometry of Neuronal Dendrites," *Neural Computation*, vol. 16, no. 7, pp. 1353–1383, 2004.

[21] T. Lee, R. Kashyap, and C. Chu, "Building Skeleton Models via 3D Medial Surface Axis Thinning Algorithms," *Computer Vision, Graphics, and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[22] K. Palagyi and A. Kuba, "A 3D 6-Subiteration Thinning Algorithm for Extracting Medial Lines," *Pattern Recognition*, vol. 19, no. 7, pp. 613–627, 1998.

[23] M. Pool, J. Thiemann, A. Bar-Or, and A. E. Fournier, "Neuritetracer: A Novel Imagej Plugin for Automated Quantification of Neurite Outgrowth," *Journal of Neuroscience Methods*, vol. 168, no. 1, pp. 134–139, 2008.

[24] J. Xu, J. Wu, D. Feng, and Z. Cui, "Dsa Image Blood Vessel Skeleton Extraction Based on Anti-Concentration Diffusion and Level Set Method," *Computational Intelligence and Intelligent Systems*, vol. 51, pp. 188–198, 2009.

[25] H. Cai, X. Xu, J. Lu, J. Lichtman, S. Yung, and S. Wong, "Repulsive Force Based Snake Model to Segment and Track Neuronal Axons in 3D Microscopy Image Stacks," *NeuroImage*, vol. 32, no. 4, pp. 1608–1620, August 2006.

[26] Z. Vasilkoski and A. Stepanyants, "Detection of the Optimal Neuron Traces in Confocal Microscopy Images," *Journal of Neuroscience Methods*, vol. 178, no. 1, pp. 197–204, 2009.

[27] A. Can, H. Shen, J. Turner, H. Tanenbaum, and B. Roysam, "Rapid Automated Tracing and Feature Extraction from Retinal Fundus Images Using Direct Exploratory Algorithms," *Transactions on Information Technology in Biomedicine*, vol. 3, no. 2, pp. 125–138, June 1999.

[28] K. Al-Kofahi, S. Lasek, D. Szarowski, C. Pace, G. Nagy, J. Turner, and B. Roysam, "Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks," *Transactions on Information Technology in Biomedicine*, vol. 6, no. 2, pp. 171–187, 2002.

[29] T. Yedidya and R. Hartley, "Tracking of Blood Vessels in Retinal Images Using Kalman Filter," in *Digital Image Computing: Techniques and Applications*, 2008, pp. 52–58.

[30] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman, "Blood Vessel Segmentation Methodologies in Retinal Images – A Survey," *Computer Methods and Programs in Biomedicine*, vol. 108, no. 1, pp. 407–433, 2012.

[31] A. Mukherjee and A. Stepanyants, "Automated Reconstruction of Neural Trees Using Front Re-Initialization," in *SPIE*, 2012.

[32] M. Law and A. Chung, "Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux," in *European Conference on Computer Vision*, 2008.

[33] ——, "An Oriented Flux Symmetry Based Active Contour Model for Three Dimensional Vessel Segmentation," in *European Conference on Computer Vision*, 2010.

[34] L. Domanski, C. Sun, R. Hassan, P. Vallotton, and D. Wang, "Linear Feature Detection on GPUs," in *Digital Image Computing: Techniques and Applications*, 2010.

[35] J. Wegner, J. Montoya-Zegarra, and K. Schindler, "A Higher-Order CRF Model for Road Network Extraction," in *Conference on Computer Vision and Pattern Recognition*, 2013.

[36] D. Fan, "Bayesian Inference of Vascular Structure from Retinal Images," Ph.D. dissertation, Dept. of Computer Science, U. of Warwick, Coventry, UK, 2006.
[37] K. Sun, N. Sang, and T. Zhang, "Marked Point Process for Vascular Tree Extraction on Angiogram," in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 467–478.
[38] E. Tempel, R. Stoica, V. Martínez, L. Liivamägi, G. Castellan, and E. Saar, "Detecting filamentary pattern in the cosmic web: a catalogue of filaments for the SDSS," *Monthly Notices of the Royal Astronomical Society*, vol. 438, no. 4, pp. 407–433, 2014.
[39] M. Fischler, J. Tenenbaum, and H. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," *Computer Vision, Graphics, and Image Processing*, vol. 15, no. 3, pp. 201–223, March 1981.
[40] H. Li and A. Yezzi, "Vessels as 4-D Curves: Global Minimal 4D Paths to Extract 3-D Tubular Surfaces and Centerlines," *IEEE Transactions on Medical Imaging*, vol. 26, no. 9, pp. 1213–1223, 2007.
[41] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken, "Ridge Based Vessel Segmentation in Color Images of the Retina," *IEEE Transactions on Medical Imaging*, 2004.
[42] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua, "Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux," in *International Conference on Computer Vision*, December 2013.
[43] J. A. Sethian, *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.
[44] D. Weinland, M. Ozuysal, and P. Fua, "Making Action Recognition Robust to Occlusions and Viewpoint Changes," in *European Conference on Computer Vision*, 2010.
[45] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Conference on Computer Vision and Pattern Recognition*, 2005.
[46] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.
[47] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT press, 2004.
[48] C. Duhamel, L. Gouveia, P. Moura, and M. Souza, "Models and Heuristics for a Minimum Arborescence Problem," *Networks*, vol. 51, no. 1, pp. 34–47, 2008.
[49] Gurobi, "Gurobi Optimizer," 2012, http://www.gurobi.com/.
[50] E. Turetken, F. Benmansour, and P. Fua, "Semi-Automated Reconstruction of Curvilinear Structures in Noisy 2D Images and 3D Image Stacks," EPFL-182839, Tech. Rep., 2013.
[51] H. Xiao and H. Peng, "APP2: Automatic Tracing of 3D Neuron Morphology Based on Hierarchical Pruning of a Gray-Weighted Image Distance-Tree," *Bioinformatics*, vol. 29, no. 11, pp. 1448–1454, 2013.
[52] A. Narayanaswamy, Y. Wang, and B. Roysam, "3D Image Pre-Processing Algorithms for Improved Automated Tracing of Neuronal Arbors," *Neuroinformatics*, vol. 9, no. 2-3, pp. 219–231, 2011.
[53] D. Mayerich, C. Bjornsson, J. Taylor, and B. Roysam, "Netmets: Software for Quantifying and Visualizing Errors in Biological Network Segmentation," *BMC Bioinformatics*, vol. 13, 2012.
[54] H. Peng, Z. Ruan, F. Long, J. Simpson, and E. Myers, "V3D Enables Real-Time 3D Visualization and Quantitative Analysis of Large-Scale Biological Image Data Sets," *Nature Biotechnology*, vol. 28, no. 4, pp. 348–353, 2010.

Engin Türetken received his Ph.D. in Computer Science in 2013 from EPFL. He is currently a postdoctoral researcher at CSEM in Neuchâtel. His research interests include computer vision, biological image analysis, pattern recognition, graph theory, and combinatorial optimization.

Fethallah Benmansour received his Ph.D. degree in applied mathematics from the University of Paris-Dauphine, France, in 2009. He is now a Senior Researcher at Roche. His research interests include computer vision, variational methods, and machine learning for biomedical applications.

Bjoern Andres holds a Ph.D. in physics from the University of Heidelberg. His junior research group Combinatorial Image Analysis at the Max Planck Institute for Informatics develops combinatorial optimization techniques for image analysis.

Przemysław Głowacki is a Ph.D. candidate at EPFL. His research interests include biomedical imaging, automatic reconstructing of neuronal networks, and analysis of aerial images.

Hanspeter Pfister is An Wang Professor of Computer Science and Director of the Institute for Applied Computational Science at the Harvard School of Engineering and Applied Sciences. His research in visual computing lies at the intersection of Visualization, Computer Graphics, and Computer Vision. It spans a wide range of topics, including bio-medical visualization, image and video analysis, 3D fabrication, and data science.

Pascal Fua is a Professor of Computer Science at EPFL. His research interests include shape modeling and motion recovery from images, analysis of microscopy images, and Augmented Reality. He is an IEEE Fellow and has been an Associate Editor of the IEEE journal Transactions for Pattern Analysis and Machine Intelligence.

Reconstructing Curvilinear Networks using Path Classifiers and Integer Programming - Appendices -

Engin Türetken, Fethallah Benmansour, Bjoern Andres, Przemysław Głowacki, Hanspeter Pfister, *Senior Member, IEEE*, and Pascal Fua, *Fellow, IEEE*

In these appendices, we first present existing software tools for reconstructing curvilinear networks. We then perform a parameter study for the path classification step. Next, we derive the likelihood term of Section 6.3 and introduce a procedure for reducing the size of the tubular graphs. Finally, we give some details about the minimum spanning tree pruning method we used as a baseline in the results section.

APPENDIX A: SOFTWARE TOOLS FOR MANUAL AND SEMI-AUTOMATED DELINEATION

Manual delineation tools require the cylindrical compartments to be sequentially provided by a user starting from the structure roots, such as soma for neural micrographs or optic disc for fundus scans, and ending at branch tips [1], [2], [3], [4]. Since they tend to be labor-intensive and time consuming, these tools are most often used for correcting the reconstructions obtained by more automated ones.

Semi-automated techniques, on the other hand, use a *tubularity* measure to reduce user input to a handful of carefully selected seed points to be manually specified. These points are linked with paths that tightly follow high tubularity pixels or voxels. Most methods employ an interactive and sequential procedure, where the user specifies one seed point at a time and the algorithm constructs a high-tubularity path that links the given seed to the current reconstruction [5], [6], [7], [3]. Other techniques include those that prompt the user for a new seed only when the algorithm is stuck [8] or require only the root and branch endpoints [9], [10], [4].

In Table 1, we list some of the existing software tools that implement these approaches. The table includes several key features such as the level of automation, the type of outputs obtained and the tubularity measure used. Note that there are only a few non-commercial software packages that are both automated and able to handle 3D imagery [11], [12], [13], [14], [15].

Tool	Type	Output	Tubularity	Platf.	Lang.	Dim.	Color	Radii	3D P.	3D V.	Code	Free
AxonTracker [8]	S	T: Greedy tracking	Gradient Vector Flow (GVF) and smoothness prior	W	C++	3D	no	no	yes	yes	no	yes
Imaris [4]	M/S/A	T: Fast marching minimal path in image space	Hessian functional	M/W	X	2D/3D	no	yes	yes	yes	no	no
Farsight [16], [11]	A	T: Open-curve snake	GVF and regularization priors	L/M/W	C++	3D	no	yes	yes	yes	yes	yes
Geodesic-SNT [17]	S	T: Fast marching minimal path with a color prior in scale space	Multi-directional oriented flux (MDOF)	L/M/W	C++ and Java	2D/3D	yes	yes	yes	yes	yes	yes
HCA-Vision [18]	S	S	Directional template responses	W	C, C++ and C#	2D	no	no	N/A	N/A	no	no

- *Engin Türetken is with the Computer Vision Laboratory, EPFL, Lausanne, and the Swiss Center for Electronics and Microtechnology (CSEM), Neuchâtel, Switzerland.*
  E-mail: engin.tueretken@alumni.epfl.ch
- *Przemysław Głowacki and Pascal Fua are with the Computer Vision Laboratory, EPFL, Lausanne, Switzerland.*
  E-mail: przemyslaw.glowacki@epfl.ch, pascal.fua@epfl.ch
- *Fethallah Benmansour is with Roche Pharma Research and Early Development, Roche Innovation Center, Basel, Switzerland.*
  E-mail: fethallah.benmansour@roche.com
- *Bjoern Andres is with the School of Engineering and Applied Sciences, Harvard University, Cambridge, US.*
  E-mail: bjoern@andres.sc
- *Hanspeter Pfister is with the Visual Computing Group, Harvard University, Cambridge, US.*
  E-mail: pfister@seas.harvard.edu

Tool	Type	Output	Tubularity	Platf.	Lang.	Dim.	Color	Radii	3D P.	3D V.	Code	Free
NeuriteTracer [19]	A	S: Thresholding and skeletonization	Pixel intensities after illum. and contrast enhancement	L/M/W	ImageJ Macro	2D	no	no	N/A	N/A	yes	yes
Neurolucida [20]	S/A	T	X	W	X	2D/3D	X	yes	yes	yes	no	no
Neuromantic [3]	M/S	T: Dijkstra shortest path in image space	Hessian eigenvalues and geometry prior	W	C++	2D/3D	no	yes	yes	yes	yes	yes
Neuron_Morpho [2]	M	T	N/A	L/M/W	Java	2D/3D	no	no	no	no	yes	yes
NeuronGrowth [6]	S	T: Greedy tracking based on Hessian eigenvector directions	Hessian max. eigenvalue	L/M/W	Java	2D+t	no	no	no	no	no	yes
NeuronJ [5]	S	T: Dijkstra shortest path in image space	Hessian eigenvalues and geometry prior	L/M/W	Java	2D	no	no	N/A	N/A	no	yes
NeuronStudio [12]	A	T: Thresholding, skeletonization and Rayburst sampling for radius estimation	Pixel intensity values after denoising and deconvolution	W	C	2D/3D	no	yes	yes	yes	yes	yes
NCTracer [13]	A	T: Voxel coding applied to binarized stacks	Center surround filter: Laplacian of Gaussian	W	Java and Matlab	3D	no	yes	yes	yes	no	yes
Reconstruct [1]	M/S	T: Region growing	A function of the region hue, saturation and brightness	W	C	2D/3D	yes	no	no	yes	yes	yes
Simple Neurite Tracer (SNT) [7]	S	T: Bidirectional A* search	Hessian functional	L/M/W	Java	3D	no	yes	yes	yes	yes	yes
TrakEM2 [21]	M	T (treeline object)	N/A	L/M/W	Java	2D/3D	no	no	no	yes	yes	yes
TREES Toolbox [14]	M/A	T: Thresholding and skeletonization	Pixel intensities	L/M/W	Matlab	2D/3D	no	yes	yes	yes	yes	yes
Vaa3D [9], [10], [15]	S, A	T: Dijkstra shortest path in image space	An exponential function of the inverse pixel intensity	L/M/W	C++	3D	no	yes	yes	yes	yes	yes

TABLE 1: Existing tools for reconstructing curvilinear structures. From left to right, the columns list tool name with a reference, algorithm type (M: manual, S: semi-automated, A: Automated), output type and short description of the algorithm (T: tracing=vector graphics, S: segmentation=binary or label image), tubularity measure, supported platforms (L: Linux, M: Apple Macintosh, W: Microsoft Windows), language of implementation, supported image dimensions (i.e., application domain), whether color information is taken into account in the processing, whether radius estimates are automatically produced, whether processing takes place directly in 3D or done slice by slice (for 3D images only), whether 3D visualization of the reconstructions is supported, whether source code is publicly available, and whether the tool is free. X denotes unknown and N/A not-applicable. The rightmost column shows whether the tools are publicly available as source code or binaries. For more detailed information, see the associated references and tool web-pages.

APPENDIX B: PATH CLASSIFICATION IMPLEMENTATION DETAILS

We have tested our approach using five different classifiers: SVM classifiers with linear and RBF kernels, random forest (RF) classifiers with oblique and orthogonal splits and a gradient boosted decision tree (GDBT) classifier. To train the classifiers, we randomly sampled 30000 positive and 30000 negative paths from the graphs. For assessing the ROC performance, we used 10000 positive and negative samples at detection time.

For image stacks of the *Brainbow* dataset, we extended the histograms of gradient deviation features introduced in Section 5.1 to include color information. This is done by first converting the stacks into the CIELAB space and clustering their voxels using the K-Means algorithm with $K = 50$ clusters. For each cluster, we then computed a normalized gray scale image whose voxel values are inversely proportional to the color distance between the original voxel and the cluster mean. This results in K gray scale images and each voxel is associated to the one its cluster corresponds to. For a voxel along any given path, the gradient features are computed on this associated image. As a result, only gradients corresponding to voxels with a similar color are taken into account.

Fig. 1(a) shows the classification performance achieved by the five classifiers on the testing set of the *Aerial* dataset. We used the GDBT classifier for all the experiments presented in the results section of the manuscript since it clearly outperforms the two RF ones, which both perform better than the SVMs. Fig. 1(b) shows the performance on all the five datasets using this classifier. The *Brightfield* and *Confocal-Vessels* datasets yield a lower true positive rate at a fixed false positive rate than the other three. In the case of the *Brightfield* dataset, this is because of the structure irregularities and significant point spread function blurring. In the *Confocal-Vessels* case, it is mainly due to non-uniformly stained blood vessels.

Fig. 1(c-f) provides an analysis of the effect of the feature parameter settings on classification accuracy. Briefly, increasing the number of radius intervals, histogram bins and codebook entries improves the accuracy at the expense of

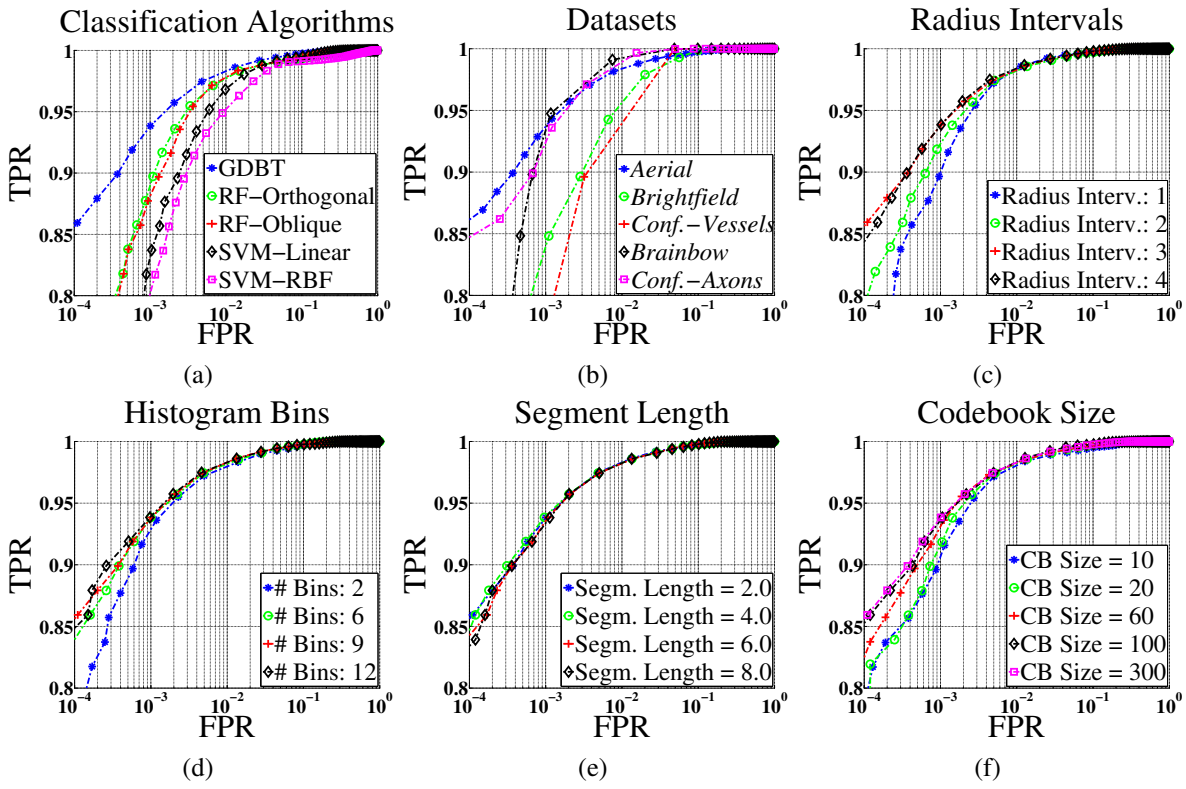


Fig. 1: (a) ROC curves for the five classifiers evaluated on the *Aerial* dataset. (b) Curves for all the five datasets using the GDBT classifier. (c-f) Influence of the feature extraction parameters on the classification accuracy for the *Aerial* dataset using the same classifier.

a higher computational cost. On the other hand, short segment lengths provide a fine grained sampling of the paths and hence slightly better performance.

APPENDIX C: LIKELIHOOD TERM DERIVATION FOR TREE STRUCTURES

We prove here that the first negative log likelihood of Eq. 3 can be written as a linear function of the t_{ijk} indicator variables as given in Eq. 5.

$$P(I, G|\mathbf{T} = \mathbf{t}) = \prod_{e_{ijk} \in F} P(I_{ijk}, e_{ijk}|T_{ijk} = t_{ijk}) \tag{C.1}$$

$$= \prod_{e_{ijk} \in F} \left[\frac{P(T_{ijk} = t_{ijk}|I_{ijk}, e_{ijk}) P(I_{ijk}, e_{ijk})}{P(T_{ijk} = t_{ijk})} \right] \tag{C.2}$$

$$\propto \prod_{e_{ijk} \in F} P(T_{ijk} = t_{ijk}|I_{ijk}, e_{ijk}) \tag{C.3}$$

$$\propto \prod_{e_{ijk} \in F} P(T_{ijk} = 1|I_{ijk}, e_{ijk})^{t_{ijk}} \quad \times \quad P(T_{ijk} = 0|I_{ijk}, e_{ijk})^{1-t_{ijk}} \tag{C.4}$$

$$\propto \prod_{e_{ijk} \in F} \left(\frac{P(T_{ijk} = 1|I_{ijk}, e_{ijk})}{P(T_{ijk} = 0|I_{ijk}, e_{ijk})} \right)^{t_{ijk}} \tag{C.5}$$

where I_{ijk} represents image data around the tubular path corresponding to e_{ijk} as defined in the manuscript. In Eq. C.1, we assume that image evidence of tubularness along edge pairs is conditionally independent if we actually know whether these pairs belong to the underlying curvilinear structures. We use Bayes' rule in Eqs. C.2 and drop the constant terms in Eq. C.3, assuming that edge pairs are *a priori* equally likely to belong to the structures. We obtain the following two equations by simple algebraic manipulations and dropping a constant term. Finally, we obtain the minimization of Eq. 4 by taking the negative logarithm of Eq. C.5

$$\underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmin}} \sum_{e_{ijk} \in F} -\log \left(\frac{P(T_{ijk} = 1|I_{ijk}, e_{ijk})}{P(T_{ijk} = 0|I_{ijk}, e_{ijk})} \right) t_{ijk} \quad = \quad \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmin}} \sum_{e_{ijk} \in F} c_{ijk} \quad t_{ijk} \quad . \tag{C.6}$$

where c_{ijk} is a cost term that accounts for the quality of the geodesic paths associated with the edge pair e_{ijk}.

APPENDIX D: GRAPH SIZE REDUCTION

We present here a pruning technique and four reduction procedures that we combine to reduce the size of the tubular graphs by removing low-probability edges and merging consecutive edge pairs.

For each edge in the graph, the pruning approach first involves finding the maximum probability edge pair containing the edge and assigning the pair's probability to it. We then traverse the edges in ascending order of their assigned probability and remove them from the graph until a certain probability value t_p is reached and the number of remaining edge pairs in the pruned graph is smaller than a specified size limit N_{ep}. This is done while preventing the edges with probabilities higher than t_p from being disconnected from the graph. As can be seen in Fig. 3(d) of the manuscript, many background edges are assigned very low probability values. Hence, the graph size can be tremendously reduced by removing them prior to the IP optimization. In all our experiments, we set $t_p = 0.2$ and $N_{ep} = 2000$.

The reduction procedures consist of a series of preprocessing steps that remove edges or merge edge pairs based on their assigned probability values. In the following, we describe four procedures in the order they are applied.

D.1. Root Edge Reduction

In our formalism, each curvilinear network in a given image has a single root point denoted by the vertex $r \in R \subset V$. We therefore require each connected component in the final solution to contain exactly a single vertex from R, which we designate as the root of that component.

The constraints introduced in Section 6.4.2 guarantee such a solution because incoming edges to the root vertices, that is, $\{e_{ir} \in E \mid \forall i \in V/v, r \in R\}$, can never be active. We therefore remove these edges from the graph, which can be done in linear time in the number of root vertices.

D.2. Accessibility Reduction

As described in Section 6.4.2 (Eq. 20) of the manuscript, oppositely directed edges e_{ij} and e_{ji} cannot be simultaneously active. Thus we remove edge e_{ji} from such a pair if all the directed paths from the virtual vertex v to vertex j contains its oppositely directed edge e_{ij}. We identify such an edge e_{ji} in linear time in the number of graph edges by first removing e_{ij} from the graph and then checking if j can be reached from v in the resulting graph.

D.3. Terminal Edge Reduction

We call an edge e_{ij} of the graph a *terminal edge* if the vertex j is adjacent only to the vertex i. We remove such a terminal edge from the graph if all its incoming edge pairs $\{e_{kij}\}$ have a non-negative cost value. That is, we remove a terminal edge e_{ij} if

$$c_{kij} \geq 0, \qquad \forall e_{kij} \in F : \ k \in V \tag{D.1}$$

because it can never be part of the optimal solution.

D.4. Edge Merger Reduction

We say a vertex $j \in V$ is a *continuation vertex* if it is not a root vertex (i.e., $j \notin R \cup \{v\}$) and it has exactly two adjacent vertices (i.e., $adj(j) = 2$). We merge all pairs of consecutive edge pairs centered at a continuation vertex if they either have all non-negative costs or all non-positive ones. In other words, we merge all the pairs of edges $\{e_{ij}, e_{jk}\}$ incident to a continuation vertex j if one of the following conditions are satisfied

$$c_{ijk} > 0, \qquad \forall e_{ijk} \in F : \ i \in V/R, \ k \in V/R \tag{D.2}$$

$$c_{ijk} < 0, \qquad \forall e_{ijk} \in F : \ i \in V/R, \ k \in V/R \tag{D.3}$$

We create a new edge e_{ik} for every merged edge pair e_{ijk}, and remove the center vertex j and the edges e_{ij}, e_{jk} from the graph. We assign the cost value c of a merged edge pair e_{ijk} to its newly created edge e_{ik}. The objective function term for this cost value then becomes $\sum_{m \in V \,:\, e_{mi} \in E} (c \ t_{mik})$.

Note that the first three reduction procedures introduced above do not compromise global optimality. That is, the global optimum of the reduced problem is equivalent to that of the original one. In contrast, the edge merger procedure we introduce here does not necessarily provide such an optimality guarantee. In practice, however, we did not observe connectivity issues in our solutions mainly because of the high accuracy of the path classifier (an average TPR of 97% at a FPR of 1%) as demonstrated by Fig. 10(b).

APPENDIX E: OPTIMAL PRUNING OF A MINIMUM SPANNING TREE

Let $G = (\{V = v_i\}, E = \{e_{ij}\})$ be the tubular graph of Section 6 with vertices V and directed edges E. If we assign to each edge e_{ij} the weight $-\log \frac{p_{ij}}{1-p_{ij}}$, a directed minimum spanning tree $\mathbf{t_{MST}} = (\{V = v_i\}, E_{MST} = \{e_{ij}\} \subset E)$ will span all the vertices while minimizing the energy of Eq. 32 [22]

$$\sum_{e_{ij} \in E} -\log\left(\frac{p_{ij}}{1-p_{ij}}\right) t_{ij} \quad . \tag{E.1}$$

In practice, some of the vertices correspond to false detections that are really part of the background. One way to eliminate them is to prune $\mathbf{t_{MST}}$, that is, remove some edges while preserving the tree topology. To guarantee that the results are as comparable as possible to the ones produced by the k-MST and IP algorithms, we find the optimal connected subset of edges $E_{PRU} \subset E_{MST}$ that represent the pruned tree:

$$\underset{E_{PRU} \subset E_{MST}}{\arg\min} \sum_{e_{ij} \in E_{MST}} -\log\left(\frac{p_{ij}}{1-p_{ij}}\right) t_{ij} \quad . \tag{E.2}$$

In other words, we find the subtree that minimizes the same energy functional of Eq 32. To this end, we exploit the fact that we can rewrite the cost of an optimal subtree rooted at v_i as a recursive function on the vertices as

$$w(v_i) = \sum_{\substack{v_j \in V\,: \\ e_{ij} \in E_{MST}}} \min(-\log\left(\frac{p_{ij}}{1-p_{ij}}\right) + w(v_j), 0) \quad , \tag{E.3}$$

where the cost at a terminal vertex with zero out degree is equal to zero. For every edge $e_{ij} \in E_{MST}$ outgoing from vertex v_i, the algorithm checks whether it should be kept or removed from the solution, In the first case, we add the edge-weight and the cost $w(v_j)$ of the optimal subtree rooted at v_j, In the second case, the contribution to the cost is zero. All the $w(v_i)$ values are computed recursively in this way starting from the root while labeling the outgoing edges of each vertex as retainable or not.

Once the edges are labelled, the final solution is taken as the largest connected set of retained edges reachable from the root. This process is illustrated by Fig. 2.

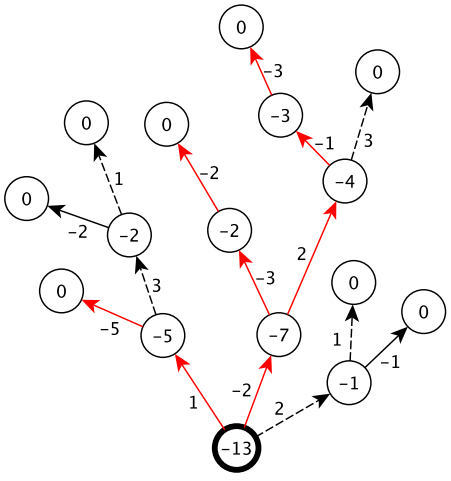


Fig. 2: A directed minimum spanning tree rooted at the bottom vertex. The numbers on the edges represent edge weights, that is, the negative log likelihood ratios, and the numbers on the vertices represent the optimal costs $w(v_i)$ of their subtrees. Edges marked as retainable are represented by solid arrows. The remaining edges appear as dashed arrows. The red edges constitute the optimal subtree. Note that not all retainable edges are in fact retained in the final solution. Also, not all of the pruned edges have positive weights and not all of the retained ones have negative ones. In this specific example, the weight of the initial minimum spanning tree is -6 and the weight of the optimal subtree is -13.

REFERENCES

[1] J. C. Fiala, "Reconstruct: A Free Editor for Serial Section Microscopy," *Journal of microscopy*, vol. 218, pp. 52–61, 2005.
[2] M. Brown, R. Szeliski, and S. Winder, "Multi-Image Matching Using Multi-Scale Oriented Patches," in *Conference on Computer Vision and Pattern Recognition*, 2005.
[3] D. R. Myatt, T. Hadlington, G. A. Ascoli, and S. J. Nasuto, "Neuromantic - from Semi Manual to Semi Automatic Reconstruction of Neuron Morphology," *Frontiers in Neuroinformatics*, vol. 6, no. 4, 2012.

[4] Bitplane, "Imaris: 3D and 4D Real-Time Interactive Image Visualization," 2013, http://www.bitplane.com/go/products/imaris/.
[5] E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser, "Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images," *Cytometry Part A*, vol. 58A, no. 2, pp. 167–176, April 2004.
[6] Z. Fanti, F. F. De-Miguel, and M. E. Martinez-Perez, "A Method for Semiautomatic Tracing and Morphological Measuring of Neurite Outgrowth from DIC Sequences," in *Engineering in Medicine and Biology Society*, 2008, pp. 1196–1199.
[7] M. Longair, D. A. Baker, and J. D. Armstrong, "Simple Neurite Tracer: Open Source Software for Reconstruction, Visualization and Analysis of Neuronal Processes," *Bioinformatics*, vol. 27, no. 17, pp. 2453–2454, 2011.
[8] R. Srinivasan, Q. Li, X. Zhou, J. Lu, J. Lichtman, and S. T. C. Wong, "Reconstruction of the Neuromuscular Junction Connectome," *Bioinformatics*, vol. 26, no. 12, pp. 64–70, 2010.
[9] H. Peng, Z. Ruan, D. Atasoy, and S. Sternson, "Automatic Reconstruction of 3D Neuron Structures Using a Graph-Augmented Deformable Model," *Bioinformatics*, vol. 26, no. 12, pp. 38–46, 2010.
[10] H. Peng, F. Long, and G. Myers, "Automatic 3D Neuron Tracing Using All-Path Pruning," *Bioinformatics*, vol. 27, no. 13, pp. 239–247, 2011.
[11] Y. Wang, A. Narayanaswamy, and B. Roysam, "Novel 4D Open-Curve Active Contour and Curve Completion Approach for Automated Tree Structure Extraction," in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1105–1112.
[12] S. Wearne, A. Rodriguez, D. Ehlenberger, A. Rocher, S. Henderson, and P. Hof, "New Techniques for Imaging, Digitization and Analysis of Three-Dimensional Neural Morphology on Multiple Scales," *Neuroscience*, vol. 136, no. 3, pp. 661–680, 2005.
[13] P. Chothani, V. Mehta, and A. Stepanyants, "Automated Tracing of Neurites from Light Microscopy Stacks of Images," *Neuroinformatics*, vol. 9, pp. 263–278, 2011.
[14] H. Cuntz, F. Forstner, A. Borst, and M. Häusser, "One Rule to Grow Them All: A General Theory of Neuronal Branching and Its Practical Application," *PLoS Computational Biology*, vol. 6, no. 8, p. 1000877, 2010.
[15] H. Xiao and H. Peng, "APP2: Automatic Tracing of 3D Neuron Morphology Based on Hierarchical Pruning of a Gray-Weighted Image Distance-Tree," *Bioinformatics*, vol. 29, no. 11, pp. 1448–1454, 2013.
[16] Y. Wang, A. Narayanaswamy, C. Tsai, and B. Roysam, "A Broadly Applicable 3D Neuron Tracing Method Based on Open-Curve Snake," *Neuroinformatics*, vol. 9, no. 2-3, pp. 193–217, 2011.
[17] E. Turetken, F. Benmansour, and P. Fua, "Semi-Automated Reconstruction of Curvilinear Structures in Noisy 2D Images and 3D Image Stacks," EPFL-182839, Tech. Rep., 2013.
[18] D. Wang, R. Lagerstrom, C. Sun, L. Bishof, P. Valotton, and M. Götte, "Hca-Vision: Automated Neurite Outgrowth Analysis," *Journal of Biomolecular Screening*, vol. 15, no. 9, pp. 1165–1170, 2010.
[19] M. Pool, J. Thiemann, A. Bar-Or, and A. E. Fournier, "Neuritetracer: A Novel Imagej Plugin for Automated Quantification of Neurite Outgrowth," *Journal of Neuroscience Methods*, vol. 168, no. 1, pp. 134–139, 2008.
[20] MBF Bioscience, "Neurolucida: Microscope Systems for Stereology and Neuron Morphology," 2013, http://www.mbfbioscience.com/neurolucida/.
[21] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. J. Douglas, "TrakEM2 Software for Neural Circuit Reconstruction," *PLoS One*, vol. 7, no. 6, p. 38011, 2012.
[22] Y. Chu and T. Liu, "On Shortest Arborescence of a Directed Graph," *Scientia Sinica*, vol. 14, no. 10, p. 1396, 1965.