Reconstructing Curvilinear Networks using Path Classifiers and Integer Programming

Engin Türetnen, 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 imaging systems can produce vast amounts of data featuring a 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
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.

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
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

$$p_{ij} = \arg\min_{\gamma} \int_{0}^{1} P(\gamma(s)) \, ds,$$

where $P$ is the negative exponential mapping of the tubularity measure, $s \in [0, 1]$ is the arc-length parameter and $\gamma$ 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
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 from the DIADEM data (2), 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 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 \( C(s) \) 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 \( \nabla I(x) \) and normal vectors \( \mathbf{N}(x) \) obtained from the points \( x \in \gamma_i(s) \) within the tubular segment. To ensure that all the gradient information surrounding the path is captured, we enlarge the segments by a margin \( m(s) = \beta \cdot r(s) \) 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 \( x \in \gamma_i(s) \), let \( C(s_x) \) be the closest centerline point, and \( \mathbf{N}(x) \) be the normal ray vector from \( C(s_x) \) to \( x \), 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:

\[
\Psi(x) = \begin{cases} 
\text{angle}(\nabla I(x), \mathbf{N}(x)), & \text{if } \| x - C(s_x) \| > \varepsilon \\
\text{angle}(\nabla I(x), \mathbf{II}(x)), & \text{otherwise},
\end{cases}
\]

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

We compute the radius interval and the angular bin indices for a point \( x \) respectively as \( \min(R - 1, [R\| \mathbf{N}(x) \|/(r(s_x) + m(s_x))] \) and
The curvature along the centerline curve of a common section, such as the one shown in Fig. 4(a), yields a feature vector of minimal distances that has the same length as the number of elements in the BoW model.

For a given path of arbitrary length, we then compute predefined number of descriptors from the training data. The resulting feature vector of distances has the same length as the number of elements in the BoW model. The words of the BoW model are generated during training by randomly sampling a compactly represent their feature space. The words of the BoW model are generated by randomly sampling a predefined number of descriptors from the training data. The above procedure produces an arbitrary number of HGD descriptors introduced in Section 5.1 are computed as

\[
\text{Hist}(x) = \text{round}(B \cdot \text{hist}(x) / \pi)
\]

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 \( C(s) \), 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

\[
\begin{align*}
\text{argmax} & \|\mathbf{T}'(s)\|, d/L, \Delta_z/L \text{ and } \Delta_r/L, \text{ where } \Delta_z \text{ and } \Delta_r \text{ are the path lengths along the image axial dimension } z \text{ and the scale dimension, and } \mathbf{T}(s) \text{ is the unit tangent vector depicted in Fig. 5.}
\end{align*}
\]

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 \( l_{g}(p_g) \) denote the length of \( p_g \)’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) \( \max(l_{g}(p_h), l_{g}(p_g)) \) is larger than a length threshold \( t_1 \), which is taken to be the minimum image spacing in our experiments.

2) The ratio \( \text{min}(l(p_h), l(p_g))/\max(l(p_h), l(p_g)) \) 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 \( E = \{e_{ij} | i \in V, j \in V\} \) 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.
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 $T_{ij}$ and $T_{ijk}$ be the corresponding hidden variables. Let also $\mathcal{T}$ and $\mathcal{T}^*$ 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 $t^*$ as the solution of

$$
t^* = \arg\max_{t \in \mathcal{T}} P(T = t | I, G),
$$

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

### 6.3 Tree Reconstruction

In this section, we take $\mathcal{P}$ 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 $\log(P(T = t))$ 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 $P(I, G | T = t)$ as a product of individual edge pair likelihoods. As shown in Appendix C, this leads to

$$
t^* = \arg\min_{t \in \mathcal{T}(G)} \sum_{e_{ijk} \in F} -\log \left( \frac{P(T_{ijk} = 1 | I_{ij}, e_{ijk})}{P(T_{ijk} = 0 | I_{ij}, e_{ijk})} \right) t_{ijk},
$$

where $I_{ijk}$ represents image data around the tubular path corresponding to $e_{ijk}$. The probability $P(T_{ijk} = 1 | I_{ij}, e_{ijk})$ 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 $t \in \mathcal{T}(G)$ amounts to solving a minimum arborescence problem [48] with a quadratic cost.
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 \( t \in T(G) \) 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

\[
\sum_{j \in V(r)} y_{ij}^l \leq 1, \quad \forall l \in V \setminus \{r\}, \quad (8)
\]

\[
\sum_{j \in V(l)} y_{ij}^l \leq 1, \quad \forall l \in V \setminus \{r\}, \quad (9)
\]

\[
\sum_{j \in V(l)} y_{ij}^l - \sum_{j \in V(l_i)} y_{ij}^l = 0, \quad \forall l \in V \setminus \{r, l_i\}, \quad (10)
\]

\[
y_{ij}^l \leq t_{ij}, \quad \forall e_{ij} \in E, \quad l \in V \setminus \{r, i, j\}, \quad (11)
\]

\[
y_{i*l} = t_{i*l}, \quad \forall e_{i*l} \in E, \quad (12)
\]

\[
y_{il} \geq 0, \quad \forall e_{il} \in E, \quad l \in V \setminus \{r, i\}, \quad (13)
\]

\[
t_{ij} \in \{0, 1\}, \quad \forall e_{ij} \in E, \quad (14)
\]

where the \( y_{ij}^l \) are auxiliary continuous variables that denote the flow from the root vertex to all others. More specifically, \( y_{ij}^l \) indicates whether the unique directed path from the root \( r \) to vertex \( l \) traverses the edge \( e_{ij} \). If the optimal tree \( t^* \) does not contain \( l \) and hence such a path does not exist, then \( y_{ij}^l = 0 \). The first two constraints ensure that there can be at most one path in \( t^* \) 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 \( t^* \) includes a path from the root to the vertex \( l \) passing through edge \( e_{il} \) if \( t^* \) 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

\[
t_{ov} = 1, \quad \forall e_{ov} \in E : \ r \in V_r, \quad (15)
\]

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

\[
-\log(P(I, G|T = t)) = \sum_{e_{ijk} \in E} (c_{ijk} + w_{ijk} t_{ijk}), \quad (16)
\]

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

To see this, let us consider two sets \( Y \) and \( Z \) of auxiliary random variables denoting respectively the presence of edges in the optimal solution and compatibility of consecutive edge pairs. More specifically, let \( Y = \{Y_{jk}\} \) be the vector of binary random variables indicating whether edges \( \{e_{jk}\} \) truly belong to the underlying curvilinear structures, and \( Y = \{y_{jk}\} \) 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 \( e_{ij} \) 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 \( e_{ij} \) and \( Z \) be the vector of all such variables. That is, \( Z_{jk} \) can take values from the set \( \{e_{ij} | e_{ij} \in E \setminus \{e_{jk}\}\} \). There is a one-to-one deterministic relation between \( T \) and \( (Y, Z) \), which we write as

\[
T_{ijk} = Y_{jk} 1(Z_{jk} = e_{ij}), \quad \forall e_{ij} \in E \quad (17)
\]

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

\[
P(I, G|T = t) = P(I, G|Y = y, Z=z) = \prod_{e_{ijk} \in E} P(I_{jk}, E_{jk}|Y_{jk} = y_{jk}, Z_{jk} = z_{jk}) \quad (18)
\]

\[
\times \prod_{e_{ijk} \in E} P(Z_{jk} = z_{jk}|Y_{jk} = y_{jk}, I_{jk}, E_{jk})P(Y_{jk} = y_{jk}|I_{jk}, E_{jk}) \quad (19)
\]

\[
\times \prod_{e_{ijk} \in E} \left[ P(Z_{jk} = z_{jk}|Y_{jk} = 1, I_{jk}, E_{jk})P(Y_{jk} = 0|I_{jk}, E_{jk})\right]^{1-y_{jk}} \quad (20)
\]

\[
\times \prod_{e_{ijk} \in E} \left[ P(Z_{jk} = z_{jk}|Y_{jk} = 0, I_{jk}, E_{jk})P(Y_{jk} = 1|I_{jk}, E_{jk})\right]^{y_{jk}} \quad (21)
\]

\[
\times \prod_{e_{ijk} \in E} \left[ \sum_{e_{ij} \in E} \left( \prod_{e_{ij} \in E} \left( \frac{p_{ijk}^{e_{ijk}}}{1-p_{ijk}^{e_{ijk}}} \right)^{y_{jk}} \right) \right] \quad (22)
\]

\[
\times \prod_{e_{ijk} \in E} \left[ \sum_{e_{ij} \in E} \left( \frac{p_{ijk}^{e_{ijk}}}{1-p_{ijk}^{e_{ijk}}} \right)^{y_{jk}} \right] \quad (23)
\]
where $E_{jk}$ denotes the set of edges containing $e_{ij}$ 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 is 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 $y_{jk}, i_{jk} \in \{0, 1\}$, 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 $p_{ijk}^t$ 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 $p_{ijk}^*$. Taking the negative logarithm of Eq. 23 yields

$$-\log(P(I,G|T = t)) = \sum_{i,j \in E} -\log \left( \frac{p_{ijk}^* p_{ijk}^* \deg^*(v_j, v_k)}{(1 - p_{ijk}^*)} \right) t_{ijk}$$

$$= \sum_{i,j \in E} \left( -\log \left( \frac{p_{ijk}^*}{1 - p_{ijk}^*} \right) - \log (p_{ijk}^* \deg^*(v_j, v_k)) \right) t_{ijk}$$

$$= \sum_{i,j \in E} \left( -\log \left( \frac{p_{ijk}^*}{1 - p_{ijk}^*} \right) + w_{ijk} \right) t_{ijk}$$

$$\approx \sum_{i,j \in E} \left( c_{ijk} + w_{ijk} \right) t_{ijk},$$

which is the formulation of Eq. 16. The probability $p_{ijk}^*$ denotes the likelihood that edge $e_{ij}$ 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 $e_{ij}$, we evaluate the path classifier on a larger neighbourhood including its in-edges $\{i_{ij}\}$ and use the corresponding probabilities in the above summation. Therefore, for each $p_{ijk}^*$ 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 $p_{ijk}^*$ 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

$$\sum_{e_{ijk} \in E} p_{ijk}^* = 1.$$ (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 $-\log(P(T = t))$, the second likelihood term of Eq. 4 to be a junction prior that penalizes unwarranted bifurcations or terminations, which we write as

$$\sum_{e_{ijk} \in E} \left( \sum_{c_{ijk} \in E} \lambda t_{mij} + \sum_{\gamma_{t} t_{ijn} + \sum_{\gamma_{t} t_{ijn}} \delta t_{t_{ijn} t_{ijk}}} \right),$$

where $\lambda, \gamma_{t}$, and $\delta_{t}$ are constants we derive below.

To this end, we treat the graph $G$ as a Bayesian network with latent variables $M_{ij} = \sum_{e_{ij} \in E} T_{mij}$ and $O_{ij} = \sum_{e_{ijk} \in E} T_{ij}$, 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

$$P(T = t) = \prod_{e_{ijk} \in E} P(T_{ijk} = t_{ijk}|O_{ij} = o_{ij}) P(O_{ij} = o_{ij}|M_{ij} = m_{ij}),$$

$$\propto \prod_{e_{ijk} \in E} P(O_{ij} = o_{ij}|M_{ij} = m_{ij}),$$

$$\propto \prod_{e_{ijk} \in E} P(O_{ij} = o_{ij}|M_{ij} = 1)^{m_{ij}} P(O_{ij} = o_{ij}|M_{ij} = 0)^{(1-m_{ij})},$$

$$\propto \prod_{e_{ijk} \in E} P(O_{ij} = o_{ij}|M_{ij} = 1)^{m_{ij}}$$

$$\propto \prod_{e_{ijk} \in E} \left( P(O_{ij} = 0|M_{ij} = 1)^{1(o_{ij} = 0)} \times P(O_{ij} = 1|M_{ij} = 1)^{1(o_{ij} = 1)} \right)^{m_{ij}}$$

$$P(O_{ij} = 1|M_{ij} = 1)^{1(o_{ij} = 1)} P(O_{ij} = 2|M_{ij} = 1)^{1(o_{ij} = 2)},$$

where $T_{ijk}$ denotes the vector of random variables $T_{t_{ijn}}, \forall e_{ijk} \in E$. In this work, we assume that all configurations $T_{ijk}$ are equally likely for an edge $e_{ijk}$, 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 \( P(O_{ij} = 0 | M_{ij} = 0)^{(1-m_{ij})} \) 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

\[
1(o_{ij} = 2) = \sum_{e_{ijn} \in E} \sum_{e_{ijk} \in E} t_{ijn} t_{ij},
\]

(32)

\[
1(o_{ij} = 1) = \sum_{e_{ijn} \in E} t_{ijn} - 2 \sum_{e_{ijn} \in E} \sum_{e_{ijk} \in E} t_{ijn} t_{ij},
\]

(33)

\[
1(o_{ij} = 0) = \sum_{e_{m} \in E} t_{m} - \sum_{e_{ijn} \in E} t_{ijn} + \sum_{e_{ijn} \in E} \sum_{e_{ijk} \in E} t_{ijn} t_{ij}.
\]

(34)

Note that multiplying these functions by \( m_{ij} = \sum_{e_{m} \in E} t_{m} \) 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

\[
-\sum_{e_{ij} \in E} \left[ \sum_{e_{ijn} \in E} \log(p^{i}) \cdot t_{ijn} + \sum_{e_{ijn} \in E} \sum_{e_{ijk} \in E} \log \left( \frac{p^{j}}{p^{i}} \right) t_{ijn} t_{ij} \right],
\]

(35)

where \( p^{i} = P(O_{ij} = 0 | M_{ij} = 1) \), \( p^{c} = P(O_{ij} = 1 | M_{ij} = 1) \) 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

\[
-\log(P(T = t)) = -\sum_{e_{ij} \in E} \left[ \sum_{e_{ijn} \in E} \log(p^{i}) \cdot t_{ijn} + \sum_{e_{ijn} \in E} \sum_{e_{ijk} \in E} \log \left( \frac{1}{p^{i}} \right) t_{ijn} t_{ij} \right],
\]

(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

\[
t^{*} = \arg \min_{t \in P} \sum_{e_{ij} \in F} a_{ij} t_{ij} + \sum_{e_{ij}, e_{ijn} \in F} b_{ij} t_{ijn} t_{ij}.
\]

(37)

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

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

We take the feasible region \( P_{c} \) 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

\[
\sum_{v, r, v \in F : e_{ij} \in v} t_{vr} \geq 1, \forall v \in V_{r}, \quad (38)
\]

\[
\sum_{v, r, v \in F : e_{ij} \in v} t_{ijr} + \sum_{v, r, v \in F : e_{ij} \in v} t_{ij} = 0, \forall v \in V_{r}. \quad (39)
\]

**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 to \( e_{ij} \) 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

\[
\sum_{e_{ij} \in E : m_{ij} \neq 0} t_{mij} + \sum_{e_{ij} \in E : m_{ij} \neq 0} t_{mji} \leq 1, \forall e_{ij} \in E : i \neq v, \quad (40)
\]

\[
\sum_{e_{ij} \in E : m_{ij} \neq 0} t_{mij} + \sum_{e_{ij} \in E : m_{ij} \neq 0} t_{mji} \leq 1, \quad (41)
\]

\[
\forall e_{mn}, e_{ij} \in F : \left\{ \begin{array}{l}
(e_{mn} = e_{ij}) \lor (e_{mn} = e_{ik}) \lor (e_{mn} = e_{ij}) \\
(e_{ij} \neq e_{ik} \land e_{mn} \neq e_{ij}) \lor (e_{mn} \neq e_{ik} \land e_{mn} \neq e_{ij}) \lor \left( e_{mn} \neq e_{ik} \land e_{mn} \neq e_{ij} \right) \\
I(p_{mn} < l(p_{ij})) \land I(p_{mn} \neq l(p_{ij}))
\end{array} \right\},
\]

where \( I(.) \) and \( l(.) \) denote the length and mean radius of a path respectively. \( \alpha \) 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 \( e_{ij} \), \( e_{kl1} \) and \( e_{kl2} \), 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

\[ \sum_{e_{kij} \in E} t_{kij} + \sum_{e_{lmq} \in E} t_{lmq} - \sum_{e_{iju} \in E} t_{iju} \leq 1, \] (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 arborizations. 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 \( e_{ij} \) is said to be connected if there exists a path in \( G \), starting at \( v \) and containing \( e_{ij} \), along which all the edge pairs are active.

Let \( y_{ij}^l (i \neq l) \) 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

\[ \sum_{e_{ij} \in E} y_{ij} = \sum_{e_{il} \in E} y_{il}, \forall l \in V \setminus \{v\}, \] (43)

\[ \sum_{e_{ij} \in E} y_{ij} \leq \text{deg}^{-}(l), \forall l \in V \setminus \{v\}, \] (44)

\[ \sum_{e_{ij} \in E} y_{ij} = \sum_{e_{jk} \in E} y_{jk}, \forall j, l \in V \setminus \{v\} : j \neq l, \] (45)

\[ y_{il} \geq t_{il}, \forall e_{il} \in F, \] (46)

\[ y_{il} = \sum_{e_{il} \in E} t_{il}, \forall e_{il} \in E : i \neq v, \] (47)

\[ y_{ij} \leq \text{deg}^{-}(l) \sum_{e_{kij} \in E} t_{kij}, \forall e_{kij} \in E : i \neq v, \forall v \in V \setminus \{v \}, \] (48)

\[ y_{ij} \geq y_{jk} + \text{deg}^{-}(l)(t_{ij} - 1), \forall e_{ijk} \in F, \forall v \in V \setminus \{v \}, j \neq l, \] (49)

where \( \text{deg}^{-}(\cdot) \) 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 \( 2e^{-10} \) 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 pairs \( t_{ij} \), 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 \( y_{ij} \) 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 in [48] to edge pairs by imposing

\[ \sum_{e_{mn} \in H} t_{mn} \geq \sum_{e_{kij} \in E} \forall e_{kij} \in E \setminus (H \cup \epsilon_{cij}) \] (50)

where \( \epsilon_{cij} \) and forming a new set of edges \( H = \{ e_{kl} \mid e_{kl} \in E_c \setminus e_{ij} \} \). 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 multicore 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 following simplified versions of Eq. 3.Implemented two additional algorithms that minimize the novel components of our approach – the path classifier of Section 5 and the IP formalism of Section 6 – 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 $\sigma$, which is set to $3\Delta t$ 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.

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

$$\sum_{e_{ijk} \in E} -\log \frac{p_{ijk}}{1 - p_{ijk}} t_{ijk} t_{jkl},$$

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.
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 Brain-bow 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
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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BRF1</td>
<td>0.91</td>
<td>0.89</td>
<td>0.87</td>
<td>0.58</td>
<td>0.10</td>
</tr>
<tr>
<td>BRF2</td>
<td>0.91</td>
<td>0.90</td>
<td>0.90</td>
<td>0.65</td>
<td>0.80</td>
</tr>
<tr>
<td>BRF3</td>
<td>0.94</td>
<td>0.93</td>
<td>0.91</td>
<td>0.42</td>
<td>0.68</td>
</tr>
<tr>
<td>BRF5</td>
<td>0.90</td>
<td>0.90</td>
<td>0.74</td>
<td>0.58</td>
<td>0.69</td>
</tr>
</tbody>
</table>

TABLE 1

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

<table>
<thead>
<tr>
<th>BRBW1</th>
<th>BRBW2</th>
<th>BRBW3</th>
<th>BRFI</th>
<th>BRF2</th>
<th>BRF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loopy-IP</td>
<td>0.50</td>
<td>0.56</td>
<td>0.66</td>
<td>0.76</td>
<td>0.50</td>
</tr>
<tr>
<td>Arbor-IP</td>
<td>0.44</td>
<td>0.52</td>
<td>0.45</td>
<td>0.59</td>
<td>0.32</td>
</tr>
<tr>
<td>kMST [8]</td>
<td>0.26</td>
<td>0.16</td>
<td>0.25</td>
<td>0.51</td>
<td>0.28</td>
</tr>
<tr>
<td>kMST+PC</td>
<td>0.51</td>
<td>0.21</td>
<td>0.30</td>
<td>0.49</td>
<td>0.36</td>
</tr>
<tr>
<td>MIST+PC</td>
<td>0.46</td>
<td>0.40</td>
<td>0.49</td>
<td>0.44</td>
<td>0.37</td>
</tr>
</tbody>
</table>

TABLE 2

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 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 MIST+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 highways 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.

7.4 Run Time Analysis

Table 4 provides running times in minutes per image for our 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 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. CONV denotes an image stack from the Confocal-Vessels dataset.

<table>
<thead>
<tr>
<th>Network Flow</th>
<th>BRBW1</th>
<th>BRBW2</th>
<th>BRBW3</th>
<th>BRFI</th>
<th>BRF2</th>
<th>BRF3</th>
<th>CONV1</th>
<th>CONV2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset + Lazy</td>
<td>32.9</td>
<td>4.9</td>
<td>19.5</td>
<td>36.6</td>
<td>10.9</td>
<td>7.7</td>
<td>6.4</td>
<td>4.2</td>
</tr>
<tr>
<td># Lazy Constr</td>
<td>1008</td>
<td>892</td>
<td>1092</td>
<td>1556</td>
<td>1165</td>
<td>2259</td>
<td>972</td>
<td>5175</td>
</tr>
</tbody>
</table>
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), in part by the Swiss Center for Electronics and Microtechnology (CSEM), and in part by the EU ERC MicroNano Project. 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


ACKNOWLEDGEMENTS

This work was supported in part by a grant from the Swiss National Science Foundation (SNSF), in part by the Swiss Center for Electronics and Microtechnology (CSEM), and in part by the EU ERC MicroNano Project. 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


Engin Türeken 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.

Bjorn 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 Glowacki 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 Ang Weng 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.