Automated Delineation of Dendritic Networks in Noisy Image Stacks

Germán González[1], François Fleuret[2,⋆], and Pascal Fua[1]

[1] Ecole Polytechnique Fédérale de Lausanne, Computer Vision Laboratory,
Bâtiment BC, CH-1015 Lausanne, Switzerland
{german.gonzalez,pascal.fua}@epfl.ch
[2] IDIAP Research Institute, P.O. Box 592, CH-1920, Martigny, Switzerland
francois.fleuret@idiap.ch

Abstract. We present a novel approach to 3D delineation of dendritic networks in noisy image stacks. We achieve a level of automation beyond that of state-of-the-art systems, which model dendrites as continuous tubular structures and postulate simple appearance models. Instead, we learn models from the data itself, which make them better suited to handle noise and deviations from expected appearance.

From very little expert-labeled ground truth, we train both a classifier to recognize individual dendrite voxels and a density model to classify segments connecting pairs of points as dendrite-like or not. Given these models, we can then trace the dendritic trees of neurons automatically by enforcing the tree structure of the resulting graph. We will show that our approach performs better than traditional techniques on brighfield image stacks.

1 Introduction

Full reconstruction of neuron morphology is essential for the analysis and understanding of their functioning. In its most basic form, the problem involves processing stacks of images produced by a microscope, each one showing a slice of the same piece of tissue at a different depth.

Currently available commercial products such as Neurolucida[1], Imaris[2], or Metamorph[3] provide sophisticated interfaces to reconstruct dendritic trees and rely heavily on manual operations for initialization and re-initialization of the delineation procedures. As a result, tracing dendritic trees in noisy images remains a tedious process. It can take an expert up to 10 hours for each one. This limits the amount of data that can be processed and represents a significant bottleneck in neuroscience research on neuron morphology.

Automated techniques have been proposed but are designed to work on very high quality images in which the dendrites can be modeled as tubular structures [1,2]. In

⋆ Supported by the Swiss National Science Foundation under the National Centre of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

[1] http://www.microbrightfield.com/prod-nl.htm

[2] http://www.bitplane.com/go/products/imaris

[3] http://www.moleculardevices.com/pages/software/metamorph.html

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part IV, LNCS 5305, pp. 214–227, 2008.

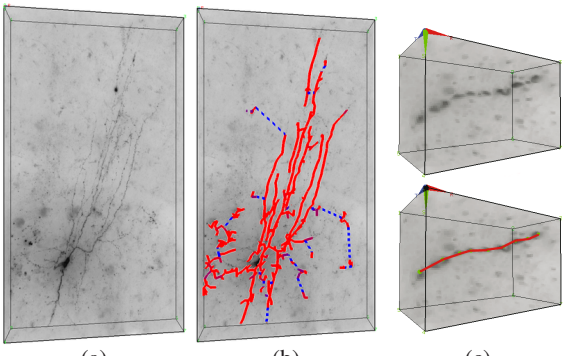


Fig. 1. (a) Minimum intensity projection of an image stack. Each pixel value is the minimum intensity value of the voxels that are touched by the ray cast from the camera through the pixel. (b) 3D tree reconstructed by our algorithm, which is best viewed in color. (c) Detail of the data volume showing the non-tubular aspect of a dendrite with the corresponding automatically generated delineation.

practice, however, due to the underlying neuron structure, irregularities in the dyeing process, and other sources of noise, the filaments often appear as an irregular series of blobs surrounded by other non-neuron structures, as is the case of the brightfield image stacks depicted by Fig. 1. Yet, such images are particularly useful for analyzing large samples. More generally, very high resolution images take a long time to acquire and require extremely expensive equipment, such as confocal microscopes. The ability to automatically handle lower resolution and noisier ones is therefore required to make these techniques more accessible. Ideally, the painstaking and data-specific tuning that many existing methods require should also be eliminated.

In this paper, we therefore propose an approach to handling the difficulties that are inherent to this imaging process. We do not assume an *a priori* dendrite model but rely instead on supervised and unsupervised statistical learning techniques to construct models as we go, which is more robust to unpredictable appearance changes. More specifically, we first train a classifier that can distinguish dendrite voxels from others using a very limited amount of expert-labeled ground truth. At run-time, it lets us detect such voxels, some of which should be connected by edges to represent the dendritic tree. To this end, we first find the minimum spanning tree connecting dendrite-like voxels. We then use an Expectation-Maximization approach to learn an appearance model for the edges that correspond to dendrites and those that do not. Finally, given these appearance models, we re-build and prune the tree to obtain the final delineation, such as the one depicted by Fig. 1(b), which is beyond what state-of-the-art techniques can produce automatically.

To demonstrate the versatility of our approach, we also ran our algorithm on retinal images, which we were able to do by simply training our classifier to recognize 2D blood vessel pixels instead of 3D dendrite voxels.

2 Related Work

Reconstructing networks of 3D filaments, be they blood vessels or dendrites, is an important topic in Biomedical Imaging and Computer Vision [3,4]. This typically involves measuring how filament-like voxels are and an algorithm connecting those that appear to be. We briefly review these two aspects below.

2.1 Finding Dendrite-Like Voxels

Most automated methods assume the filaments to be locally tubular and model them as generalized cylinders. The most popular approach to detecting such cylindrical structures in image stacks involves computing the Hessian matrix at individual voxels by convolution with Gaussian derivatives and relying on the eigenvalues of the Hessian to classify voxels as filament-like or not [5,6,7]. The Hessians can be modified to create an oriented filter in the direction of minimum variance, which should correspond to the direction of any existing filament [8,9]. To find filaments of various widths, these methods perform the computation using a range of variances for the Gaussian masks and select the most discriminant one. The fact that intensity changes inside and outside the filaments has also been explicitly exploited by locally convolving the image with differential kernels [1], finding parallel edges [10], and fitting *superellipsoids* or cylinders to the vessel based on its surface integral [2,11].

All these methods, however, assume image regularities that are present in high-quality images but not necessarily in noisier ones. Furthermore, they often require careful parameter tuning, which may change from one data-set to the next. As a result, probabilistic approaches able to learn whether a voxel belongs to a filament or not have begun to be employed. Instead of assuming the filaments to be cylinders, they aim at learning their appearance from the data. In [12], the eigenvalues of the structure tensor, are represented by a mixture model whose parameters are estimated via E-M. Support Vector Machines that operates on the Hessian's eigenvalues have also been used to discriminate between filament and non-filament voxels [13].

The latter approach [13] is closest to our dendrite detection algorithm. We however go several steps further to increase robustness: First, we drop the Hessian and train our classifier directly on the intensity data, thereby making fewer assumptions and being able to handle structures that are less visibly tubular. Second, we also learn an appearance model for the filament itself as opposed to individual voxels.

2.2 Reconstructing Filaments

Existing approaches to building the dendritic tree all rely on a *dendritness* measure of how dendrite-like filaments look, usually based on the voxel-based measures discussed above. They belong to one of two main classes.

The first class involves growing filaments from seed points [2,14,15,16]. This has been successfully demonstrated for confocal fluorescent microscopy images. It is computationally effective because the dendritness of filaments need only be evaluated in a small subset of the voxels. However, it may easily fail in noisy data because of its sequential nature. If the growing process diverges at one voxel, the rest of the dendritic tree will be lost.

The second class requires optimizing the path between seed points, often provided by the operator, to maximize the overall dendritness [8,11,17]. In these examples, the authors use active contour models, geometrical constraints and the live-wire algorithm between to connect the seeds.

By contrast to these methods that postulate an *a priori* cost function for connecting voxels, our approach learns a model at run-time, which lets it deal with the potentially changing appearance of the filaments depending on experimental conditions. Furthermore, we do this fully automatically, which is not the case for any of the methods discussed above.

3 Methodology

Our goal is to devise an algorithm that is fully automatic and can adapt to noisy data in which the appearance of the dendrites is not entirely predictable. Ideally we would like to find the tree maximizing the probability of the image under a consistent generative model. Because such an optimization is intractable, we propose an approximation that involves the three following steps:

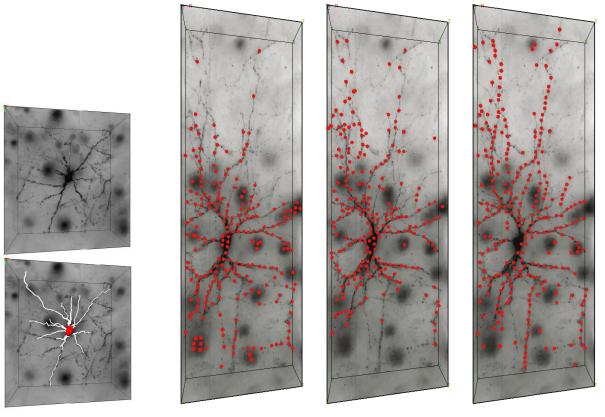
1. We use a hand-labeled training image stack to train once and for all a classifier that computes a voxel's probability to belong to a dendrite from its neighbors intensities.
2. We run this classifier on our stacks of test images, use a very permissive threshold to select potential dendrite voxels, apply non-maximum suppression, and connect all the surviving voxels with a minimum spanning tree. Some of its edges will correspond to actual dendritic filaments and other will be spurious. We use both the correct and spurious edges to learn filament appearance models in an EM framework.
3. Under a Markovian assumption, we combine these edge appearance models to jointly model the image appearance and the true presence of filaments. We then optimize the probability of the latter given the former and prune spurious branches.

As far as detecting dendrite voxels is concerned, our approach is related to the Hessian-based approach of [13]. However, dropping the Hessian and training our classifier directly on the intensity data lets us relax the cylindrical assumption and allows us to handle structures that are less visibly tubular. As shown in Fig. 2, this yields a marked improvement over competing approaches.

In terms of linking, our approach can be compared to those that attempt to find optimal paths between seeds [11,8] using a dendrite appearance model, but with two major improvements: First our seed points are detected automatically instead of being manually supplied, which means that some of them may be spurious and that the connectivity has to be inferred from the data. Second we do not assume an *a priori* filament model but learn one from the data as we go. This is much more robust to unpredictable appearance changes. Furthermore, unlike techniques that model filaments as tubular structures [1,2], we do not have to postulate regularities that may not be present in our images.

3.1 Notations

Given the three step algorithm outlined above, we now introduce the notations we will use to describe it in more details.



(a) Training volume (b) Grayscale (c) Hessian (d) Boosting

Fig. 2. (a) Training data. On top: image stack representing one neuron. Bellow: Manually delineated filaments overlaid in white. (b,c,d) Voxels labeled as potentially belonging to a dendrite. (b) By thresholding the grayscale images. (c) By using the Hessian. (d) By using our classifier. Note that the seed points obtained with our method describe better the underlying neuron structure.

Let $Z_1, \ldots, Z_N$ be the voxels corresponding to the local maxima of the classifier response and will serve as vertices for the dendritic tree we will build. For $1 \leq n \leq N$, let X_n be a Boolean random variable standing for whether or not there truly is a filament at location Z_n. Finally, Let $\boldsymbol{x} = (x_1, \ldots, x_N)$ and $\boldsymbol{x}_{\setminus i} = (x_1, \ldots, x_{i-1}, x_{i+1}, x_N)$.

For $1 \leq i \leq N$ and $1 \leq j \leq N$, let $J_{i,j}$ denote a random variable standing for the appearance of the edge going from Z_i to Z_j and let $L_{i,j} = ||Z_i - Z_j||$ be its length. $J_{i,j}$ is obtained by sampling the voxel response of the classifier in a regular lattice between (Z_i, Z_j). Let $A_{i,j}$ be a vector composed by the projection of $J_{i,j}$ in a latent space and $L_{i,j}$.

Let T denote the true dendritic tree we are trying to infer. It is a graph whose vertices are a subset of $Z_1, \ldots, Z_N$ and whose edges are defined by $\mathcal{G}$, a set of pairs of indexes in $\{1, \ldots, N\} \times \{1, \ldots, N\}$.

3.2 Local Dendrite Model

As discussed in Section 2, the standard approach to deciding whether voxels are inside a dendrite or not is to compute the Hessian of the intensities and look at its eigenvalues. This however implicitly makes strong assumptions on the expected intensity patterns. Instead of using such a hand-designed model, we train a classifier from a small quantity of hand-labeled neuron data with AdaBoost [18], which yields superior classification performance as shown in Fig. 2.

More specifically, the resulting classifier f is a linear combination of *weak learners* h_i:

$$f(x, y, z) = \sum_{i=1}^{N} \alpha_i h_i(x, y, z) \ , \tag{1}$$

where the h_i represent differences of the integrals of the image intensity over two cubes in the vicinity of (x, y, z) and T_i is the weak classifier threshold. We write

$$h_i(x, y, z) = \sigma\left(\sum_{V_i^1} I(x', y', z') - \sum_{V_i^2} I(x'', y'', z'') - T_i\right) \quad (2)$$

where σ is the sign function, V_i^1, V_i^2 are respectively the two volumes defining h_i, translated according to (x, y, z). These weak classifiers can be calculated with just sixteen memory accesses by using precomputed integral cubes, which are natural extensions of integral images.

During training, we build at each iteration 10^3 h_i weak learners by randomly picking volume pairs and finding an optimal T_i threshold for each. After running Adaboost, $N = 1000$ weak learners are retained in the f classifier of 1. The training samples are taken from the manual reconstruction of Fig. 2. They consist of filaments at different orientations and of a certain width. The final classifier responds to filaments of the predefined width, independently of the orientation.

At run time, we apply f on the whole data volume and perform non-maximum suppression by retaining only voxels that maximize it within a $8 \times 8 \times 20$ neighborhood, such as those shown in Fig. 2. The anisotropy on the neighborhood is due to the low resolution of the images in the z axis, produced by the point spread function of the microscope.

3.3 Learning an Edge Appearance Model

The process described above yields $Z_1, \ldots, Z_N$, a set of voxels likely, but not guaranteed to belong to dendrites. To build an edge appearance model, we compute their minimum spanning tree. Some of its edges will correspond to filaments and some not. We therefore create a low dimensional descriptor for the edges, and use it to learn a gaussian mixture model that we can use to distinguish the two classes of edges.

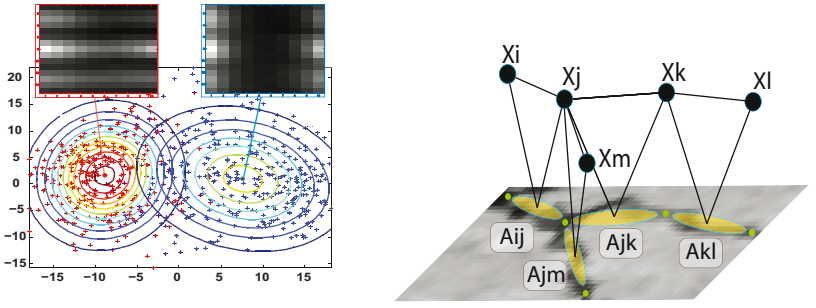
To obtain an edge descriptor, we first sample the voxel response on a regular lattice centered around each edge and perform PCA on the resulting set of vectors. For each edge, we retain the first N PCA components. We construct a $N + 1$-D edge feature vector, $A_{i,j}$ by appending the edge length $L_{i,j}$ to this N-D vector.

This population of $N + 1$-D vectors is a mixture of edges truly located on filaments, and of edges located elsewhere. We therefore apply an E-M procedure to derive both a prior and a Gaussian model for both. The only specificity of this unsupervised training is to force the covariance between the length and the other N components to be zero, since the length of an edge is only weakly correlated with its length-normalized appearance.

Hence, given a subgraph $\mathcal{G}$ with a population of edges that are both in the dendrite and elsewhere, this E-M procedure produces two Gaussian models μ_0 and μ_1 on R^{N+1} that represent respectively the edges truly on filaments and those elsewhere.

3.4 Building and Pruning the Tree

We can now use the edge appearance model to reconstruct the dendritic tree. To this end we first compute the maximum spanning tree using as weight for the edges their

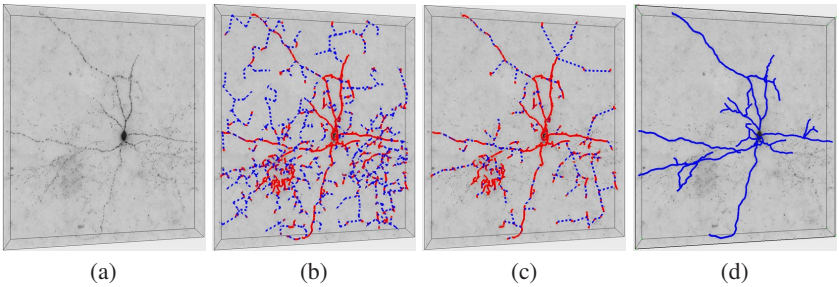


(a) (b)

Fig. 3. (a) First two dimensions of the PCA space of the edge appearance models. The Gaussian models are shown as contour lines. The two small figures at the top represent the projection of the means in the original lattice. The top-left one represents the model μ_1 for filaments, which appear as a continuous structure. The top-right one represents the non-filament model μ_0. Since, by construction the endpoints of the edges are local maxima, the intensity there is higher than elsewhere. (b) Hidden Markov Model used to estimate the probability of a vertex to belong to the dendritic tree.

likelihood to be part of a dendrite. Nevertheless, the tree obtained with this procedure is over-complete, spanning vertices that are not part of the dendrites, Fig. 4(b). In order to eliminate the spurious branches, we use the tree to evaluate the probability that individual vertices belong to a dendrite, removing those with low probability. We iterate between the tree reconstruction and vertex elimination until convergence, Fig. 4(c).

We assume that the relationship between the hidden state of the vertices and the edge appearance vectors can be represented in terms of a hidden Markov model such



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

Fig. 4. Building and pruning the tree. (a) Image stack (b) Initial maximum spanning tree. (c) After convergence of the iterative process. (d) Manually delineated ground truth. Red solid lines denote edges that are likely to be dendrites due to their appearance. Blue dashed lines represent edges retained by the minimum spanning tree algorithm to guarantee connectivity.The main filaments are correctly recovered. Note that our filament detector is sensitive to filaments thinner than the ones in the ground truth data. This produces the structures in the right part of the images that are not part of the ground truth data.

as the one depicted by Fig. 3(b). More precisely, we take $\mathcal{N}(\mathcal{G}, i)$ to be the neighboring vertices of i in $\mathcal{G}$ and assume that

$$P(X_i \,|\, \boldsymbol{X}_{\setminus i}, (A_{k,l})_{(k,l)\in\mathcal{G}}) = P(X_i \,|\, (X_k)_{k\in\mathcal{N}(\mathcal{G},i)}, (A_{i,k})_{k\in\mathcal{N}(\mathcal{G},i)}) \;, \tag{3}$$

$$P(A_{i,j} \,|\, \boldsymbol{X}, (A_{k,l})_{(k,l)\in\mathcal{G}\setminus(i,j)}) = P(A_{i,j} \,|\, X_i, X_j) \quad . \tag{4}$$

Under these assumptions, we are looking for a tree consistent with the edge appearance model of section 3.3. This means that the labels of its vector of maximum posterior probabilities $\boldsymbol{x}$ are all 1s. To do so we alternate the building of a tree spanning the vertices currently labeled 1 and the re-labeling of the vertices to maximize the posterior probability. The tree we are looking for is a fixed point of this procedure.

Building the Tree. We are looking for maximum likelihood tree that spans all vertices. Formally:

$$\begin{aligned}
&\operatorname*{argmax}_{\mathcal{T}} \{\log P(T = \mathcal{T} \,|\, (A_{i,j})_{1\le i,j\le N})\} \\
&= \operatorname*{argmax}_{\mathcal{T}} \{\log P((A_{i,j})_{1\le i,j\le N} \,|\, T = \mathcal{T})\} = \operatorname*{argmax}_{\mathcal{T}} \sum_{i,j\in\mathcal{T}} \log \frac{\mu_1(A_{i,j})}{\mu_0(A_{i,j})} \quad .
\end{aligned}$$

To this end, we use a slightly modified version of the minimum spanning tree algorithm. Starting with an empty graph, we add to it at every iteration the edge (i, j) that does not create a cycle and maximizes

$$\log(\mu_1(A_{i,j})/\mu_0(A_{i,j})) \quad .$$

While this procedure is not guaranteed to find a global optimum, it gives good results in practice. The main weakness we have to deal with is the over-completeness of the resulting tree. While it is very rare to miss an important vertex or part of filament, we have to discard many spurious branches spanned on non-filaments.

Eliminating Unlikely Vertices. From the appearance models μ_0 and μ_1 learned in section 3.3, and the Markovian assumption of Section 3.3, we can estimate for any graph $\mathcal{G}$ the most probable subset of nodes truly on filaments. More specifically, we are looking for the labeling $\boldsymbol{x}$ of maximum posterior probability given the appearance, defined as follow

$$\operatorname*{argmax}_{\boldsymbol{x}} P(\boldsymbol{X} = \boldsymbol{x} \,|\, (A_{i,j})_{i,j\in\mathcal{G}})$$

Since full optimization is intractable we propose an iterative greedy search. We loop through each point i, flipping the value of x_i if it increases the posterior probability. This can be seen as a component-wise optimization where the updating rule consists of fixing all $x_j, j \neq i$ and applying the following update to x_i

$$\begin{aligned}
x_i &\leftarrow \operatorname*{argmax}_{x} P(X_i = x, \boldsymbol{X}_{\setminus i} = \boldsymbol{x}_{\setminus i} \,|\, (A_{i,j})_{i,j\in\mathcal{G}}) \\
&= \operatorname*{argmax}_{x} P(X_i = x \,|\, \boldsymbol{X}_{\setminus i} = \boldsymbol{x}_{\setminus i},\, (A_{i,j})_{i,j\in\mathcal{G}}),
\end{aligned}$$

and under assumptions (3) and (4), we have

$$P(X_i = x \mid \boldsymbol{X}_{\setminus i} = \boldsymbol{x}_{\setminus i}, (A_{i,j})_{i,j \in \mathcal{G}})$$
$$= \prod_{j \in \mathcal{N}(\mathcal{G},i)} P(X_j = x_j \mid X_i = x) P(A_{i,j} \mid X_i = x, X_j = x_j),$$

where $P(X_j = 0 \mid X_i = 0) = P(X_j = 1 \mid X_i = 1) = 1 - \epsilon$ and $P(X_j = 1 \mid X_i = 0) = P(X_j = 0 \mid X_i = 1) = \epsilon$. ϵ is a parameter chosen to be 0.2. $P(A_{i,j} \mid X_i = x, X_j = x_j)$ comes from our appearance model, with the assumption that the only true filaments correspond to $X_i = X_j = 1$.

The initialization of each x_i is done according to the posterior probability of the edges going through it. If there is an edge with $\mu_1(a_{i,j}) > \mu_0(a_{i,j})$, then $x_i = 1$. The termination condition for the loop is that all points are visited without any flip, or that the number of flips excess ten times the number of points. In practice the second condition is never met, and only $10 - 20\%$ of the points flip their hidden variable.

4 Results

In this section we first describe the images we are using. We then compare the discriminative power of our dendrite model against simple grayscale thresholding and the baseline Hessian based method [6]. Finally, we validate our automated tree reconstruction results by comparing them against a manual delineation.

4.1 Image Data

Our image database consists of six neuron image stacks, in two of which the dendritic tree has been manually delineated. We use one of those trees for training and the other for validation purposes.

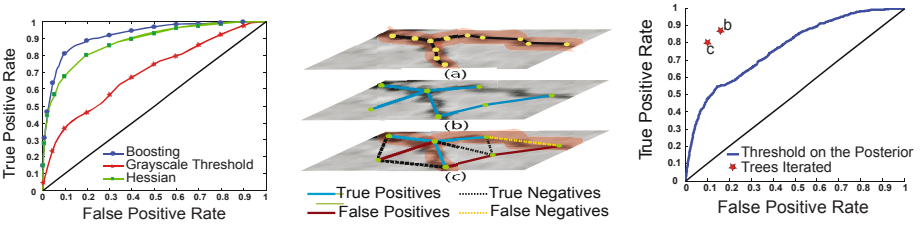
The neurons are taken from the somatosensory cortex of Wistar-han rats. The image stacks are obtained with a standard brightfield microscope. Each image of the stack shows a slice of the same piece of tissue at a different depth. The tissue is transparent enough so that these pictures can be acquired by simply changing the focal plane.

Each image stack has an approximate size of $5 * 10^9$ voxels, and is downsampled to a size of 10^8 voxels to make the evaluation of the image functional in every voxel computationally tractable. After down-sampling, each voxel has the same width, height and depth, of $0.8\,\mu\mathrm{m}$.

4.2 Image Functional Evaluation

The f classifier of 1 is trained using the manual delineation of Fig. 2. As positive samples, we retain 500 voxels belonging to filaments of width ranging from two to six voxels and different orientations. As negative samples, we randomly pick 1000 voxels that are no closer to a neuron than three times the neuron width and are representative of the image noise. Since the training set contains filaments of many different orientations, Adaboost produces a classifier that is orientation independent.

Fig. 2 depicts the candidate dendrite voxels obtained by performing non maxima suppression of images calculated by simply thresholding the original images, computing



(a) (b) (c)

Fig. 5. (a) ROC curve for all three measures using the validation data of figure 4(d). The boosting classifier outperforms the baseline hessian method of [6] in noisy brightfield images. (b) Defining a metric to compare our results against a manual delineation. Top: portion of a manual delineation in which the vertices are close to each other and the tolerance width painted in red. Middle: Portion of the tree found by our algorithm at the same location. Bottom: The fully-connected graph we use to evaluate our edge appearance model and plot the corresponding ROC curves. (c) ROC curve for the detection of edges on filament obtained by thresholding the individual estimated likelihood of the edges of the graph of (b). The individual points represent the iterations of the tree reconstruction algorithm. Two of them are depicted by Fig. 4(b,c). After five iterations we reach a fixed point, which is our final result.

a Hessian-based measure [6], or computing the output of our classifier at each voxel. The same procedure is applied in the validation data of Fig 4(d). Considering correct the vertices that are within $5\,\mu\mathrm{m}$ (6 voxels) of the neuron, we can plot the three ROC curves of Fig. 5(a) that show that our classifier outperforms the other two.

4.3 Tree Reconstruction

To evaluate the quality of the tree, we compare it against the validation data of Fig. 4(d), which is represented as a set of connected points. As shown in Fig. 5(a,b), performing this comparison is non-trivial because in the manual delineation the vertices are close to each other whereas our algorithm allows for distant points to be connected.

To overcome this difficulty, we introduce a measure of whether an edge linking X_i to X_j is present in the manual delineation. First, we use the manually labeled points to construct a volume in which every voxel closer than $5\,\mu\mathrm{m}$ to one such point is assigned the value 1, and 0 otherwise. We then compute the average value in the straight line linking X_i and X_j in that volume. If it is greater than a threshold, we consider that the edge is described by the graph. Here, we take the threshold to be 0.8.

Given this measure, labeling the edges of the tree returned by our algorithm as true or false positives is straightforward. However, since we also need to compute rates of true and false negatives to build ROC curves such as the one of Fig. 5, we create graphs such as the one depicted by Fig. 5(c) in which each vertex is connected to all its nearest neighbors.

In Fig. 5, we plot a ROC curve obtained by thresholding the likelihood that the edges of the graph of Fig. 5(c) belong to a neuron based on the edge appearance model of Section 3.3. Note that this model is not very discriminative by itself. The individual points in Fig. 5 represent true and false positive rates for the successive trees built by the procedure of Section 3.4 and depicted by Fig. 4(b,c,d). As the iterations proceed, the false

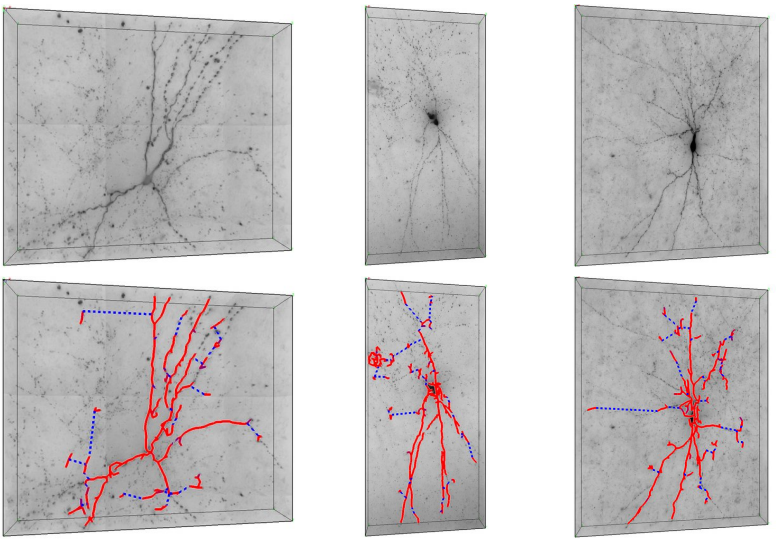


Fig. 6. Three additional reconstructions without annotations. Top row: Image stacks. Bottom row: 3D Dendritic tree built by our algorithm. As in Fig. 4, the edges drawn with a solid red lines are those likely to belong to a dendrite given their appearance. The edges depicted with dashed blue lines are kept to enforce the tree structure through all the vertices. This figure is best viewed in color.

positive rate is progressively reduced. Unfortunately, so is the true positive rate as we loose some of the real dendrite edges. However, the main structure remains and cleaning up this result by hand is much faster than manually delineating the tree of Fig. 4(e).

In Fig. 6, we show reconstruction results in four more image stacks. Our algorithm recovers the main dendrites despite their irregularities and the high noise level and, again, cleaning up this tree is much easier than producing one from scratch. Some of incorrect edges are also retained because the minimum spanning algorithm enforces connectivity of all the vertices, even when it is not warranted.

4.4 From Dendrites to Blood Vessels

Since we learn filament models as we go, one of the strengths of our approach is its generality. To demonstrate it, we ran our algorithm on the retina images of Fig. 7 and 8 without any changes, except for the fact that we replaced the 3D weak classifiers of Section 3.2 by 2D ones, also based on Haar wavelets. The algorithm learned both a local blood-vessel model and 2D filament model.

In Fig. 7(b), we evaluate the performance of our boosted classifier against that of other approaches discussed in [19]. It performs similarly to most of them, but a bit worse than the best. This can be attributed to the fact that it operates at a single scale and is optimized to detect large vessels, whereas the others are multiscale. As a consequence, when we run the full algorithm we obtain the results of Fig. 8 in which the large vessels are correctly delineated, but some of the small ones are missed. This would be fixed by training our initial classifier to handle different widths.

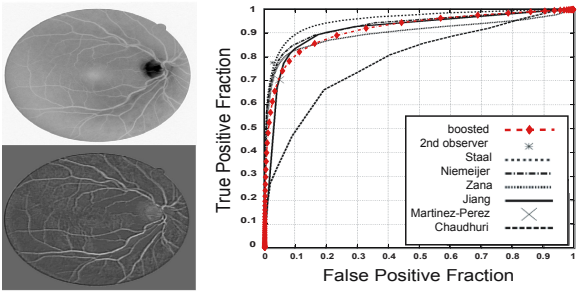


Fig. 7. (a) Top: image of the retina. Bottom: response of our boosting classifier in this image. (b) Comparison of our classifier against other algorithms evaluated in the DRIVE database [19]. It performs similarly to most of them, but worse than algorithms designed specifically to trace blood vessels in images of the retina. This can be attributed to the fact that our boosted classifier operates at a single scale and is optimized to detect large vessels, whereas the others are multiscale.

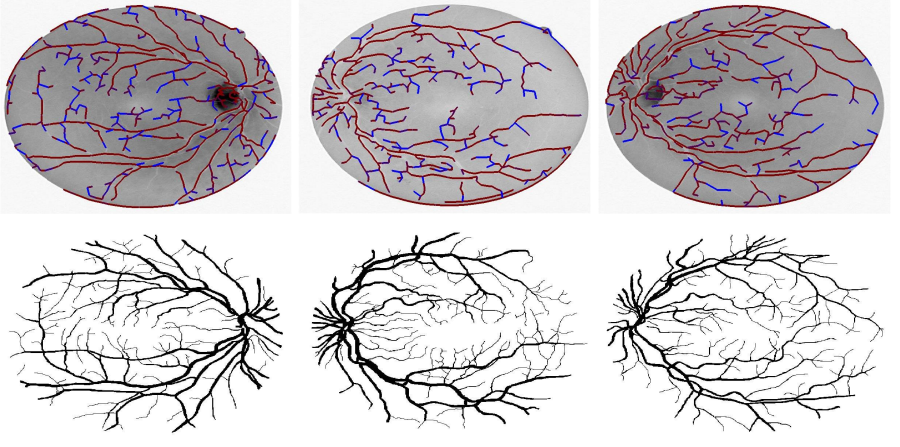


Fig. 8. Retinal trees reconstructed with our method. Top row: original image with the reconstructed tree overlay. As in Fig. 6, edges likely to belong to filaments are drawn in red, while edges kept to enforce the tree structure are colored in blue. Bottom row: manually obtained ground truth. Note that thick filaments are correctly delineated, whereas thin filaments are prone to errors because our classifier is trained only for the thick ones.

5 Conclusion

We have proposed a novel approach to fully-automated 3D delineation of dendritic networks in noisy brightfield images based on statistical machine learning techniques and tree-optimization methods.

By contrast to state-of-the-art methods, we do not postulate *a priori* models for either the dendrite or the edge model between dendrite-like voxels. Instead, we generate

the dendrite measure using discriminative machine learning techniques. We model the edges as a gaussian mixture model, whose parameters are learned using E-M on neuron-specific samples.

To demonstrate the generality of the approach, we showed that it also works for blood vessels in retinal images, without any parameter tuning.

Our current implementation approximates the maximum likelihood dendritic tree under the previous models by means of minimum spanning trees and markov random fields. Those techniques are very easy to compute, but tend to produce artifacts. In future work we will replace them by more general graph optimization techniques.

References

1. Al-Kofahi, K., Lasek, S., Szarowski, D., Pace, C., Nagy, G., Turner, J., Roysam, B.: Rapid automated three-dimensional tracing of neurons from confocal image stacks. IEEE Transactions on Information Technology in Biomedicine (2002)
2. Tyrrell, J., di Tomaso, E., Fuja, D., Tong, R., Kozak, K., Jain, R., Roysam, B.: Robust 3-d modeling of vasculature imagery using superellipsoids. Medical Imaging 26(2), 223–237 (2007)
3. Kirbas, C., Quek, F.: Vessel extraction techniques and algorithms: A survey. In: Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering, p. 238 (2003)
4. Krissian, K., Kikinis, R., Westin, C.F.: Algorithms for extracting vessel centerlines. Technical Report 0003, Department of Radiology, Brigham and Women's Hospital, Harvard Medical School, Laboratory of Mathematics in Imaging (September 2004)
5. Sato, Y., Nakajima, S., Atsumi, H., Koller, T., Gerig, G., Yoshida, S., Kikinis, R.: 3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. Medical Image Analysis 2, 143–168 (1998)
6. Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A.: Multiscale vessel enhancement filtering. In: Wells, W.M., Colchester, A.C.F., Delp, S.L. (eds.) MICCAI 1998. LNCS, vol. 1496, pp. 130–137. Springer, Heidelberg (1998)
7. Streekstra, G., van Pelt, J.: Analysis of tubular structures in three-dimensional confocal images. Network: Computation in Neural Systems 13(3), 381–395 (2002)
8. Meijering, E., Jacob, M., Sarria, J.C.F., Steiner, P., Hirling, H., Unser, M.: Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. Cytometry Part A 58A(2), 167–176 (2004)
9. Aguet, F., Jacob, M., Unser, M.: Three-dimensional feature detection using optimal steerable filters. In: Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP 2005), Genova, Italy, September 11-14, 2005, vol. II, pp. 1158–1161 (2005)
10. Dima, A., Scholz, M., Obermayer, K.: Automatic segmentation and skeletonization of neurons from confocal microscopy images based on the 3-d wavelet transform. IEEE Transaction on Image Processing 7, 790–801 (2002)
11. Schmitt, S., Evers, J.F., Duch, C., Scholz, M., Obermayer, K.: New methods for the computer-assisted 3d reconstruction of neurons from confocal image stacks. NeuroImage 23, 1283–1298 (2004)
12. Agam, G., Wu, C.: Probabilistic modeling-based vessel enhancement in thoracic ct scans. In: CVPR 2005: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 2, pp. 684–689. IEEE Computer Society, Washington (2005)

13. Santamaría-Pang, A., Colbert, C.M., Saggau, P., Kakadiaris, I.A.: Automatic centerline extraction of irregular tubular structures using probability volumes from multiphoton imaging. In: Ayache, N., Ourselin, S., Maeder, A. (eds.) MICCAI 2007, Part II. LNCS, vol. 4792, pp. 486–494. Springer, Heidelberg (2007)
14. Al-Kofahi, K.A., Can, A., Lasek, S., Szarowski, D.H., Dowell-Mesfin, N., Shain, W., Turner, J.N., et al.: Median-based robust algorithms for tracing neurons from noisy confocal microscope images (December 2003)
15. Flasque, N., Desvignes, M., Constans, J., Revenu, M.: Acquisition, segmentation and tracking of the cerebral vascular tree on 3d magnetic resonance angiography images. Medical Image Analysis 5(3), 173–183 (2001)
16. McIntosh, C., Hamarneh, G.: Vessel crawlers: 3d physically-based deformable organisms for vasculature segmentation and analysis. In: CVPR 2006: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1084–1091. IEEE Computer Society Press, Washington (2006)
17. Szymczak, A., Stillman, A., Tannenbaum, A., Mischaikow, K.: Coronary vessel trees from 3d imagery: a topological approach. Medical Image Analisys (08 2006)
18. Freund, Y., Schapire, R.: Experiments with a New Boosting Algorithm. In: International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann, San Francisco (1996)
19. Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., van Ginneken, B.: Ridge based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging 23, 501–509 (2004)