Segment 2D and 3D Filaments by Learning Structured and Contextual Features

Lin Gu, Xiaowei Zhang, He Zhao, Huiqi Li, and Li Cheng

***Abstract*— We focus on the challenging problem of filamentary structure segmentation in both 2D and 3D images, including retinal vessels and neurons, among others. Despite the increasing amount of efforts in learning based methods to tackle this problem, there still lack proper data-driven feature construction mechanisms to sufficiently encode contextual labelling information, which might hinder the segmentation performance. This observation prompts us to propose a data-driven approach to learn structured and contextual features in this paper. The structured features aim to integrate local spatial label patterns into the feature space, thus endowing the follow-up tree classifiers capability to grouping training examples with similar structure into the same leaf node when splitting the feature space, and further yielding contextual features to capture more of the global contextual information. Empirical evaluations demonstrate that our approach outperforms state-of-the-arts on well-regarded testbeds over a variety of applications. Our code is also made publicly available in support of the open-source research activities.**

I. Introduction

The problem of segmenting 2D and 3D image-based filaments is crucial in a wide range of applications, including neuronal reconstruction and tracing in microscopic images [1], blood vessel tracing in fundus images [2], [3], human vasculature segmentation in 2D digital subtraction angiography and 3D magnetic resonance angiography images [4], to name a few. Existing filament segmentation (depending on the context, also referred to as vessel segmentation or reconstruction, curvilinear structure segmentation in literature) methods can be roughly grouped into two types: model-based and learning-based. Hessian-based models characterize filament edges [5] by the second order derivatives. However, they could be awkward in tackling irregular-shaped filamentary structures, such as irregular cross sections caused by imaging noise or non-uniform staining. Meanwhile, other model-based methods work by fitting filaments with known geometric shapes. One example is the widely used optimally oriented flux (OOF) [6] method, which is based on the assumption of circular filament cross-sections. This idea is further extended by Turetken *et al.* [7] to segment filamentary structures via a set of regularly-spaced anchor points. Recently, we have evidenced an increasing development of learning-based methods [2], [8], [9], [10], [11], [12], which tackle the challenging irregular-shaped filamentary structures by exploiting similar patterns from training samples in a supervised manner. In [8], a boosting framework is proposed to learn filters that often lead to the state-of-the-art performance. On the other hand, there still lack proper data-driven mechanisms to construct features or filters that sufficiently encode contextual labelling information, which might be the bottleneck that hinders the segmentation performance. Due to the vast literature on filament segmentation, it is not possible to mention all important research efforts. Interested readers may consult [3], [13], [14] for more thorough reviews.

Despite these research efforts, it remains challenging to precisely segment 2D and 3D image-based filaments. This is evidenced by e.g. the recent BigNeuron initiative [15] that calls for innovations in addressing the demands from neuronal science community where a significant number of neuronal images have been routinely produced in wet labs, while there still lack sufficiently accurate tools to automatically segment the neurite structures. To address this challenge, we propose in this paper a dedicated pipeline by learning structured and contextual features from data. In practice, our approach has outperformed existing state-of-the-art methods by a noticeable margin on testbeds of 2D and 3D neuronal and retinal segmentation applications. The main contributions of our work are as follows: First, a novel scheme is developed to learn structured features, each encodes a distinct local spatial label pattern. Moreover, this feature construction scheme enables the incorporation of features of variable sizes and locations into a single feature vector, as illustrated in the left hand side of Fig. 1. Compared to the otherwise more involved multi-resolution approaches, it is simple to construct, and these heterogeneous features are acquired and normalized in a unified and natural manner. In addition, a set of context distance features involving tree leave indices is proposed to capture more of the global contextual information. Second, our feature construction scheme and in particular the context distance features work specifically well with the boosted tree classifiers. Practically our approach is shown to be capable of delivering superior performance over existing state-of-the-arts on a variety of application benchmarks. Last but not least, to support the open-source convention our package is also made publicly available [1].

II. Related Work

The work of [8] also learns features from data. Our approach is however quite different. Instead of each feature being single label-pixel based, we consider features each being related to

Lin Gu and Xiaowei Zhang are with Bioinformatics Institute, A*STAR, Singapore. E-mail: {gulin, zhangxw}@bii.a-star.edu.sg

He Zhao is with Bioinformatics Institute, A*STAR, Singapore, and Beijing Institute of Technology, China. E-mail: zhaoh@bii.a-star.edu.sg

Huiqi Li is with Beijing Institute of Technology, China. E-mail: huiqili@bit.edu.cn

Li Cheng is with Bioinformatics Institute, A*STAR, Singapore. E-mail: chengli@bii.a-star.edu.sg

[1]The code and detailed information can be found at a dedicated project webpage `http://web.bii.a-star.edu.sg/~zhangxw/learnStructFeatures/index.htm`.

a local label patch centered around current pixel of interest. This allows us to learn structured features in term of modeling similarity patterns among pixels of the patch, as well as discriminative features that retains label information. Moreover, a context distance feature is proposed to include more global contextual information when making local decisions. In what follows, we also provide a succinct review of related machine learning topics.

Instead of manual feature engineering, feature learning aims to automatically extract features from data that are discriminative and ideally interpretable. The visual Bag-of-Words (VBoW) methods (e.g. [16], [17]) are probably the most widely-used feature learning techniques, which has been extended to accommodate spatial co-occurrences of features, capture relative positions of codewords, and have multi-resolution capacities [18], [19], [20]. For instance, sparse coding has been employed in [21] to learn appearance filters. In addition, a number of other methods have been developed. Kernel boost [8] learns pixel-wise discriminative features at each stage of the gradient boosting in the form of linear filters. In [22], a regression-based approach has been proposed for centerline detection. These methods nevertheless encounter difficulties in explicitly modeling the spatial structured label information. Rather than examining only the label of current pixel, the auto-context features in [23] are learned to yield spatially consistent results, where a cascade approach is used to support the realization of iterative prorogation of label information in inputs during feature generation. The very recent structured feature learning paradigm also seeks a feature map that considers spatial-neighboring labels together with spatial-neighboring pixel observations. The structured forest efforts of [24], [25] present such examples where a label patch is considered as the output space when learning the feature maps. The neural network and the more recent deep learning approaches such as [11], [26], [27], [28] emphasize on implicit learning of feature representations that are sufficiently discriminative for prediction purposes, where it is also relatively convenient to incorporate structured label information by the back-prorogation trick. On the other hand, they often lack the interpretability of their learned features which are the internal network weights, which might not be desirable for domain experts. We note in the passing that, to a degree, the idea of learning structured features is also related with the topic of structured prediction, for which we refer the readers to a comprehensive survey [29].

The design of our proposed features and especially the context distance feature makes them mostly suitable for boosted tree classifiers. Therefore, it is meaningful to give a brief overview about this research line here. In particular, we focus on boosting [30], [31], which has wide applications and various variants e.g. AdaBoost [32], LogitBoost [33], gradient boosting [34], probabilistic boosted trees [35]. In what follows, we provide a concise account of the gradient boosting trees (also known as gradient boosted regression trees) as background context.

Given training data $\{\boldsymbol{f}_i, y_i\}_{i=1}^N$ where $\boldsymbol{f}_i \in \mathbb{R}^n$ denotes feature vector with n features and $y_i \in \{+1, -1\}$ denotes the corresponding label. In our context $y_i = 1$ denotes a filament sample while $y_i = -1$ refers to the background. Gradient boosting trees are composed of an ensemble of weak decision trees $h_j(\boldsymbol{f})$, which collectively predict the target value of input data $\boldsymbol{f}$ by a function $F_M(\boldsymbol{f})$ defined as

$$F_M(\boldsymbol{f}) = \sum_{j=1}^{M} \gamma_j h_j(\boldsymbol{f}). \tag{1}$$

The weak decision tree $h_j(\boldsymbol{f})$ is iteratively added to minimize the loss as $h_j = \arg\min_h \sum_i^N L(F_{j-1}(\boldsymbol{f}_i) + h(\boldsymbol{f}_i), y_i)$, where for the loss function $L(\cdot, \cdot)$ we adopt the widely used exponential loss $L(F_j(\boldsymbol{f}_i), y_i) = \exp(-y_i F_j(\boldsymbol{f}_i))$. Specifically, in each iteration, we minimize a quadratic approximation of the loss function in the following steepest descent strategy: In each iteration j, we train a decision tree $h_j(\boldsymbol{f})$ to minimize $\sum_{i=1}^N w_i^j (h(\boldsymbol{f}_i) - r_i^j)^2$, where $w_i^j = \nabla_F^2 L(F_{j-1}(\boldsymbol{f}_i), y_i)$ and $r_i^j = -\nabla_F L(F_{j-1}(\boldsymbol{f}_i), y_i)/w_i^j$ denotes the gradient descent direction. To grow a decision tree, we choose splitting function $t(\boldsymbol{f})$ that selects a single feature in $\boldsymbol{f}$, as well as a threshold τ. A training sample is assigned to the left child if $t(\boldsymbol{f}) < \tau$, otherwise assigned to the right child. We exhaustively search all n choices for $t(\boldsymbol{f})$ and seek the optimal τ to minimize the quantity $\sum_{i, t(\boldsymbol{f}_i) < \tau} w_i^j (r_i^j - \eta_l)^2 + \sum_{i, t(\boldsymbol{f}_i) \geq \tau} w_i^j (r_i^j - \eta_r)^2$, where η_l and η_r are the mean values of r_i^j in the left and right child nodes, respectively.

III. Our Approach

In this section, we describe details of our new approach, including techniques used to extract structured features and context distance features as well as the training of boosted tree classifiers. The pipeline of our approach is shown in Fig. 1. We first develop a scheme for structured feature learning in Subsection III-A, aiming to integrate local spatial label patterns into the feature space. Then, we feed the resulting features to train two boosted tree classifiers, as shown in Fig. 2. The first boosted tree classifier is used to obtain context distance features, as described in Subsection III-B. The resulting context distance features, together with structured features, are used to train the second boosted tree classifier which is adopted for testing image segmentation.

A. Structured Feature Learning

To start with, a set of N representative patches or cubes (for 2D or 3D images, respectively) of the *same* size is randomly selected from the training images, as is presented in the left-most panel of Fig. 1. The patches or cubes, together with their corresponding labels, aggregate to form a set of training image patches or cubes $\{\boldsymbol{p}_i, y_i\}_{i=1}^N$, where $y_i \in \{-1, 1\}$ is the label of the central pixel in $\boldsymbol{p}_i$. For each patch or cube $\boldsymbol{p}_i$, we randomly select m (e.g., $m = 1024$) pairs of labels for pixels in $\boldsymbol{p}_i$ and construct a m-dimensional binary vector based on these label pairs. Here, the element of the binary vector contains either 1 if the label pair are of the same type, or 0 if they differ. This gives rise to a $N \times m$ matrix $\hat{Y}$ that contains a rich set of spatially structured label information. To further reduce the dimensionality, principal component analysis (PCA) is subsequently performed on $\hat{Y}$ to reduce the dimension from m to l (In our experiments, we choose

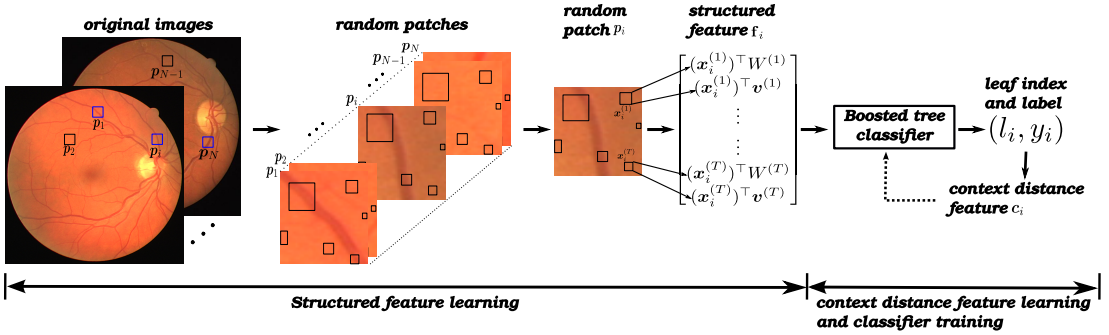


Fig. 1: Pipeline of our approach, which consists of two main components: The first component is for learning structured features as described in Subsection III-A, the second component is for learning context distance features and training boosted tree classifiers as described in Subsection III-B. Details of the second component are shown in Fig. 2, while the growing of a single tree as well as an illustration of why structured features are useful are shown in Fig. 3.

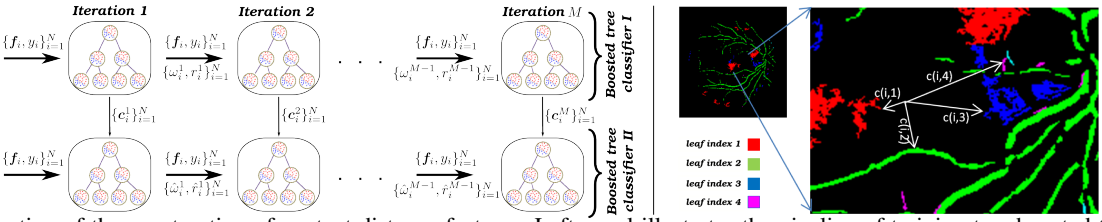


Fig. 2: Illustration of the construction of context distance features. Left panel illustrates the pipeline of training two boosted tree classifiers. Right panel presents the construction of context distance features for a specific pixel, where only top d_c context distance features are preserved. See text for details. Best viewed in color.

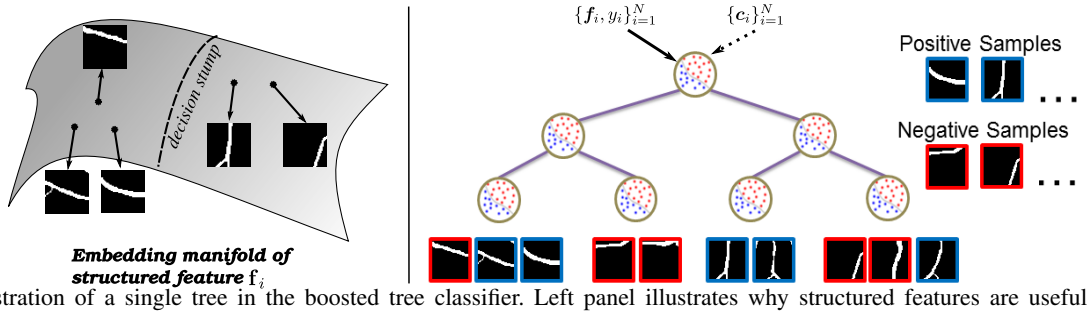


Fig. 3: Illustration of a single tree in the boosted tree classifier. Left panel illustrates why structured features are useful for partitioning patches containing similar filamentary structures into the same leaf node. Right panel shows the function of a single tree, which corresponds to a tree in boosted tree classifier I when input contains only $\{\boldsymbol{f}_i, y_i\}_{i=1}^N$ and corresponds to a tree in boosted tree classifier II when input contains both $\{\boldsymbol{f}_i, y_i\}_{i=1}^N$ and $\{\boldsymbol{c}_i\}_{i=1}^N$.

$l = 10$), resulting a matrix Y satisfying $Y^\top Y = \Sigma$, with $\Sigma \in \mathbb{R}^{l\times l}$ being a diagonal matrix consisting of the first l largest singular values of $\hat{Y}^\top \hat{Y}$.

To take topological structure into account, for each $\boldsymbol{p}_i$, we randomly select a sub-patch or sub-cube $\boldsymbol{x}_i \in \mathbb{R}^d$, then learn a linear filter $W \in \mathbb{R}^{d\times l}$ such that the learned feature $\boldsymbol{x}_i^\top W$ has consistent topological structure as Y_i, with $Y = [Y_1 \ \cdots \ Y_N]^\top \in \mathbb{R}^{N\times l}$. Mathematically, we would like to learn the optimal filter W by solving

$$\min_{W\in\mathbb{R}^{d\times l}} \sum_{i=1}^N \|\boldsymbol{x}_i^\top W - Y_i^\top\|_2 + \lambda R_w(W) \quad \text{s.t.} \sum_{i=1}^N W^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top W = \Sigma, \tag{2}$$

where $R_w(W)$ is a regularization term and $\lambda > 0$ is a parameter controlling the data fidelity and the model complexity. In our experiment, we choose the fused Lasso [36] as the regularization, that is $R_w(W) = \sum_{j=1}^l \sum_{i=1}^{d-1} |W_{i+1j} - W_{ij}|$, to impose smoothness in the filter. Denote $X = [\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times d}$, and define the $\ell_{1,2}$-norm as $\|A\|_{1,2} := \sum_{i=1}^N \|\boldsymbol{a}_i\|_2$ for any matrix $A = [\boldsymbol{a}_1 \cdots \boldsymbol{a}_N]^\top \in \mathbb{R}^{N\times l}$, then optimization problem in (2) can be reformulated as

$$\min_{W\in\mathbb{R}^{d\times l}} \|XW - Y\|_{1,2} + \lambda R_w(W) \quad \text{s.t. } W^\top X^\top X W = \Sigma. \tag{3}$$

The construction of Y is inspired by [25], aiming to measure the similarity of pixels within a patch over the label space. In this way, the features learned by W can model similarity patterns among pixels in an image patch. A main difference between [25] and model (3) is that [25] uses Y to facilitate the computation of information gain in random forests while model (3) exploits structured information in Y and uses it to extract structured features. We also introduce the orthogonality constraint to model the topological structure between different patches. Moreover, one important reason for us to adopt the $\ell_{1,2}$-norm here as loss function is that the $\ell_{1,2}$-norm is known to be robust to outliers in data points, as shown in e.g. [37].

The final piece of our structured feature is a discriminative feature vector. To ensure this feature being as uncorrelated with the rest features (i.e. W) as possible, we penalize the correlation during feature learning. Formally, we solve

$$\min_{\boldsymbol{v}\in\mathbb{R}^d} \|X\boldsymbol{v} - \boldsymbol{y}\|_1 + \rho R_v(\boldsymbol{v}) + \frac{\mu}{2}\|W^\top X^\top X\boldsymbol{v}/N\|_2^2, \tag{4}$$

where $\boldsymbol{y} = [y_1, \cdots, y_N]^\top$ is the label vector, $R_v(\boldsymbol{v})$ is a regularization term, and $\rho > 0$ and $\mu > 0$ are tuning parameters (In our experiments, we use $R_v(\boldsymbol{v}) := \sum_{j=1}^{d-1} |\boldsymbol{v}_{j+1} - \boldsymbol{v}_j|$ to impose smoothness in $\boldsymbol{v}$, and let $\mu = 0.1$.). In the

last term of (4), we use $W^\top X^\top X\boldsymbol{v}/N$ to approximate the sample covariance matrix between topological features $\boldsymbol{x}^\top W$ and discriminative feature $\boldsymbol{x}^\top \boldsymbol{v}$, and attempt to minimize the correlation. Similar to the model of (3), we use ℓ_1-norm as the loss function, instead of least square loss, to make sure the model in (4) is robust to outliers.

The alternating direction method of multipliers (ADMM) is applied to solve both optimization problems (3) and (4). We repeat T times the random selection process of image sub-patches (In our experiments $T = 200$), which yields linear filters $\{W^{(k)}\}_{k=1}^T$ and $\{\boldsymbol{v}^{(k)}\}_{k=1}^T$. Therefore, for each image patch or cube $\boldsymbol{p}_i$, the following feature vector is constructed:

$$\boldsymbol{f}_i := [(\boldsymbol{x}_i^{(1)})^\top W^{(1)},\ (\boldsymbol{x}_i^{(1)})^\top \boldsymbol{v}^{(1)}, \cdots,\ (\boldsymbol{x}_i^{(T)})^\top W^{(T)},\ (\boldsymbol{x}_i^{(T)})^\top \boldsymbol{v}^{(T)}],$$

which is also illustrated in Fig. 1.

It is worth mentioning that features learned by W and $\boldsymbol{v}$ contain quite different information of the current pixel. On one hand, the construction of Y enables W to learn similarity patterns of the neighbor pixels in the label space, while the orthogonality constraint enables W to model topological structure among different patches. On the other hand, feature $\boldsymbol{v}$ provide necessary discriminative information since $\boldsymbol{y}$ alone records only the label of the current pixel instead of difference of labels. In addition, due to the equality constraint in problem (3), the resulting feature vectors $\{\boldsymbol{f}_i\}_{i=1}^N$ must lie on a manifold. By exploiting such constraints, our structured features can work well with tree structured classifiers. As shown in Fig. 3, feature vectors with similar structure lie close to each other on the manifold, and the decision stumps of a tree amount to partitioning the curved space of the manifold, which force image patches with similar filamentary structure to the same leaf node. A standard application of decision trees is to compute posterior probability for classification or mean values in leaf nodes for regression, and usually ignores the leaf index. In the next subsection, we show how to take advantage of leaf index information to construct context distance feature aiming to capture global contextual information.

B. Context Distance Feature

We learn a boosted tree classifier (Boosted tree classifier I in Fig. 2) to construct context distance features as follows: Firstly, we grow a decision tree using $\{\boldsymbol{f}_i, y_i\}_{i=1}^N$, and index all leaf nodes. Secondly, we input all training images into the tree. Since each pixel will be clustered into one leaf node, we get a index map for each image as shown in the left panel of Fig. 2, where we highlight pixels in different leaf nodes with different colors. Therefore, for each patch $\boldsymbol{p}_i$, we get the leaf index l_i recording the leaf node into which the central pixel of $\boldsymbol{p}_i$ is clustered. Lastly, for each patch $\boldsymbol{p}_i$ we compute the distance from its central pixel to each leaf node, where the distance is computed as the Euclidean distance between the central pixel and the nearest pixel within the leaf node. Therefore, for patch $\boldsymbol{p}_i$ we get a distance feature vector $\boldsymbol{c}_i$ whose length equals to the number of leaf nodes. Iterate this process M times, we can grow M trees and get a collection of context distance feature vectors $\{\boldsymbol{c}_i^j\}_{i=1,\cdots,N}^{j=1,\cdots,M}$ as well as leaf indices $\{\boldsymbol{l}_i\}_{i=1}^N$ where $\boldsymbol{l}_i = [l_i^1, \cdots, l_i^M]$ represents the structure label of $\boldsymbol{p}_i$.

Note in each iteration of classifier I, a new set of context distance features are learned and used to train the corresponding tree in classifier II, as shown in Fig. 2. In the implementation, for the computational cost to compute the whole context distance feature vector in 3D space or large 2D space would be computationally expensive, we only collect the context distance features for the top d_c leaf nodes of the highest weight, that is, keep only d_c components in each distance feature vector $\boldsymbol{c}_i$. For the sake of clarity, we use a toy example to illustrate the construction process of context distance feature, as follows. Suppose we have trained a tree at some iteration of classifier I, for a given image we learn structured feature for each pixel, and by processing the image, each pixel passes through a particular path of the tree and lands onto a leaf node. The weight of a leaf node is calculated as the number of pixels in the node, among which we select the top $d_c = 4$ leaf nodes of the highest weight. Then, for an arbitrary pixel x in the image, we find its nearest pixel $x_1, \cdots, x_4$ from the selected 4 nodes, respectively. The context distance feature of pixel x is defined as $c_x = [dist(x, x_1), \cdots, dist(x, x_4)]$, where $dist$ denotes the Euclidean distance of two pixels in an image.

At the same time of constructing context distance features, we also train a second a boosted tree classifier (Boosted tree classifier II in Fig. 2). The training of classifier II is the same as that of classifier I except that when growing each tree in classifier II, we input both the structured features $\{\boldsymbol{f}_i\}_{i=1}^N$ and context distance features $\{\boldsymbol{c}_i\}_{i=1}^N$. In the testing phase, we use classifier I to construct context distance features and classifier II to perform segmentation. In the experiments, we demonstrate that introducing the context distance feature would significantly improve the segmentation performance.

IV. Empirical Evaluation

To examine the effectiveness of the proposed approach, in what follows a series of empirical experiments are carried out on different applications, including 2D retinal vessel segmentation, 2D neuronal segmentation, as well as 3D neuronal segmentation, all on widely-used testbeds. We start with introducing the experimental configuration.

A. Experimental Configurations

Our Approach and Variants: As discussed in Fig. 1, our pipeline contains the learning of two types of features, namely the structured features, or **SF** in short, as well as the context distance features, which we may refer to as **SF + context distance**, which is the complete version of our approach.

Datasets and Evaluation Metrics: Six sets of publicly available datasets are employed, each dedicates to one application, as follows.

Four datasets are engaged for the task of 2D retinal vessel segmentation, including DRIVE [38], STARE [39], CHASEDB1 [40] and HRF [41]. DRIVE dataset [2] contains 40 fundus images of size 584×565, while STARE dataset [3]

[2]DRIVE dataset can be downloaded at `http://www.isi.uu.nl/Research/Databases/DRIVE/`.

[3]STARE dataset can be downloaded at `http://www.ces.clemson.edu/~ahoover/stare/`.

contains 20 fundus images of size 605×700. CHASEDB1 dataset [4] contains eye images of 14 children, which were captured using 8 bit color channel with a resolution of 1280×960 pixels, yielding 28 images in total. Two experts' segmentations are available for each images as ground-truth. HRF dataset [5] contains 45 images, of which one third are images of healthy patients, one third are patients with diabetic retinopathy and the rest are images of glaucomatous patients. Binary gold standard vessel segmentation images, generated by a group of experts, are available for each image. The size of fundus images is 3304×2336. Regarding the training/testing partition, DRIVE dataset has its own partition, while for other datasets we use the first half as training subset and the other half as testing subset.

Then, to facilitate the analysis of 2D neuronal segmentation systems, the neuronal dataset [2] [6] is utilized. This dataset contains 112 images of in total 675 neurons. The image size is within the range of 512×572. The same training and testing splits in [2] are adopted in our experiments.

Finally, to demonstrate the application of our approach on 3D neuronal segmentation, the Gold166 dataset shared by BigNeuron initiative [15] is engaged here. The Gold166 dataset consists of 79 3D neuronal images along with the corresponding manual annotations. The sizes of the 3D images or image stacks vary from $511 \times 511 \times 597$ to $1024 \times 1024 \times 62$. As some annotations of the image stacks might not be proper, e.g. annotated filaments are visibly diverging from the 3D point clouds in raw data, or the annotated 3D filaments are noticeably much thinner than others in the dataset to match up consistently with the 3D point clouds, we then end up with a subset of 34 image stacks [7] by filtering away the ones with questionable annotations. Among them, 17 images are randomly selected to form the training set, and the rest form the testing set.

Evaluation Metric: To evaluate the performance, we follow the common practice in e.g. [42], [2] to adopt the standard F1 measure (computed as $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$) and the precision-recall curve in our experiments. We also compute the Specificity as $\frac{TN}{TN+FP}$, and Matthews Correlation Coefficient (MCC) [43] as $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$, where TP, TN, FP, FN refer to true positives, true negatives, false positives and false negatives, respectively. For the 3D neuronal (i.e. Gold166) dataset, we only use F1 measure. In particular, to account for the near-boundary annotation issue [8], a tolerance factor σ is introduced. Similar to that of [42], [44], [22], this tolerance zone is used here to exclude the influence of these near-boundary voxels from the boundary of human-annotated 3D filaments outward within σ voxels. In other words, only voxels outside this zone are considered during performance evaluation. It is worth noting in the passing that we have in fact examined different schemes (one such alternative scheme is presented in the supplementary materials.) to counter the effect of near-boundary annotation issue, and found out all leads to similar results. Throughout the experiments on 3D Neuronal dataset, we report the F1 measure computed using tolerance $\sigma = 2$.

TABLE I: An evaluation of various related features and classifiers on DRIVE. Two values separated by "/" are F1 measures in percentage (%) obtained with / without context distance features. See text for details.

	Gradient boosting	AdaBoost	LogitBoost	Random forest
SF	**78.56 / 76.82**	**74.40 / 70.82**	**78.33 / 76.61**	**73.22 / 68.77**
raw feature	76.88 / 76.62	69.63 / 62.00	76.10 / 76.09	62.36 / 49.86
SE [25]	73.52 / 71.31	63.83 / 59.49	72.63 / 71.31	62.73 / 58.13
Gabor	62.96 / 67.12	56.75 / 62.32	62.13 / 63.91	54.83 / 60.80

Comparison Methods: For tasks of 2D Retinal and Neuronal Segmentation, a range of state-of-the-art methods are considered covering both supervised and unsupervised ones. They are: (1) Kernel Boost [8] that utilizes gradient boosting to learn convolutional features from data; (2) Optimally Oriented Flux (OOF) [6] that uses manual filters to delineate tubular structures; (3) IUWT [5] that is based on isotropic undecimated wavelet transform, to segment 2D image in a unsupervised manner; (4) Eigen [4], which is a multiscale Hessian-based unsupervised method for 2D segmentation; (5) T2T [45], which is a supervised 2D segmentation system that integrates pixel classification, medial sub-tree generation and global tree linking; (6) structured edge (SE), the structured edge detection technique of [25]; (7) B-COSFIRE [43] that responds to vessels selectively by computing the weighted geometric mean of the outputs by applying a pool of Difference-of-Gaussian filters. (8) LCMBoost [2] which utilizes an iterative learning-based approach to boost the performance of an existing base segmentation method.

Unlike our approach, most existing 2D filament segmentation methods could not work with 3D problems. Thus, for 3D neuronal segmentation, our approach is compared with a different set of state-of-the-art methods, as follows: (1) Adaptive Enhancement [46] is a Hessian-based method dedicated to 3D neuronal segmentation by detecting salient features via adaptive context windows; (2) GWDT [47] is a 3D neuronal segmentation method based on a region-growing scheme; (3) Regression Tubularity [22], which could be considered as an extension of Kernel Boost [8], formulates the linear structure centreline detection as a regression problem. These three standard 3D segmentation methods could produce a probability map, thus allowing us to obtain the F1 measure as well as the precision-recall curve. As these methods come with the BigNeuron Gold166 dataset, their default parameters are assumed to be optimal and are used as is in our experiments.

B. Results and Analysis

Classifiers and Features: As discussed previously, a number of boosted tree classifiers are applicable in our pipeline. A systematic study is thus conducted on DRIVE dataset to gauge the performance of incorporating related classifiers and typical features. As in Table I, empirically gradient boosting delivers the overall best results in our context, when comparing to

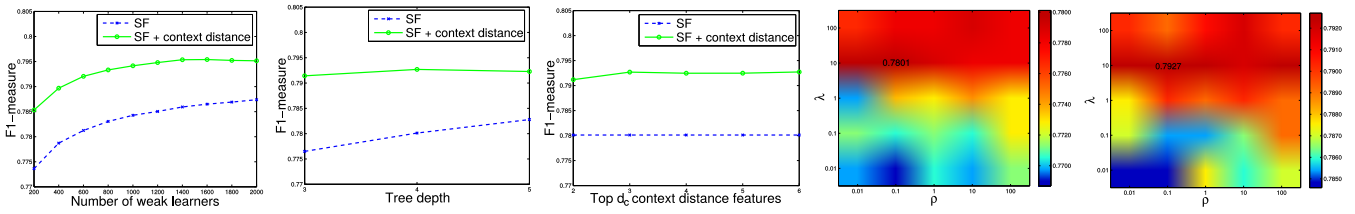
[4]CHASEDB1 dataset can be downloaded at `https://blogs.kingston.ac.uk/retinal/chasedb1/`.

[5]HRF dataset can be downloaded at `https://www5.cs.fau.de/research/data/fundus-images/`.

[6]Downloadable from `http://web.bii.a-star.edu.sg/~zhaoh/data/2D_Neuron_dataset.zip`.

[7]Downloadable from `http://web.bii.a-star.edu.sg/~zhangxw/learnStructFeatures/3D_Neuron_dataset.zip`.

[8]As shown in Fig. 7(a) & 7(f), human annotation is often too conservative to envelop in the actual 3D filamentary point clouds. In other words, the ground-truth label for 3D filaments could be smaller than the size it should be, thus is not sufficiently accurate in terms of segmentation.



(a) F1 vs number of weak learners M (b) F1 vs tree depth d_t (c) F1 vs number of context distance features d_c (d) F1 using SF only vs parameters λ and ρ (e) F1 using SF + context distance vs parameters λ and ρ

Fig. 4: Parameter sensitivity analysis results. Each panel presents the F1 measure as a function a specific parameter while the rest parameters remain unchanged at the default values.

options including AdaBoost, LogitBoost, and random forest. This motivates us to adopt gradient boosting throughout our experiments. On the feature side, four distinct feature types are considered, including our structured features (SF), raw features, structured edge (SE) features, and the widely-used Gabor features. The raw feature is simply the image patch $\boldsymbol{p}_i$ centered around the target pixel as defined in Section III-A. The SE features are obtained by the structured edge detection technique of [25], which includes three CIE-LUV color channels, two magnitudes and eight gradient channels as well as 13 pair-wise self similarity features. From Table I, clearly our structured features outperform the alternative (raw or SE) features on all classifiers considered. This is due to the fact that our structured features could capture both discriminate and topology structure information. Furthermore, the incorporation of context distance features would usually lead to additional improvement. It is noted that although raw features sometimes achieve nearly comparable performance, its combination with the context distance feature then fails to further advance the performance, as raw features could not facilitate their context distance features in term of exploiting topology and structure information as what structured features can do. The performance of raw features on other datasets are not consistently as well. For example, on STARE dataset, the raw features achieve a F1 measure of 73.29%, while in comparison SF alone has already 77.48%, which is 4.19% higher.

1) Influence of Internal Parameters: Our approach contains a handful of internal parameters, including the number of weak learners M, tree depth d_t of the weak learner, the number d_c of top context distance features we used, and regularization parameters λ and ρ in optimization problems (3) and (4), as also illustrated in Fig. 1, Fig. 2 and Fig. 3. Now, we proceed to evaluate its robustness with respect to the change of parameter values. While experiments are conducted only on the DRIVE dataset, practical observation suggests similar trend on other datasets, so what we have shown here is quite representative. To avoid overfitting issue, a small fraction (10 images) of the DRIVE training images are randomly picked and retained as a validation set. Our parameter sensitivity analysis experiments are thus evaluated on this validation set. To investigate the effect of a specific parameter, we assign various values to the parameter while keeping all other parameters at the default value, and compute the F1 measure of our methods. The following default values are used in this section: $M = 500$, $d_t = 4$, $d_c = 3$, $\lambda = 10$ and $\rho = 0.1$. As indicated in Fig. 4, overall our approach is often robust with respect to varying parameter values, such as M, d_t, d_c. It is relatively more sensitive to the value of λ and ρ. In what follows we give more detailed analysis.

Number of Weak Learners: To show how the performance changes with respect to tree number M, we evaluate the two variants of our approach under varying values of M and plot the corresponding F1 measure in Fig. 4(a). We observe that the performance improves continuously with the growing number of trees before considerably slowing down after around $M = 500$. In practice, we choose to use $M = 500$ to tradeoff between performance and computational burden.

Tree Depth d_t and Top d_c Context Distance Features: For each regression tree in the gradient boosting, the tree depth also matters. As suggested in Fig. 4(b), similar to the number of trees, deeper tree might improve the performance but at a cost of higher computational demand. Similar pattern is also observed in Fig. 4(c) when using different values of d_c. In practice, we fix $d_t = 4$ and $d_c = 3$.

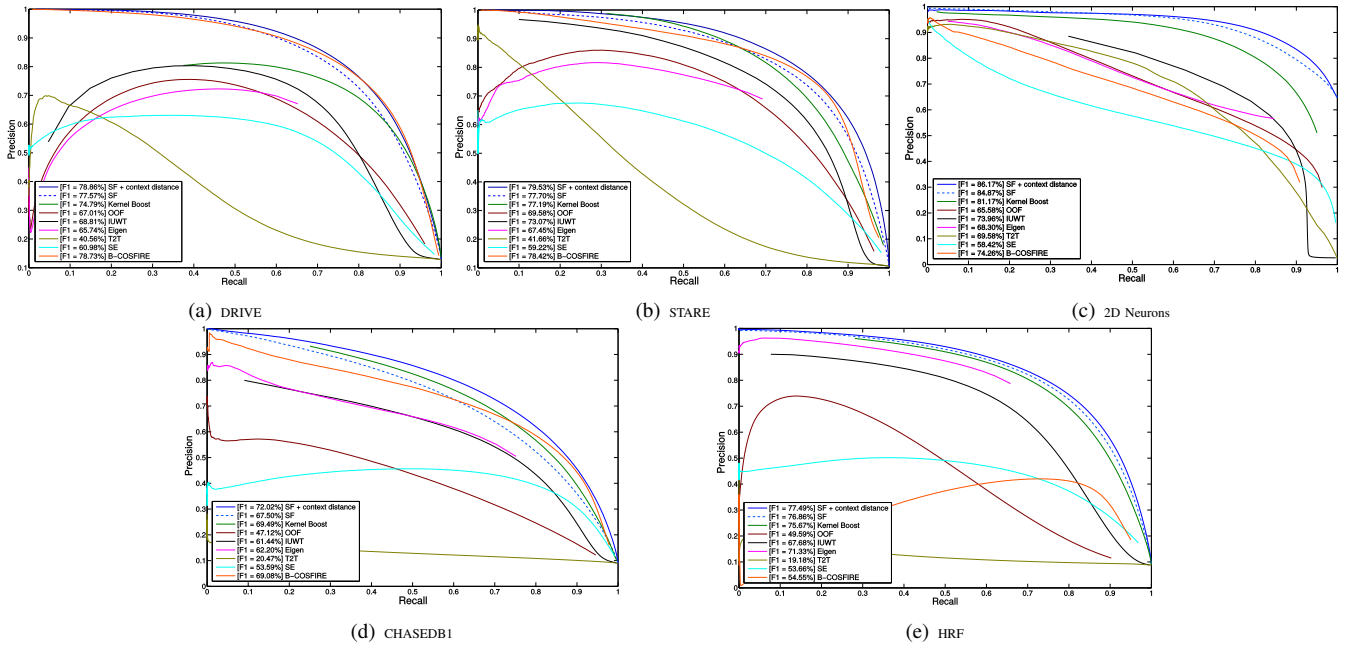
λ and ρ in Learning Structured features (SF) and Context Distance Features: Structured and context distance features are produced using different λ and ρ values from set $\{0.01, 0.1, 1, 10, 100\}$. As displayed in Fig. 4(d) and Fig. 4(e) for SF and SF + context distance, respectively, the performance of our approach is relatively stable, as the overall change of F1 measures is less than 3%. In practice we set $\lambda = 10$, and $\rho = 0.1$.

2) 2D Retinal and Neuronal Segmentation: Table II summarizes the performance statistics of the competing 2D segmentation methods on five datasets, namely DRIVE, STARE, 2D Neurons, CHASEDB1, and HRF. Additionally, Figure 5 presents the more detailed precision-recall curves on each of these datasets. Representative visual results are displayed in Fig. 6 which shows (a) the raw image, (b) ground-truth, (c) the posterior probability estimated by our SF + context distance variant and (d) its result, (e) the result of our SF only variant, as well as two most competing methods, (f) Kernel Boost [8] and (g) SE [25]. As shown in Table II, in terms of F1 measure, the SF variant of our approach alone is capable of slightly outperforming the best state-of-the-art methods, Kernel Boost, B-COSFIRE and LCMBoost, on all testbeds. Besides, the full version of our approach, namely SF + context distance, outperforms the SF variant by a noticeable margin of 1% to 2% higher F1 scores on average. This exemplifies the usefulness of the context distance features in our approach.

Fig. 5(a) presents the precision-recall curves of comparison methods on DRIVE dataset. Our SF + context distance variant significantly outperforms existing methods in a wide range of

TABLE II: Performance statistics of 2D segmentation using F1 measure (%), Precision (%), Recall (%), Specificity (%) and MCC.

		Method									
		SF	**SF + context distance**	**Kernel Boost** [8]	**OOF** [6]	**IUWT** [5]	**Eigen** [4]	**T2T** [45]	**SE** [25]	**B-COSFIRE** [43]	**LCMBoost** [10]
DRIVE	F1 measure	77.57 ± 2.16	**78.86** ± 2.15	74.79 ± 2.67	67.01 ± 3.12	68.81 ± 3.31	65.74 ± 4.85	40.56 ± 2.26	60.98 ± 2.75	78.73 ± 1.95	75.74 ± 3.57
	Precision	79.31 ± 2.65	**80.50** ± 2.53	71.65 ± 3.57	65.76 ± 4.40	69.23 ± 4.71	67.43 ± 4.71	42.72 ± 3.82	55.33 ± 3.74	78.87 ± 2.21	78.59 ± 6.87
	Recall	75.95 ± 2.65	77.33 ± 2.68	**78.30** ± 2.49	68.42 ± 2.72	68.57 ± 3.74	64.82 ± 8.16	38.80 ± 2.31	68.23 ± 4.14	78.67 ± 3.14	74.11 ± 7.42
	Specificity	97.11 ± 0.46	97.28 ± 0.37	95.50 ± 0.58	94.81 ± 0.76	95.57 ± 0.66	95.43 ± 0.95	92.31 ± 1.52	91.95 ± 1.15	96.93 ± 0.39	**97.41** ± 1.16
	MCC	0.7442 ± 0.0241	**0.7589** ± 0.0239	0.7106 ± 0.0293	0.6214 ± 0.0352	0.6436 ± 0.0370	0.6116 ± 0.0496	0.3244 ± 0.0282	0.5511 ± 0.0273	0.7567 ± 0.0210	0.7322 ± 0.0349
STARE	F1 measure	77.70 ± 6.19	**79.53** ± 5.59	77.19 ± 6.35	69.58 ± 6.40	73.07 ± 5.62	67.45 ± 9.17	41.66 ± 4.81	59.22 ± 5.74	78.42 ± 4.11	78.13 ± 5.35
	Precision	77.61 ± 7.80	79.89 ± 6.73	78.61 ± 6.07	70.26 ± 6.88	74.60 ± 6.90	71.93 ± 3.75	43.53 ± 6.86	52.67 ± 4.94	77.78 ± 4.55	**82.33** ± 5.58
	Recall	77.91 ± 4.85	**79.24** ± 4.67	75.94 ± 7.18	68.98 ± 6.21	71.66 ± 4.74	66.10 ± 7.04	40.24 ± 4.15	67.94 ± 8.39	79.18 ± 4.75	75.19 ± 9.83
	Specificity	97.41 ± 1.01	97.74 ± 0.74	97.60 ± 0.86	96.63 ± 0.97	97.19 ± 0.84	96.67 ± 1.72	93.78 ± 1.92	92.86 ± 1.92	97.36 ± 0.71	**98.04** ± 0.93
	MCC	0.7519 ± 0.0649	**0.7724** ± 0.0588	0.7470 ± 0.0657	0.6613 ± 0.0672	0.7008 ± 0.0587	0.6488 ± 0.0791	0.3525 ± 0.0441	0.5448 ± 0.0574	0.7593 ± 0.0438	0.7614 ± 0.0527
2D Neurons	F1 measure	84.87 ± 3.92	**86.17** ± 3.33	81.17 ± 4.67	65.58 ± 5.29	73.96 ± 4.37	68.30 ± 4.50	69.58 ± 4.44	58.42 ± 6.33	74.26 ± 3.79	83.96 ± 3.78
	Precision	83.19 ± 4.65	**85.02** ± 3.86	80.58 ± 4.97	60.33 ± 6.97	69.71 ± 7.19	62.39 ± 7.15	66.26 ± 6.30	48.82 ± 7.92	69.65 ± 6.49	74.96 ± 7.32
	Recall	86.71 ± 3.87	87.45 ± 3.83	81.92 ± 5.49	72.57 ± 6.31	79.41 ± 4.79	76.76 ± 7.81	73.73 ± 4.98	74.06 ± 6.49	80.08 ± 3.90	**96.71** ± 3.47
	Specificity	99.51 ± 0.30	**99.57** ± 0.27	99.45 ± 0.34	98.71 ± 0.72	99.07 ± 0.49	98.76 ± 0.65	99.02 ± 0.47	97.96 ± 0.91	99.09 ± 0.44	99.19 ± 0.36
	MCC	0.8448 ± 0.0402	**0.8581** ± 0.0347	0.8068 ± 0.0487	0.6504 ± 0.0518	0.7353 ± 0.0431	0.6804 ± 0.0438	0.6894 ± 0.0447	0.5866 ± 0.0577	0.7384 ± 0.0367	0.8441 ± 0.0321
CHASEDB1	F1 measure	67.50 ± 3.77	**72.02** ± 3.11	69.49 ± 3.06	47.12 ± 3.46	61.44 ± 3.10	62.20 ± 4.08	20.47 ± 1.69	53.59 ± 4.07	69.08 ± 3.08	–
	Precision	66.60 ± 3.89	**71.08** ± 2.89	69.32 ± 2.59	41.32 ± 4.64	59.80 ± 4.08	57.95 ± 5.28	13.16 ± 1.41	42.48 ± 4.84	64.74 ± 3.66	–
	Recall	68.50 ± 4.32	73.03 ± 3.72	69.71 ± 3.97	55.28 ± 3.18	67.04 ± 2.61	67.40 ± 3.92	47.14 ± 5.54	73.19 ± 3.14	**74.18** ± 3.77	–
	Specificity	96.64 ± 0.49	**97.10** ± 0.38	96.97 ± 0.52	92.08 ± 2.14	94.98 ± 0.86	95.14 ± 1.19	69.25 ± 5.87	90.19 ± 1.69	96.00 ± 0.86	–
	MCC	0.6432 ± 0.0379	**0.6928** ± 0.0314	0.6652 ± 0.0312	0.4167 ± 0.0403	0.5758 ± 0.0333	0.5745 ± 0.0442	0.1005 ± 0.0200	0.5007 ± 0.0370	0.6604 ± 0.0331	–
HRF	F1 measure	76.86 ± 4.53	**77.49** ± 4.66	75.67 ± 5.10	49.59 ± 6.52	67.68 ± 6.06	71.33 ± 6.60	19.18 ± 1.94	53.66 ± 5.10	54.54 ± 5.46	–
	Precision	77.75 ± 5.28	**78.25** ± 5.47	76.42 ± 6.12	50.96 ± 8.29	69.65 ± 8.09	79.85 ± 6.83	13.16 ± 1.88	44.40 ± 6.39	41.55 ± 5.67	–
	Recall	76.02 ± 4.13	**76.78** ± 4.16	74.99 ± 4.38	48.44 ± 4.92	65.97 ± 4.45	65.09 ± 8.33	36.54 ± 4.26	68.58 ± 3.74	79.92 ± 3.84	–
	Specificity	97.95 ± 0.41	97.99 ± 0.43	97.82 ± 0.47	95.49 ± 0.96	97.25 ± 0.73	**98.43** ± 0.59	76.27 ± 5.84	91.61 ± 1.93	89.24 ± 0.92	–
	MCC	0.7472 ± 0.0474	**0.7541** ± 0.0489	0.7343 ± 0.0535	0.4495 ± 0.0696	0.6477 ± 0.0644	0.6959 ± 0.0623	0.0856 ± 0.0196	0.4971 ± 0.0515	0.5218 ± 0.0511	–



(a) DRIVE (b) STARE (c) 2D Neurons

(d) CHASEDB1 (e) HRF

Fig. 5: Precision-recall curves of comparison methods on (a) DRIVE, (b) STARE, (c) 2D Neurons, (d) CHASEDB1, and (e) HRF, respectively. Best viewed in color.

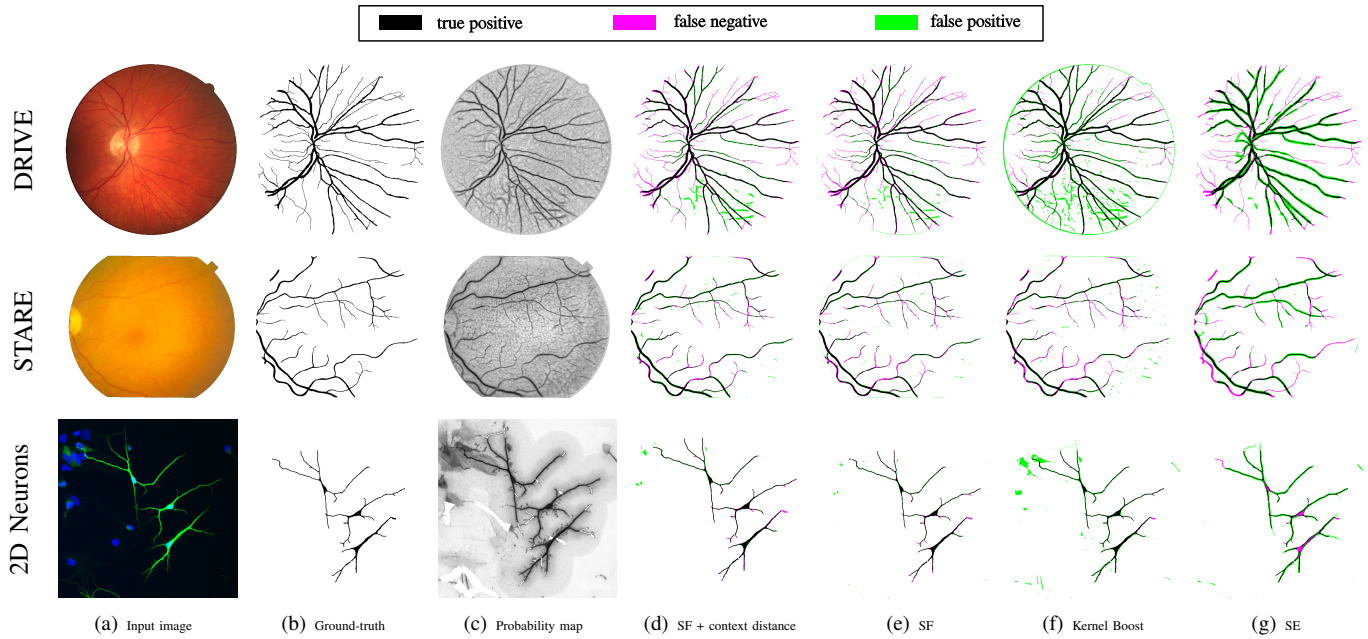
TABLE III: Comparison evaluation focusing specifically on the optical disk region, as illustrated in the Exemplar optical disk region masks of supplementary Fig. 1. Numbers are in percentage (%) of their F1 measure.

	SF	SF + context distance	Kernel Boost [8]	OOF [6]	IUWT [5]	Eigen [4]	T2T [45]	SE [25]	B-COSFIRE [43]	LCMBoost [2]
DRIVE	81.64 ± 2.79	**83.27** ± 2.49	80.60 ± 4.26	71.79 ± 3.52	69.73 ± 4.50	67.18 ± 4.31	48.85 ± 3.06	65.76 ± 3.76	82.91 ± 2.87	78.27 ± 4.04

areas, especially during the recall value range of $[0.6, 0.8]$, which corresponds to the most useful zone in most practical applications. The best existing method, Kernel Boost method, shows slightly better precision when the recall is more than 0.95, which is too extreme to be of any practical use. On the other hand, Kernel Boost are far inferior to both of our variants in most of the cases (i.e. in the recall range of $[0, 0.9]$). A representative example presented in the first row of Fig. 6 provides visual evidence that our approach is capable of suppressing false alarms and missing ones, while securing reasonable amount of true positive vessel foregrounds, when compared to existing methods like Kernel Boost and SE.

Fig. 5(b) again illustrates the superior performance of our SF + context distance variant over any other methods on STARE dataset. This point is also illustrated in Fig. 6 that both of our variants successfully avoid the ambiguous vessels while recover equivalent amount of detailed vessels. It is worth noting that, in this dataset, our SF only variant is sufficiently effective in high recall region when compared to Kernel Boost. Additionally, a 5-fold cross-validation is conducted on STARE, where the average F1 measure is 78.30% for SF + context distance feature, and 76.41% for SF, which is consistent with our aforementioned experiments where the first 10 images were used for training.

For the rest datasets, as presented in Fig 5(c), Fig 5(d), and Fig 5(e), our SF + context distance again outperforms other methods by a large margin, which nicely coincides the performance summary of Table II, where our approach



(a) Input image (b) Ground-truth (c) Probability map (d) SF + context distance (e) SF (f) Kernel Boost (g) SE

Fig. 6: Exemplar results on segmenting 2D retinal and neuronal images. (a): Input images; (b): Ground-truth; (c): Probability maps of our approach (SF + context distance variant); (d & e): Error images of our approach (SF + context distance & SF only); (f): Error images of Kernel Boost; (g): Error images of SE. Here green denotes false alarm and the magenta denotes the missing error. Best viewed in color.

overtakes other competing methods most of the times.

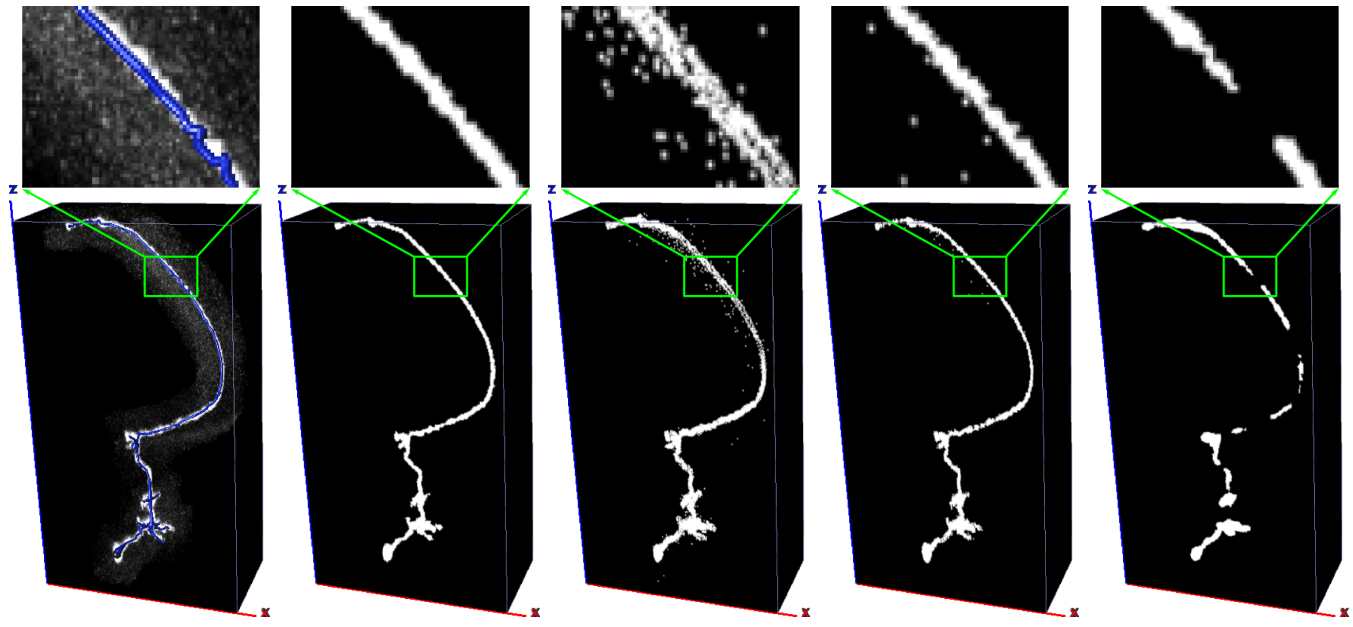
Exemplar results on 2D retinal and neuronal datasets are presented in Fig. 6, which provides visual evidence that our approach outperforms existing methods like Kernel Boost and SE. We also observe that the 2D neuronal dataset seems to pose less challenge than the retinal datasets. This could be possibly due to the fact that the neurons dyed in fluorescent protein exhibit a clearer and relatively more visible boundaries. On the other hand, both variants of our approach are able to ignore the out-of-focus neurons in the background while Kernel Boost is unable to tell them apart from the high quality neurons.

In addition, experiments are also carried out to focus specifically on the optic disk region. This helps to examine the performance of our approach specifically on this difficult region, where existing methods are often performing less well due to the existence of packed nerve heads. A masked area is centered around the optical disk with a radius of 100 pixels is applied to extract the region of interest, as illustrated in Fig.1 of the supplementary file. Table III displays the quantitative F1 results. Similarly, our complete approach still outperforms rest competing methods.

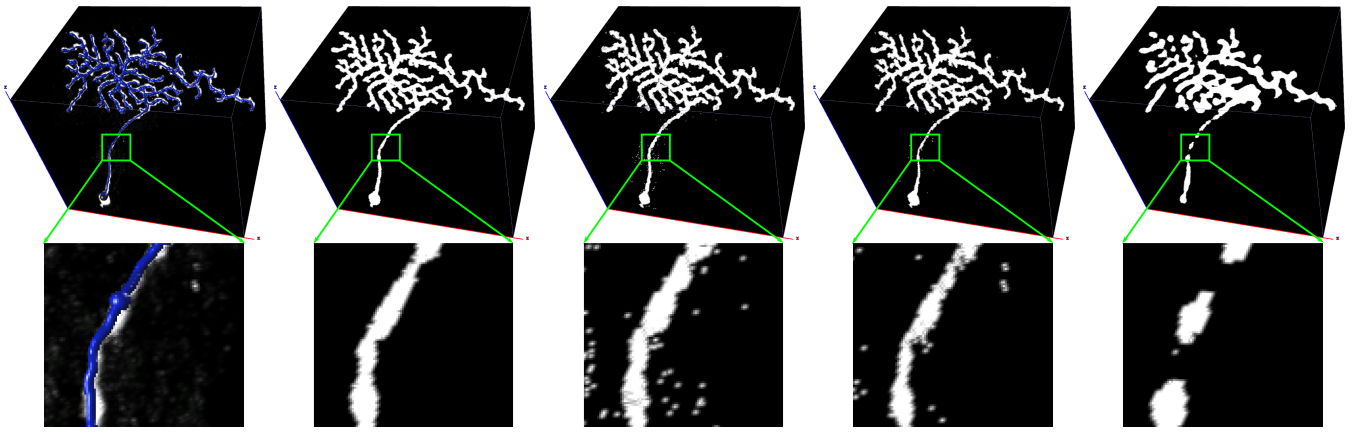
3) 3D Neuronal Segmentation: Here, the experiment focuses on the task of 3D neuronal segmentation using the Gold166 dataset. Exemplar visual results are presented in Fig. 7, where the input images are overlayed by ground-truth annotations in blue color as presented in the leftmost column. It can be observed from especially the zoom-ins that, compared with the three state-of-the-art segmentation methods (namely Adaptive Enhancement, GWDT, and Regression Tubularity), our filamentary structure predictions are more capable of filtering away background noises while still retaining the connectedness along the neuron branches. Quantitative results are presented as the precision-recall curves in Fig. 8, where our approach (i.e. SF + context distance) outperforms the best comparison method, GWDT, by a noticeable margin most of the time, and is only overtaken by GWDT when recall is over 92% and precision is below 50%. The SF variant of our approach performs only slightly inferior to the SF + context distance variant and is clearly the second best. It is also observed that the Regression Tubularity is able to achieve the highest precision at the cost of missing much of the neurons. One reason is that this approach is particularly designed to produce the maximal response at the centreline of the curvilinear object, thus making it sometimes less sensitive to the whole body. Table IV summarizes the F1 measures as well as of the error bars (i.e. 1-standard deviation) of the quantitative evaluation.

C. Time Complexity Analysis

The complexity of our algorithm consists of three main parts: computing structured features, computing context distance features and training gradient boosting classifiers. We focus on the first two parts since the complexity of training gradient boosting classifiers has been extensively studied (e.g. [31]) and hence omitted. To compute both structured features W and $\boldsymbol{v}$, we employed ADMM, which is an iterative algorithm, we assume the numbers of iterations are given by N^W_{iter} and N^v_{iter}, respectively. Then, the complexity of computing both W and $\boldsymbol{v}$ is given by $O((N+d)\min(N,d)^2 + (N+d)dlN^W_{iter} + ((d+l)l^2 + (N+d)d)N^v_{iter})$. Computing context distance feature vector $\boldsymbol{c}^j_i$ for the center pixel of patch $\boldsymbol{p}_i$ in the j-th tree requires $O(d\sum_{k=1}^{d_c} N_k)$, where N_k denotes the number of pixels in the top k-th leaf node. Thus, the total complexity of computing $\{\boldsymbol{c}^j_i\}^{j=1,\cdots,M}_{i=1,\cdots,N}$ is given by $O(dMN\sum_{k=1}^{d_c} N_k)$. We note that in practice the time of computing structured features usually dominates that of computing context distance features. In testing phase, on 2D dataset DRIVE, the average computational time of **SF** and **SF + context distance** is 173.3 seconds and 350.0 seconds per image, respectively; on the



(a) Input image with ground-truth (b) SF + context distance (c) Adaptive Enhancement (d) GWDT (e) Regression Tubularity



(f) Input image with ground-truth (g) SF + context distance (h) Adaptive Enhancement (i) GWDT (j) Regression Tubularity

Fig. 7: Exemplar 3D neuronal segmentation results on Gold166 dataset. (a & f): Input images with ground-truth in blue; (b & g): Results of our SF + context distance variant; (c & h): Results of Adaptive Enhancement [46]; (d & i): Results of GWDT [47]; (e & j): Results of Regression Tubularity [22].

TABLE IV: Comparison of 3D neuronal segmentation methods on Gold166 dataset using F1 measure (%) with tolerance $\sigma = 2$.

SF	SF + context distance	Adaptive Enhancement [46]	GWDT [47]	Regression Tubularity [22]
79.38 ± 9.75	**79.89** ± 9.33	58.53 ± 14.27	75.77 ± 10.52	65.75 ± 12.48

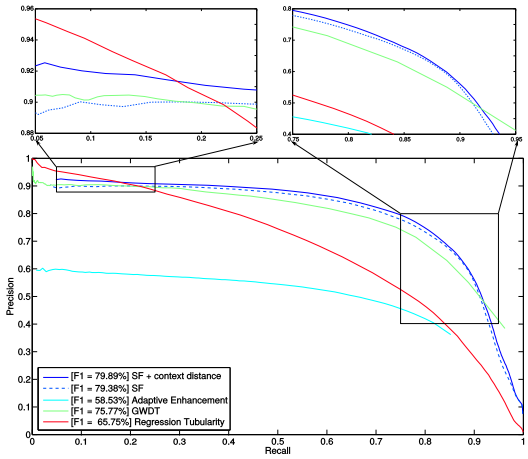


Fig. 8: Precision Recall Curves of 3D neuronal segmentation methods on Gold166 dataset.

3D dataset, the average computational time of **SF** and **SF + context distance** is 243.2 seconds and 486.7 seconds per image, respectively.

V. Conclusion

We present a supervised feature learning approach for filamentary structure segmentation. Empirical evaluations on 2D and 3D applications demonstrate the competitiveness of our approach. For future work, we aim to investigate its applications in e.g. 2D and 3D human vasculature segmentation.

Acknowledgements

This research was supported by A*STAR JCO grant 1231BFG040.

REFERENCES

[1] K. Brown, G. Barrionuevo, A. Canty, P. De, J. Hirsch, G. Jefferis, J. Lu, M. Snippe, I. Sugihara, and G. Ascoli, "The DIADEM data sets: Representative light microscopy images of neuronal morphology to advance automation of digital reconstructions," *Neuroinformatics*, vol. 9, pp. 143–157, 2011.

[2] J. De, L. Cheng, X. Zhang, F. Lin, H. Li, K. Ong, W. Yu, Y. Yu, and S. Ahmed, "A graph-theoretical approach for tracing filamentary structures in neuronal and retinal images," *IEEE Trans. Med. Imag.*, vol. 35, pp. 1–17, 2015.

[3] C. Kirbas and F. Quek, "A review of vessel extraction techniques and algorithms," *ACM Computing Surveys*, vol. 36, pp. 81–121, 2000.

[4] A. Frangi, W. Niessen, K. Vincken, and M. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention*, 1998, pp. 130–137.

[5] P. Bankhead, C. Scholfield, J. McGeown, and T. Curtis, "Fast retinal vessel detection and measurement using wavelets and edge location refinement," *PLoS ONE*, vol. 7, pp. 1–12, 2012.

[6] M. Law and A. Chung, "Three dimensional curvilinear structure detection using optimally oriented flux," in *European Conference on Computer Vision*, 2008, pp. 368–382.

[7] E. Turetken, G. Gonzalez, C. Blum, and P. Fua, "Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors," *Neuroinformatics*, vol. 9, pp. 279–302, 2011.

[8] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua, "Supervised feature learning for curvilinear structure segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2013, pp. 526–533.

[9] J. Orlando and M. Blaschko, "Learning fully-connected CRFs for blood vessel segmentation in retinal images," in *Medical Image Computing and Computer-Assisted Intervention*, 2014, pp. 634–641.

[10] L. Gu and L. Cheng, "Learning to boost filamentary structure segmentation," in *International Conference on Computer Vision*, 2015, pp. 639–647.

[11] Q. Li, B. Feng, L. Xie, P. Liang, H. Zhang, and T. Wang, "A cross-modality learning approach for vessel segmentation in retinal images," *IEEE Trans. Med. Imaging*, vol. 35, no. 1, pp. 109–118, 2016.

[12] J. Orlando, E. Prokofyeva, and M. Blaschko, "A discriminatively trained fully connected conditional random field model for blood vessel segmentation in fundus images," *IEEE Transactions on Biomedical Engineering*, 2016, in press.

[13] E. Meijering, "Neuron tracing in perspective," *Cytometry A.*, vol. 77, no. 7, pp. 693–704, 2010.

[14] D. Lesage, E. Angelini, I. Bloch, and G. Funka-Lea, "A review of 3D vessel lumen segmentation techniques: models, features and extraction schemes," *Medical Image Analysis*, vol. 13, no. 6, pp. 819–45, 2009.

[15] H. Peng, E. Meijering, and G. Ascoli, "From DIADEM to BigNeuron," *Neuroinformatics*, vol. 13, no. 3, pp. 259–260, 2015.

[16] O. Cula and K. Dana, "Compact representation of bidirectional texture functions," in *Conference on Computer Vision and Pattern Recognition*, 2001, pp. 1041–1047.

[17] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *International Conference on Computer Vision*, vol. 2, 2003, pp. 1470–1477.

[18] S. Savarese, J. Winn, and A. Criminisi, "Discriminative object class models of appearance and shape by correlatons," in *Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2033–2040.

[19] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky, "Describing visual scenes using transformed objects and parts," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 291–330, 2008.

[20] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.

[21] R. Rigamonti and V. Lepetit, "Accurate and efficient linear structure segmentation by leveraging ad hoc features with learned filters," in *Medical Image Computing and Computer-Assisted Intervention*, 2012, pp. 189–197.

[22] A. Sironi, E. Tretken, V. Lepetit, and P. Fua, "Multiscale centerline detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1327–1341, 2016.

[23] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3D brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1744–1757, 2010.

[24] P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling," in *International Conference on Computer Vision*, 2011, pp. 2190–2197.

[25] P. Dollar and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2015.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1106–1114.

[27] Y. Ganin and V. Lempitsky, "N4-Fields: Neural network nearest neighbor fields for image transforms," in *Assian Conference on Computer Vision*, 2014, pp. 536–51.

[28] P. Liskowski and K. Krawiec, "Segmenting retinal blood vessels with deep neural networks," *IEEE Transactions on Medical Imaging*, 2016, in press.

[29] S. Nowozin and C. Lampert, "Structured learning and prediction in computer vision," *Found. Trends. Comput. Graph. Vis.*, vol. 6, no. 3-4, pp. 185–365, 2011.

[30] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, 1996, pp. 148–156.

[31] R. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[32] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[33] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 1998.

[34] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems*, 2000, pp. 512–518.

[35] Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *International Conference on Computer Vision*, 2005, pp. 1589–1596.

[36] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[37] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization," in *Advances in Neural Information Processing Systems*, 2010, pp. 1813–1821.

[38] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken, "Ridge based vessel segmentation in color images of the retina," *IEEE Trans. Med. Imag.*, vol. 23, no. 4, pp. 501–509, 2004.

[39] A. Hoover, V. Kouznetsova, and M. Goldbaum, "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response," *IEEE Trans. Medical Imaging*, vol. 19, no. 3, pp. 203 –210, 2000.

[40] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman, "An ensemble classification-based approach applied to retinal blood vessel segmentation," *IEEE Trans. on Biomed. Eng.*, vol. 59, no. 9, pp. 2538–48, 2012.

[41] T. Kohler, A. Budai, M. Kraus, J. Odstrcilik, G. Michelson, and J. Hornegger, "Automatic no-reference quality assessment for retinal fundus images using vessel segmentation," in *IEEE Int. Symp. on Computer-Based Medical Systems*, 2013, pp. 95–100.

[42] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *International Conference on Computer Vision*, vol. 2, 2001, pp. 416–423.

[43] G. Azzopardi, N. Strisciuglio, M. Vento, and N. Petkov, "Trainable COSFIRE filters for vessel delineation with application to retinal images," *Medical image analysis*, vol. 19, no. 1, pp. 46–57, 2015.

[44] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International Conference on Machine Learning*, J. Langford and J. Pineau, Eds. New York, NY, USA: ACM, 2012, pp. 567–574.

[45] S. Basu, A. Aksel, B. Condron, and S. T. Acton, "Tree2tree: Neuron segmentation for generation of neuronal morphology," in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, April 2010, pp. 548–551.

[46] Z. Zhou, S. Sorensen, H. Zeng, M. Hawrylycz, and H. Peng, "Adaptive image enhancement for tracing 3D morphologies of neurons and brain vasculatures," *Neuroinformatics*, vol. 13, no. 2, pp. 153–166, 2015.

[47] H. Xiao and H. Peng, "APP2: automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree," *Bioinformatics*, vol. 29, no. 11, pp. 1448–54, 2013.