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

***Abstract*— We focus on the challenging problem of filamentary structured segmentation in both 2D and 3D images, which includes retinal vessels and neurons, among others. In particular, we propose a data-driven approach to learn structured and contextual features. 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 three types: Hessian-based, 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, model-based methods work by fitting filaments with known geometric shapes. The widely used Optimally oriented flux (OOF) [6] method 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 [8], [**?**], [9], [2], [10], which on the other hand tackle the challenging irregular-shaped filamentary structures by exploiting similar patterns from training sample in a supervised manner. In [8], a boosting framework is proposed to learn the filters that often leads 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], [11], [**?**] 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 [12] that calls for innovations in addressing the demands from neuronal sciences 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 Figure 1. Comparing 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].

The most related work might be that of [8] that 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 a local label patch centered around current pixel of interest; We also consider a context distance feature to include more global contextual information when making local decisions. In what follows, we provide a succinct review of related machine learning topics.

Related Machine Learning Literature 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. [13], [14]) are probably the most widely-used feature learning techniques, which possess many variants and extensions: For example, it has been strengthened to accommodate spatial co-occurrences of features [15]; One may also choose to capture relative positions of codewords from a generative model perspective, as in [16]; The spatial pyramid match work of [17] further augments VBoW with spatial and multi-resolution capacities. Rather than examine only the label of current pixel, the very recent structured feature learning paradigm seeks a feature map that consider spatial-neighboring labels together with spatial-neighboring pixel observations.

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

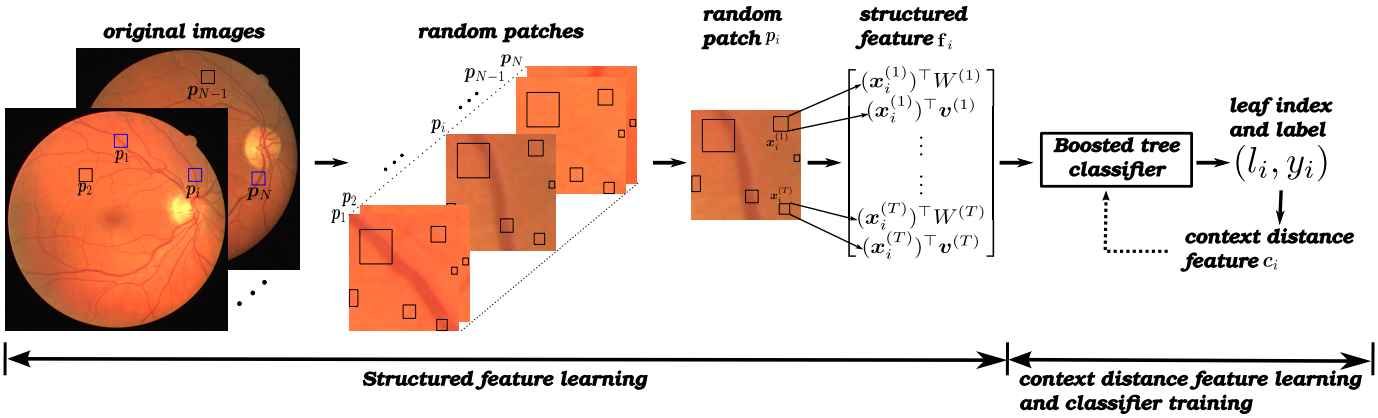


Fig. 1: Pipeline of our approach, which consists of two main components: The first component is for learning structured features as described in Subsection II-A, and the second component is for learning context distance features and training boosted tree classifiers as described in Subsection II-B. Details 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. See text for details.

The structured forest efforts of [18], [19] present such examples where a label patch is considered as the output space when learning the feature maps. The auto-context features in [20] are also related, where a cascade approach is used to support the realization of iterative prorogation of label information in inputs during feature generation. The neural network and the more recent deep learning approaches such as [21], [22], [10] 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 with the back-prorogation trick. On the other hand, they might 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 with structure features is also related with the topic of structured prediction, which we will refer the readers to a comprehensive survey [23].

The design of our proposed features and especially the contexture distance feature make them mostly suitable for boosted tree classifiers. Therefore it is meaningful to give a brief overview about this research line here. It is known that constructing optimal decision tree is a NP-complete problem [24], where optimal tree refer to the one that minimizes the expected number of tests for classification purpose. In practice, one is usually content with learning several weak decision trees and try to learn an ensemble strategy that collectively produce better performance than any individual tree classifier. This gives rise to the seminar work of boosting by Schapire and Freund [25] that affirmatively shows that it is possible to create a single strong learner based on a set of weak learners based on an iterative procedure. The practical success also leads to the development of a variety of boosting variants including e.g. AdaBoost [26], LogitBoost [27], gradient boosting [28], probabilistic boosted trees [29]. 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 $\boldsymbol{f}_i$ by a function $F_M(\boldsymbol{f}_i)$ 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 $\sum_i^N L(F_{j-1}(\boldsymbol{f}_i) + h(\boldsymbol{f}_i), y_i)$, where for the loss function 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.

II. 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 II-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 II-B. The resulting context distance features, together with structured features, are used to

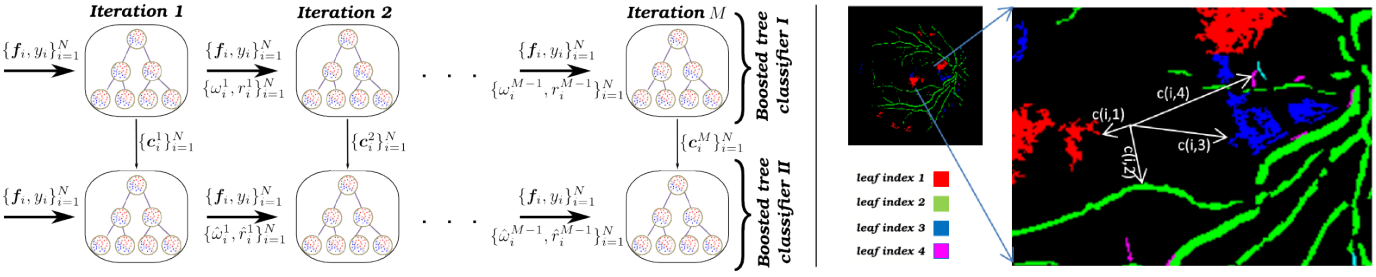


Fig. 2: Illustration of the construction of context distance features. Left panel illustrates the pipeline of training two boosted tree classifiers. Right panel presents the construction of context distance features for a specific pixel, where only top d_c context distance features are preserved. See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

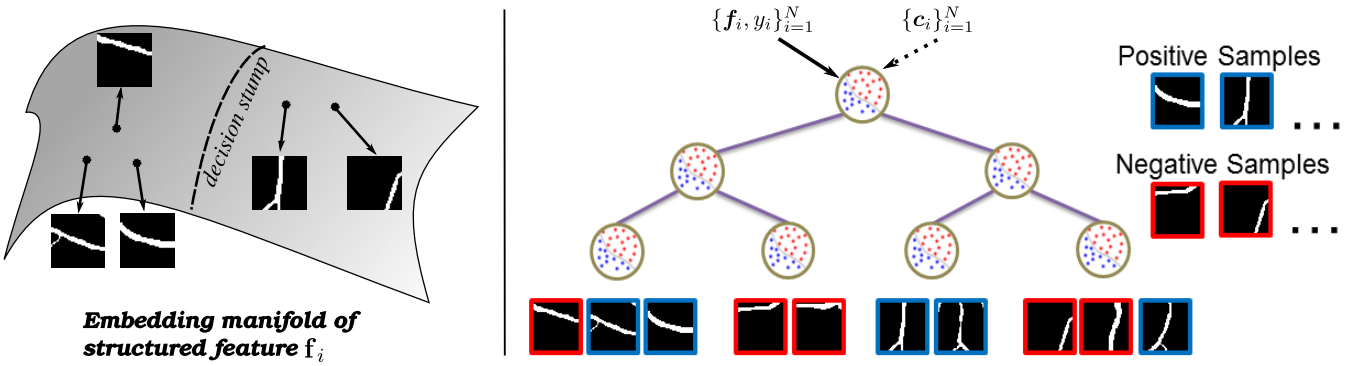


Fig. 3: Illustration of a single tree in the boosted tree classifier. Left panel illustrates why structured features are useful for partitioning patches containing similar filamentary structures into the same leaf node. Right panel shows the function of a single tree, which corresponds to a tree in boosted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

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 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 Figure 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 $l = 10$), 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 $\boldsymbol{p}_i$, next 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

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

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 [30] 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

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

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 e.g. shown in [31]. Moreover, 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

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

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 $R_v(\boldsymbol{v}):=\sum_{j=1}^{d-1}|\boldsymbol{v}_{j+1}-\boldsymbol{v}_j|$, 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 subpatches (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:

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

which is also illustrated in Fig. 1.

It is worth mentioning that 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-structure 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 for regression in leaf nodes, 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 figure 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 $\boldsymbol{p}_i$.

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 confidence, that is, keep only d_c components in each distance feature vector $\boldsymbol{c}_i$.

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.

III. Experiments

To examine the usefulness 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 by introducing the datasets and the evaluation metrics.

Datasets and Evaluation Metric: Three set of publicly available datasets are employed, each dedicates to one applications, as follows.

Two datasets are engaged for the task of 2D retinal vessel segmentation, that are DRIVE [32] and STARE [33]. DRIVE dataset [2] contains 40 fundus images of size 584×565, while STARE dataset [3] contains 20 fundus images of size 605×700. Both datasets have their own partitions of training and testing subsets, that is, in DRIVE 20 images are used training and 20 images are retained for testing purpose; Similarly, in STARE 10 are for training and the remaining 10 are for testing. These standard train and test splits are followed in this paper.

Then, to facilitate the analysis of 2D neuronal segmentation systems, The neuronal dataset [2] [4] is utilized. This dataset contains 112 images of in total 675 neurons. The image size is within the range of 512×572. The same train and test splits in [2] are adopted here in our experiments.

Finally, to demonstrate the application of our approach on 3D neuronal segmentation, the Gold166 dataset shared by BigNeuron initiative [12] is engaged here. The Gold166 dataset consists of 79 3D neuronal images along with the corresponding manually annotations. The sizes of the 3D images or image stacks vary from $511\times511\times597$ to $1024\times1024\times62$. 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

[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/`.

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

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[5] by filtering away the ones with questionable annotations. Among them, 17 images are randomly selected to form the training set, and the rest comprises of the testing set.

To evaluate the performance, we follow the common practice in e.g. [34], [2] to adopt the standard F1 measure (as $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$) and the precision-recall curve in our experiments. This metric helps to alleviate the issue caused by severe imbalance between the number of filamentary vs. background pixels in our scenarios, when comparing to metrics like accuracy and ROC curve. Note for the 3D neuronal (i.e. Gold166) dataset in particular, to account for the near-boundary annotation issue [6], a tolerance factor σ is introduced. Similar to that of [34], [35], [36], 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 examine different schemes (One such alternative scheme is presented in appendix) to counter the effect of near-boundary annotation issue, and found out all leads to similar results.

Our Approach and Variants: As discussed in Fig. 1, our pipeline contains the learning of two 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. Moreover, by ignoring the context distance features we may end up with a simplified version, referred to as **SF**.

A. Parameter Sensitivity Analysis

Our approach contains several parameters, including the number of weak learners M, tree depth of the weak learner d_t, 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 would like to evaluate its robustness w.r.t. change of parameter values. While experiments are conducted only on the DRIVE dataset, practical observation suggests similar trend is also obtained from other datasets, so what we have shown here is quite representative. The following default values are used here: $M = 500$, $d_t = 4$, $d_c = 3$, $\lambda = 0.1$ and $\rho = 1$. As indicated in Fig. 4, overall our approach is often robust w.r.t. varying parameter values, such as M, d_t, d_c, and $\rho = 1$. It is relatively more sensitive to the value of λ. In what follows we give more detailed analysis.

Number of weak learners: Although other boosted tree classifiers such as AdaBoost,and LogitBoost are also applicable here, we used the gradient boosting regression trees (GBRT) classifier here, due to the observations that GBRT achieves better segmentation results than other classifiers when combined with structured and context distance features as shown in Table **??**. To show how the performance changes with respect to tree number M, we evaluate our methods under different values of M and plot the corresponding mean F1 measure in Fig. 4(a). We observe that the performance consistently improves with the growing number of trees before reaching a saturation after $M = 500$, but the speed of improvement slows down as M increases. On the other hand, it is well known that increasing the tree number M would reduce the error in training at the potential risk of overfitting and higher computational burden. Therefore, we let $M = 500$ to balance the tradeoff.

Tree depth d_t and top d_c context distance features: For individual regression tree in the GBRT, the tree depth also matters. Deeper tree would improve the regression but at the same time increase the risk of overfitting and computational burden. We plot the mean F1 measure of our methods using different depths ($d_t = 3, 4, 5$) in Fig. 4(b). Similarly, we plot the mean F1 measure of our methods using different values of d_c in Fig. 4(c). From both figures, we observe similar trend as in Fig. 4(a). In other words, increasing the value of d_t or d_c would improve the performance, but the speed of improvement slows down and computational burden increases. Therefore, we let $d_t = 4$ and $d_c = 3$ in the rest of experiments.

λ and ρ in structured feature (SF) learning: We generate structured and context distance features using λ and ρ using values from set $\{0.01, 0.1, 1, 10, 100\}$. The results of our method with structured features only and our method with both structured and context distance features are shown in Fig. 4(d) and Fig. 4(e), respectively. We observe that the change of F1 measure with respect to different values of λ and ρ is less than 0.03, and the best performance is achieved at $\lambda = 10$, $\rho = 100$. When our method uses SF only, λ and ρ have larger influence on the F1 measure, but when our method uses both SF and context distance features, their influence can be negligible.

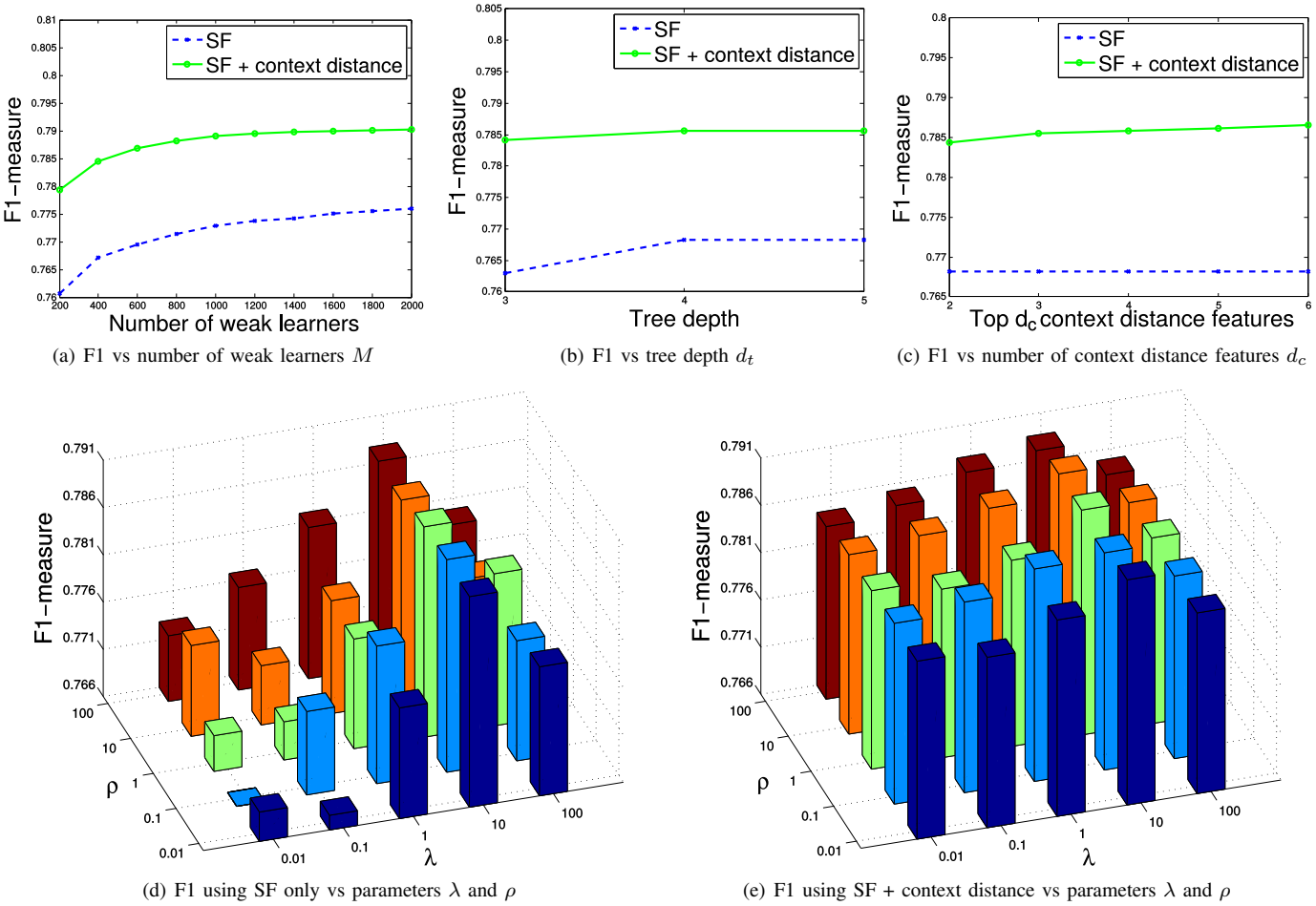
Besides investigating the effect of parameters, we also conducted an experiment to study the performance of different classifiers combined with different features. Specifically, we study the combinations of four tree structured classifiers: GBRT, AdaBoost, LogitBoost and random forests, with three different features: structured features, raw features and edge features. The raw feature is simply the image patch p_i surrounding the target pixel as defined in Section II-A. We also compare with a set of features, namely SE features, employed by the Structured Edge detection [19]. This SE features is originally developed for pedestrian detection [37] and then used in edge detection applications [19]. Specifically, there are three CIE-LUV colour, two magnitude and eight gradient channels as well as 13 pair-wise self similarity features. We refer [19] for further details.

TABLE I: Different feature and classifier selection with and without context distance features.

	GBRT	adaboost	logistic	random forest
structured feature	**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 feature	73.52 / 71.31	63.83 / 59.49	72.63 / 71.31	62.73 / 58.13

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

[6]As shown in Fig. 10(a) & 10(f), human annotation is often too conservative to envelop in the actual 3D filamentary point clouds. In other words, the groundtruth 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 are kept unchanged at the default values.

Besides investigating the effect of parameters, we also conducted an experiment to study the performance of different classifiers combined with different features. Specifically, we study the combinations of four tree structured classifiers: GBRT, AdaBoost, LogitBoost and random forests, with three different features: structured features, raw image and edge features [**?**]

Either with or without context distance feature, our structured features outperform the alternatives under all of four classifiers. This is due to the fact that our structured features could capture both discriminate and topology structure information. What is more, it also shows that the context distance features based on structured feature consistently improve the segmentation accuracy under the tree structure classifiers. We find that the boosting classifiers, especially the gradient boost and LogitBoost boost give the better performance through the iterative learning procedure. Therefore, throughout the experiment, we use the gradient boost as our classifier.

It is noted that though raw features achieve the comparable performance under gradient boost and logistic boost classifiers, its combination with the context distance feature fails to further improve the performance because raw features could endows the topology structure information to its context distance feature as what structured features have done. What is more, tableI shows that structured features significantly outperform the raw features on STARE and 2D neuronal dataset.

B. 2D Retinal and Neuronal Segmentation

Comparison Methods

In the experiment, our methods are compared against a wide range of state-of-the-art 2D segmentors covering both supervised and unsupervised ones. Specifically, we compare against the following methods: (1) Kernel Boost [8] utilizes Gradient-Boost to learn convolutional features from data. (2) Optimally Oriented Flux (OOF) [6] use manual filters for delineating tubular structures.(3) IUWT [5], based on isotropic undecimated wavelet transform, segment the 2D image in a unsupervised manner. (4) Eigen [4] is a multiscale Hessian-based unsupervised method for 2D segmentation. (5) T2T [38] is a supervised 2D segmentation system that integrates the pixel classification, medial sub-tree generation and global tree linking.

2D Results

In tableII, we gives the statistics of the performance on 2D segmentation algorithms. We also present the precision recall curves for individual dataset in Figure**??**. The representative

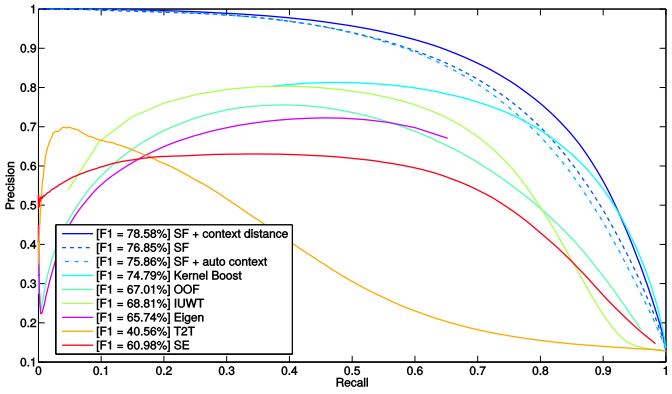


Fig. 5: Precision Recall Curve on DRIVE dataset

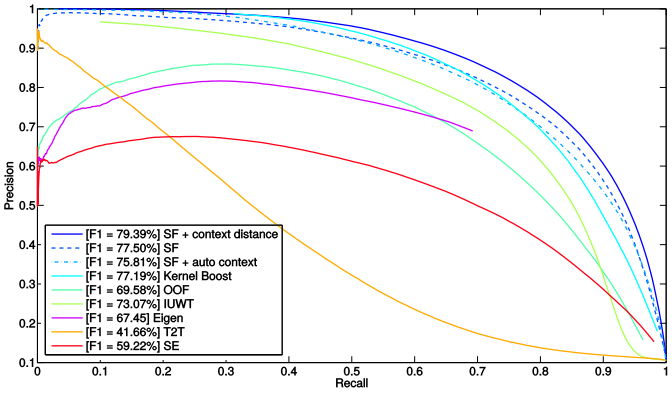


Fig. 6: Precision Recall Curve on STARE dataset

samples are given in Fig8 which shows the raw image (a), groundtruth (b) the posterior probability estimated by our SF + context variant (c), the error image under the optimal threshold of our SF + context variant (d), SF (e) and two most competing methods Kernel Boost (f) and Structured Edge(SE) .

Note that to demonstrate the difference between our context distance feature and the existing auto context feature [36], we replace the context distance features of our SF + context distance variant with the auto context features and name it as **SF + auto context**. More specifically, we refer to the original implementation of [36] that uses classification score of previous iterations as the additional input features in next iteration.

In terms of F1 measure, our method SF + context distance that augmented by context distance features have outperformed all state-of-the-art methods on every dataset. The SF + context distance achieve an average of 1% or 2% higher F1 measure than Kernel Boost method with most comparable accuracy. As shown in tableII, the context distance features have consistently improved the SF variant by 2% F1 score. What is more, our method solely relying on the structured features is able to outperform the alternative methods on both Retinal Dataset. As discussed in Section ?, our context distance features is similar to the widely used auto context features. However this auto context features actually lower the segmentation accuracy.

Fig5 compares the performance on DRIVE dataset. Our SF context variant achieves the best performance in the recall region ranging $[0.6, 0.8]$ which interests most practical application. Kernel Boost method shows slightly better precision when the threshold is too low to cover almost of the all of the details. Representative samples in Fig8 show that, our method is able to suppress the false alarms while securing recovery reasonable amount of vessels.

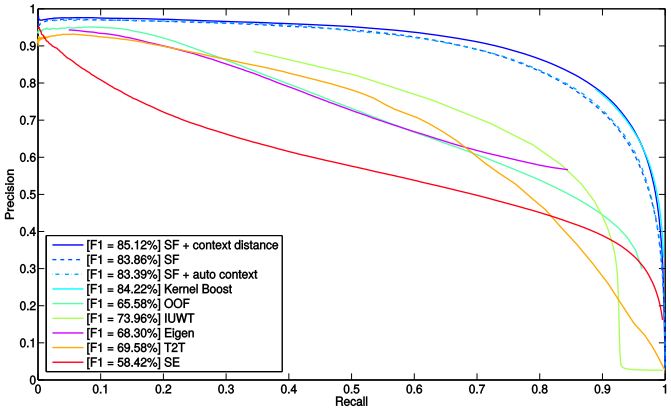


Fig. 7: Precision Recall Curve on NeuB1 dataset

Fig6 illustrates the superior performance of our SF + context distance variant over the other methods on STARE dataset. It is noted that, in this dataset, our variant only using the SF is effective in high recall region even compared to the most competing Kernel Boost Method. This is also illustrated in Fig8 that both of our variants successfully avoid the ambiguous vessels while recover equivalent amount of detailed vessels.

The 2D Neuronal Dataset, as depicted in Fig8, poses less challenge in the high recall region than the retinal dataset because the neurons dyed in fluorenscent protain exhibit a clear and well defined boundary. As the result, we find a almost same performance between the our SF + context distance variant and the kernel boost method when the precision is less than 0.8. However, as shown in Fig8, all of our variants methods are able to ignore the out-of-focus neurons in backgrounds while the Kernel Boost method could not tell them apart from the high quality neurons. Therefore, the Kernel Boost fails to get a precision higher than 0.8.

Comparison with Neural Network Approach

Deep learning approaches have become the state-of-the-art in image segmentation. In this part, we compare our method with N4-Fields [22], a deep learning method combining convolutional neural networks with nearest neighbor search. Since the authors of N4-Fields did not release their implementation, and only provided results of N4-Fields on the DRIVE dataset [7], we only do the comparison on the same dataset. We noticed that the results of N4-Fields were obtained on a eroded DRIVE dataset, where an eroded mask is used as indicated by the blue mask in Fig. 9(a). For the fairness of comparison, we applied our method on the same eroded dataset. The comparison results are shown in Fig. 9(b).

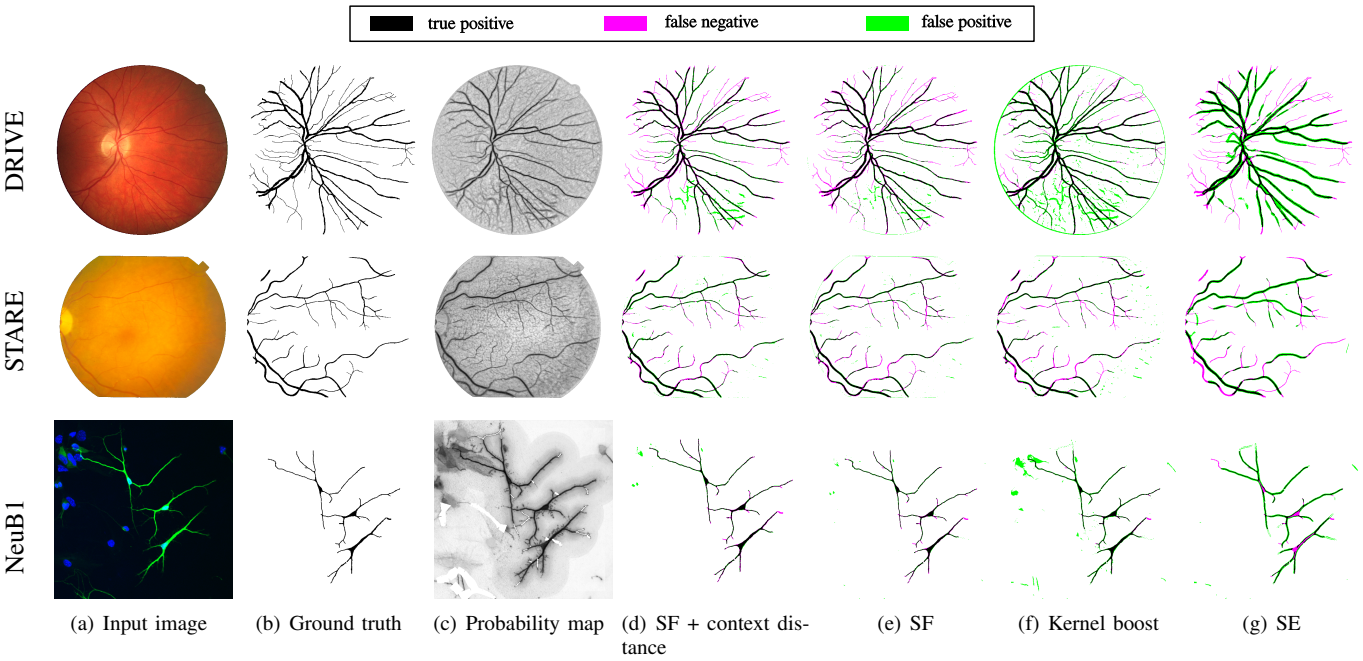
C. 3D Segmentation

Comparison Methods

[7] http://sites.skoltech.ru/compvision/projects/n4/drive.html

TABLE II: The Evaluation of the 2D Segmentation using F1 measure (%)

	SF	SF + context	SF + auto context	Kernel Boost [8]	OOF [6]	IUWT [5]	Eigen [4]	T2T [38]	SE [19]
DRIVE	76.82 ± 2.20	**78.56** ± 2.13	75.87 ± 2.45	74.17 ± 2.67	66.27 ± 3.11	68.25 ± 3.31	65.19 ± 4.85	40.30 ± 2.25	60.96 ± 2.74
STARE	77.48 ± 6.21	**79.37** ± 5.73	75.81 ± 6.92	77.19 ± 6.34	69.55 ± 6.40	72.97 ± 5.61	67.39 ± 4.48	41.65 ± 4.80	59.21 ± 5.74
2D Neurons	83.83 ± 3.67	**85.08** ± 3.65	83.36 ± 4.00	84.22 ± 3.79	65.58 ± 5.28	73.96 ± 4.37	68.30 ± 4.49	69.58 ± 4.43	58.42 ± 6.32



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

Fig. 8: Exemplar results on segmenting 2D retinal and neuronal images. (a): Input images; (b): Groundtruth; (c): Probability maps of our approach (SF + context variant); (d & e) Results of our approach (SF + context & SF only). (e) error image of Kernel Boost. Here green denotes false alarm and the magenta denotes the missing error.

When evaluating the 3D segmentation, we compare the three variants of our methods with two sets of state-of-the-art method. At first, we compare with the following three segmentation methods: (1) Regression Tubularity [36], which could be viewed as an extension of Kernel Boost [8], formulates the linear structure centreline detection in terms of a regression problem. (2) Adaptive Enhancement [39] is a Hessian based method dedicated to 3D neuron segmentation by detecting salient features by adaptive context windows. (3) Gray-Weighted Image Distance Transform or GWDT [40] is a 3D neuron segmentation method based on a region-growing scheme. These three methods are standard segmentations methods which produce a probability map, thus allowing us to obtain the F1 measure and the precision recall curve.

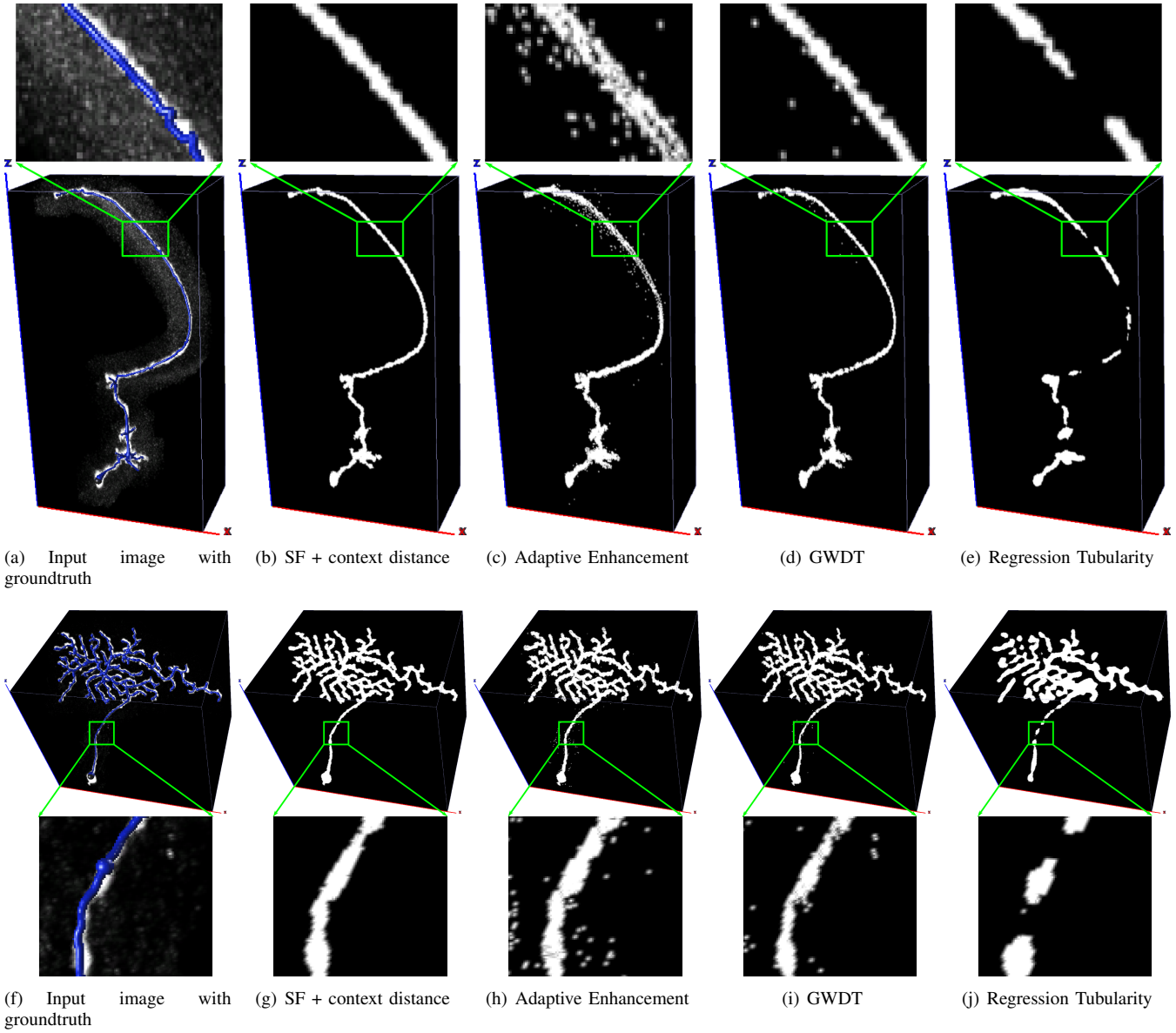
In addition, we find the neuron tracing algorithm, which aims to reconstruct the neuronal structure from the raw image, could be also viewed as a special case of segmentation. However there are two slightly different between segmentation and tracing in this paper's context. One is that we can only get a binary segmentation instead of the probability map, making our evaluation limited in F1 measure. Another issue worth notice is that these tracing algorithm focuses on recovering the path of neuronal processes rather than determine their exact boundary. With the courtesy of Big Neuron Challenge, we conduct a through comparison with all of the 11 participant tracing algorithms [8] which are published by the time of submission. These 11 tracing algorithms are the latest ones under the parameters optimised by their contributors.

3D Results

Figure10 shows the representatives samples of BigNeuron Gold166 dataset. We show the input image attached by the groundtruth annotation in blue and the binary segmentation results of the optimal threshold.

We also present the precision recall curve in Fig11, which shows our method achieves a higher precision over the most competing method, GWDT. As depicted in Fig10, our method manages retain much of the neuron without introducing the false alarm observed in the result of GWDT and Adaptive Enhancement. It is observed that the Regression Tubularity is able to achieves the highest precision at the cost of missing much of the neurons. On reason is that this approach is particularly designed to produce the maximal response at the centreline of the curvilinear object, thus making less sensitive to the whole body. It is also necessary to point out that the public released version that we used for comparison has been simplified for the sake of computational burden and the memory issue. We are looking forward to compare with the full version of Regression Tubularity [36] in the future study.

[8] https://github.com/BigNeuron/BigNeuron-Wiki/wiki/Neuron-Reconstruction-Algorithms



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

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

Fig. 10: Exemplar segmentation results on the Gold166 dataset [12]. (a&f): Input images with groundtruth in blue; (b&g): Results of our SF + context distance; (c&h) Results of Adaptive Enhancement [39]; (d&i) Results of GWDT [40]; (e&j) Results of Regression Tubularity [36].

Table III quantitatively compares the 3D segmentation methods on BigNeuron Gold166 showing that three variant methods have significantly outperformed the alternative approaches by at least 4% F1 measure. In 3D application, the context distance features are still able to boost the performance as they do in 2D application.

We also compare with the latest tracing algorithms that are currently participating the BigNeuron Challenge [12] in tableIV.

IV. Conclusion

We present a supervised 2D and 3D filament method which relies on a novel structured features learning process which encodes the topology structure. These features enable the gradient boosting trees to enjoy the benefits of regression tree although it is trained as normal classifier. As a result, our method now could not only segment the filament also identify their structure label, thus allowing us to introduce a simple but effective context distance feature. In addition, based on the structured label, we introduce context distance features to boost the performance of gradient boost regression tree. In the experiments, we demonstrate our method outperforms the state of the art method on both retinal and 3D neuron segmentation.

Appendix

A. An Alternative Evaluation Metric in 3D Experiments

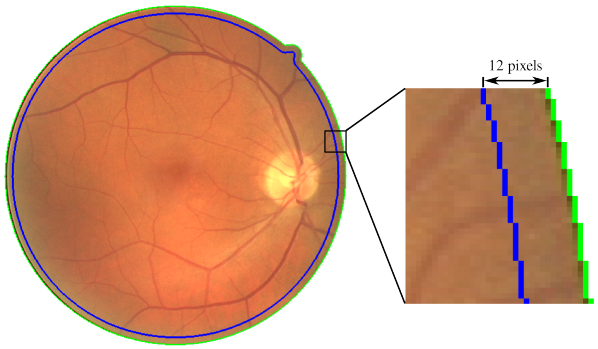
such evaluation metric may not be suitable for the BigNeuron Gold166 dataset. The reason is that, as shown in Fig. 10(a) & 10(f), human annotation is often too conservative to envelop the actual neuron. In other words, the groundtruth is often

TABLE III: Comparison of 3D neuron segmentation methods on BigNeuron Gold166 dataset [12] using F1 measure (%) with tolerance $\sigma = 3$

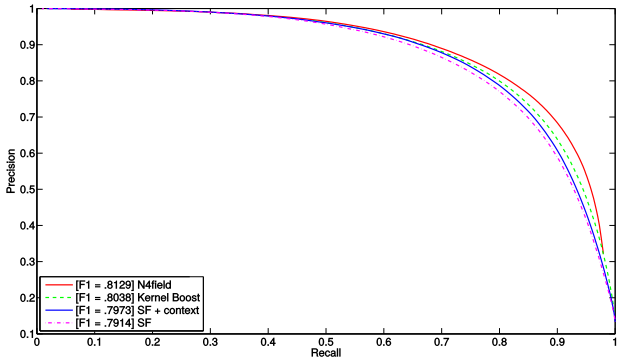
SF	SF + context distance	SF + auto context	Adaptive Enhancement [39]	GWDT [40]	Regression Tubularity [36]
79.38 ± 9.75	**79.89** ± 9.33	80.09 ± 9.18	$58.53pm14.27$	75.77 ± 10.52	65.75 ± 12.48

TABLE IV: Comparison with neuron tracing methods on the Gold166 dataset [12] using F1 measure (%) with tolerance $\sigma = 3$

SF + context distance	APP1 [41]	APP2 [40]	LCMBoost [9]	MOST [42]	nctuTW [43]	
86.60	56.01 ± 20.15	33.63 ± 13.48	31.46 ± 13.64	30.25 ± 14.21	24.45 ± 17.52	
NeuroGPSTree [44]	neuTube [45]	Rivulet [46]	SimpleTracing [47]	SmartTracing [48]	Snake [49]	TreMap [50]
51.85 ± 10.96	60.93 ± 7.14	55.89 ± 16.38	38.21 ± 21.33	67.25 ± 17.53	52.83 ± 18.91	52.30 ± 13.40



(a) Standard mask (green) and eroded mask (blue)



(b) Precision-recall curves

Fig. 9: Segmentation results on the eroded DRIVE dataset.

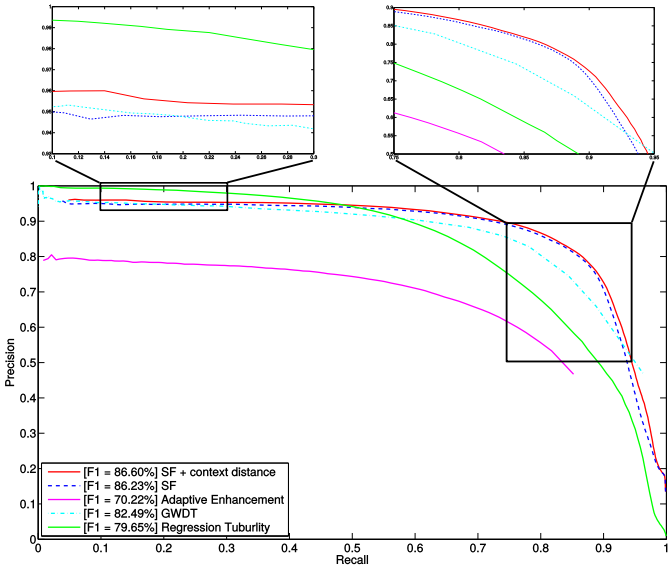


Fig. 11: Precision Recall Curve on BigNeuron dataset

smaller than the size it should be, and is not sufficiently accurate in terms of segmentation. To remedy this problem, when evaluating the 3D segmentation, we introduce a tolerance factor σ similar to [34], [35], [36], to perform the evaluation. Specifically, when pixel-wisely evaluating the 3D segmentation, we assign each pixel a weight w according to its shortest distance d_{gt} to the groundtruth (The pixel inside the groundtruth is considered as $d_{gt} = 0$) as

$$w = \begin{cases} (d_{gt})/\sigma & \text{if } 0 < d_{gt} < \sigma \\ 1 & \text{otherwise} \end{cases}$$

To remedy this problem, when evaluating the 3D segmentation, we introduce a tolerance factor σ similar to [34], [35], [36], to perform the evaluation. Specifically, when pixel-wisely evaluating the 3D segmentation, we assign each pixel a weight w according to its shortest distance d_{gt} to the groundtruth (The pixel inside the groundtruth is considered as $d_{gt} = 0$) as

$$w = \begin{cases} (d_{gt})/\sigma & \text{if } 0 < d_{gt} < \sigma \\ 1 & \text{otherwise} \end{cases}$$

As shown in Fig12, the pixels close to the groundtruth surface have less influence on evaluation due to the inaccurate in this region. The parameter $\sigma = 3$ is fixed throughout the experiments.

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 *MICCAI*, 1998.

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

[6] M. Law and A. Chung, "Three dimensional curvilinear structure detection using optimally oriented flux," in *ECCV*, 2008.

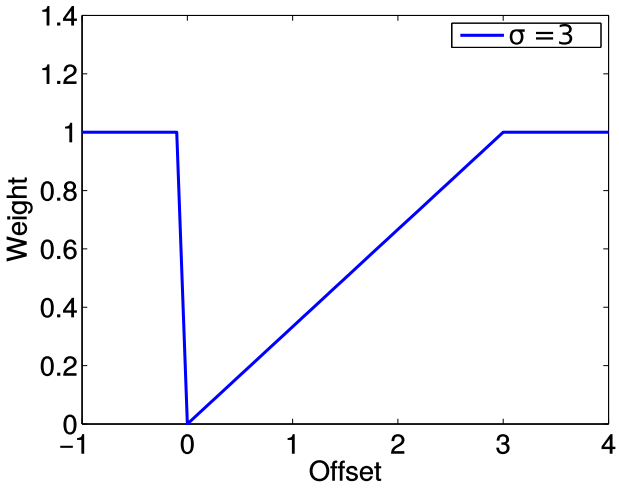


Fig. 12: The weight w with respect to the d_{gt} when $\sigma = 3$

[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 *MICCAI*, 2013.

[9] L. Gu and L. Cheng, "Learning to boost filamentary structure segmentation," in *ICCV*, 2015.

[10] 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.

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

[12] H. Peng, E. Meijering, and G. Ascoli, "From DIADEM to BigNeuron," *Neuroinformatics*, pp. 1–2, 2015.

[13] O. Cula and K. Dana, "Compact representation of bidirectional texture functions," in *CVPR*, 2001.

[14] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, 2003.

[15] S. Savarese, J. Winn, and A. Criminisi, "Discriminative object class models of appearance and shape by correlatons," in *CVPR*, 2006.

[16] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky, "Describing visual scenes using transformed objects and parts," *IJCV*, 2008.

[17] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.

[18] P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling," in *ICCV*, 2011.

[19] 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.

[20] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE TPAMI*, vol. 32, no. 10, pp. 1744–1757, 2010.

[21] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[22] Y. Ganin and V. Lempitsky, "N4-Fields: Neural network nearest neighbor fields for image transforms," in *ACCV*, 2014, pp. 536–51.

[23] 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.

[24] L. Hyafil and R. Rivest, "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[25] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *ICML*, 1996.

[26] ——, "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.

[27] 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.

[28] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *NIPS*, 2000.

[29] Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *ICCV*, 2005.

[30] 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*, pp. 91–108, 2005.

[31] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization," in *NIPS*, 2010.

[32] 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.

[33] 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.

[34] 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 *ICCV*, vol. 2, 2001, pp. 416–423.

[35] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, J. Langford and J. Pineau, Eds. New York, NY, USA: ACM, 2012, pp. 567–574.

[36] A. Sironi, E. Turetken, V. Lepetit, and P. Fua, "Multiscale centerline detection," in *IEEE Trans. PAMI*, 2016.

[37] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.

[38] S. Basu, A. Aksel, B. Condron, and S. Acton, "Tree2Tree: Neuron segmentation for generation of neuronal morphology," in *ISBI*, 2010.

[39] 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.

[40] 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.

[41] H. Peng, F. Long, and G. Myers, "Automatic 3D neuron tracing using all-path pruning," *Bioinformatics*, vol. 27, no. 13, pp. 239–247, 2011.

[42] J. Wu, Y. He, Z. Yang, C. Guo, Q. Luo, W. Zhou, S. Chen, A. Li, B. Xiong, T. Jiang, and H. Gong, "3D BrainCV: Simultaneous visualization and analysis of cells and capillaries in a whole mouse brain with one-micron voxel resolution," *NeuroImage*, vol. 87, pp. 199 – 208, 2014.

[43] P. Lee, C. Chuang, A. Chiang, and Y. Ching, "High-throughput computer method for 3D neuronal structure reconstruction from the image stack of the drosophila brain and its applications," *PLoS Comput Biol*, vol. 8, no. 9, pp. 1–12, 2012.

[44] T. Quan, H. Zhou, J. Li, S. Li, A. Li, Y. Li, X. Lv, Q. Luo, H. Gong, and S. Zeng, "NeuroGPS-Tree: automatic reconstruction of large-scale neuronal populations with dense neurites," *Nature Methods*, no. 13, pp. 51–54, 2016.

[45] T. Zhao, J. Xie, F. Amat, N. Clack, P. Ahammad, H. Peng, F. Long, and E. Myers, "Automated reconstruction of neuronal morphology based on local geometrical and global structural models," *Neuroinformatics*, vol. 9, no. 2, pp. 247–261, 2011.

[46] D. Zhang, S. Liu, S. Liu, D. Feng, H. Peng, and W. Cai, "Sub-voxel reconstruction of 3D neuron morphology using RIVULET back-tracking," *Neuroinformatics*, vol. 9, no. 2, pp. 181–191, 2011.

[47] J. Yang, P. T. Gonzalez-Bellido, and H. Peng, "A distance-field based automatic neuron tracing method," *BMC Bioinformatics*, vol. 14, no. 1, pp. 1–11, 2013.

[48] H. Chen, H. Xiao, T. Liu, and H. Peng, "Smarttracing: self-learning-based neuron reconstruction," *Brain Informatics*, vol. 2, no. 3, pp. 135–144, 2015.

[49] A. Narayanaswamy, Y. Wang, and B. Roysam, "3-D image pre-processing algorithms for improved automated tracing of neuronal arbors," *Neuroinformatics*, vol. 9, no. 2, pp. 219–231, 2011.

[50] Z. Zhou, X. Liu, B. Long, and H. Peng, "TReMAP: Automatic 3D neuron reconstruction based on tracing, reverse mapping and assembling of 2D projections," *Neuroinformatics*, vol. 14, no. 1, pp. 41–50, 2015.