Multiscale Centerline Detection

Amos Sironi, Engin Türetken, Vincent Lepetit, and Pascal Fua, *IEEE Fellow*

Abstract—Finding the centerline and estimating the radius of linear structures is a critical first step in many applications, ranging from road delineation in 2D aerial images to modeling blood vessels, lung bronchi, and dendritic arbors in 3D biomedical image stacks. Existing techniques rely either on filters designed to respond to ideal cylindrical structures or on classification techniques. The former tend to become unreliable when the linear structures are very irregular while the latter often has difficulties distinguishing centerline locations from neighboring ones, thus losing accuracy. We solve this problem by reformulating centerline detection in terms of a *regression* problem. We first train regressors to return the distances to the closest centerline in scale-space, and we apply them to the input images or volumes. The centerlines and the corresponding scale then correspond to the regressors local maxima, which can be easily identified. We show that our method outperforms state-of-the-art techniques for various 2D and 3D datasets. Moreover, our approach is very generic and also performs well on contour detection. We show an improvement above recent contour detection algorithms on the BSDS500 dataset.

1 Introduction

Linear structures appear in many contexts and at many scales. They can be axons and dendrites in the brain, blood vessels, roads, rivers or cracks in buildings, among others. As a result, their study is required in many fields such as neuroscience, biology, and cartography. Many tracing and reconstruction algorithms start by finding the centerline and estimating the radius of the linear structures. Since this first step is critical, it had been addressed many times in the literature. Most existing techniques rely on filters designed to respond to locally cylindrical structures [15], [24], [27], [35], [44], [58], optimized for specific profiles [22], or learnt [5], [17], [43]. They compute a scale-dependent measure that, ideally, should be maximal at the centerline of linear structures when computed for the correct scale.

Among these approaches, the learning-based ones tend to outperform the hand-designed ones when the linear structures become very irregular and deviate from the idealized models on which their design is based. Some works only aim at segmenting the linear structures from the background [5], and it is not clear how to reliably extract the centerlines from the segmentation. Others focus on the centerlines, but they typically rely on classification and this results in poor localization accuracy. As shown in Fig. 1, this is because it is hard for the classifier to distinguish points on the centerline itself from those immediately next to it.

In this paper, we show that this problem can be solved by reformulating centerline detection and radius estimation in terms of a regression problem. More precisely, we train several regressors to return distances to the closest centerline in scale-space, each regressor being trained for a specific scale. In this way, performing non-maximum suppression on their output yields both centerline locations and corresponding radii. We will show that, on very irregular structures, it outperforms the powerful OOF approach with and without anti-symmetry term [26], [27], which is widely acknowledged as one of the best among those relying on hand-designed filters; a very

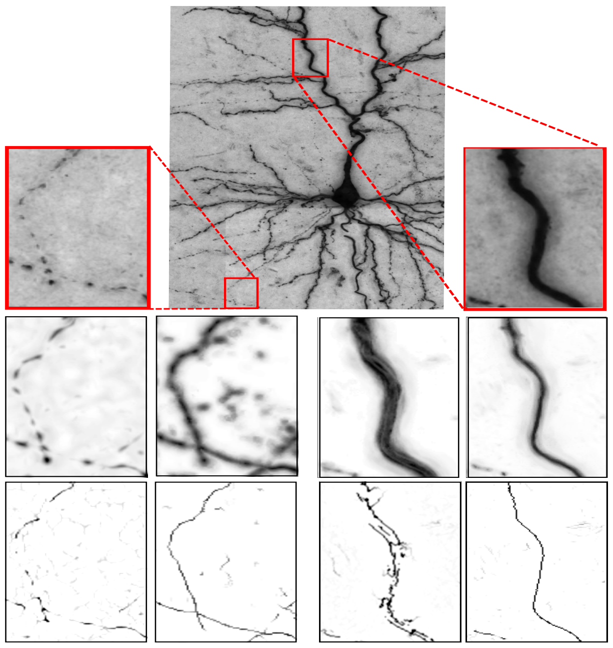


Fig. 1: Detecting dendrites in a 3D brightfield image stack. **Top row:** Minimal intensity projection with two enlarged details. **Middle row:** Comparison of the responses of our method against a recent model based approach [53] and a classification based one [5]. **Bottom row:** Centerlines detected after performing Non-Maximum Suppression on the response images. Model-based methods have trouble modeling highly irregular structures. Classification-based approaches respond on the whole body of the tubular structure and do not guarantee maximal response at the centerline. Our method combines robustness against image artifacts and accurate centerline localization.

- *A. Sironi, E. Türetken and P. Fua are with the Computer Vision Laboratory, IC Faculty, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland.*
E-mail: firstname.lastname@epfl.ch
- *V. Lepetit is with the Institute for Computer Graphics and Vision, Graz University of Technology, Graz 8010, Austria.*
E-mail: lepetit@icg.tugraz.at

This work was supported in part by EU ERC project MicroNano.

recent extension of it [53] designed to improve its performance on irregular structures; and a similarly recent classification-based method [5]. We will also evaluate the ability of our method to trace the linear structures. In particular, we will demonstrate that feeding our output as input to a complete tracing algorithm [54] instead of that of [53] increases final performance.

Finally, the idea of using regression instead of classification is generic and can be applied to other problems. For example, in contour detection, due to low resolution, blurring, and other image artifacts, the exact boundary location is often hard to find. Training a classifier to separate boundary points from others typically produce multiple responses on the boundaries and poor localization accuracy. We will show that applying our approach to detect boundaries in natural images avoid these problems. A direct application of our method on the Berkeley BSDS500 benchmarked dataset [2] actually outperforms albeit by a small margin state-of-the-art algorithms.

We first introduced the idea of using regression to extract centerlines in [48]. Here we improve upon our previous work by introducing an additional refinement inspired by the Auto-Context algorithm [52], which was originally proposed for image segmentation. More precisely, we use the output of the original regressors as features to a layer of new ones. By iterating this process, we can progressively correct earlier mistakes by exploiting information across a widening portion of the image. In particular, this helps eliminate false detections on the background and fill gaps in the linear structures.

In the remainder of the paper, we first review related work in Section 2. Then, in Section 3 we describe our method. Finally, in Section 4 we present the results obtained on five challenging datasets and prove the superiority of our approach over the state-of-the-art.

2 Related Work

Centerline detection methods can be classified into two main categories, those that use hand-designed filters and those that learn them from training data. We briefly review both kinds below.

2.1 Hand-Designed Filters

Such filters also fall into two main categories. The first is made of Hessian-based approaches [14], [15], [35], [36], [43], [44] that combine the eigenvalues of the Hessian to estimate the probability that a pixel or voxel lies on a centerline. The main drawback of these approaches is that the required amount of Gaussian blur to compute the Hessian may result in confusion between adjacent structures, especially when they are thick.

This has led to the development of a second class of methods based on Optimally Oriented Flux (OOF) [26]. They rely on the second order derivatives of an N-dimensional ball and are less sensitive to the presence of adjacent structures. Moreover, the radius of the ball provides a reliable estimate of the tubular structure scale. Remaining difficulties, however, are that OOF can also respond strongly to edges as opposed to centerlines and that its performance degrades when the structures become very irregular. A number of schemes have been proposed to solve the first problem [1], [27], [39], [50], [58]. For example, in [27], an Oriented Flux Antisymmetric (OFA) term was added and has proved effective. There has been less work on improving OOF's performance on truly irregular structures, except for the very recent approach of [53] that attempts to maximize the image gradient flux along multiple radii in different directions instead of only one as in [26].

The method proposed in [61] can be seen as a mixture of these two classes. Hessian computation implicitly assumes an ellipsoidal model whereas in [61] the ellipsoid is explicitly fitted to the data. Because this is harder to do than fitting OOF balls, it is achieved by a learning a regression model from image data to ellipse parameters. However, this has only been demonstrated in a tracking context.

2.2 Learned Filters

Even if care is taken to add computational machinery to handle irregular structures [43], [53], the performance of hand-designed filters tends to suffer in severe cases such as the one depicted by Fig. 1. This is mostly because it is very difficult to explicitly model the great diversity of artifacts that may be present.

Some works therefore aim at segmenting linear structures in biomedical images [5], [17], [42], [59] or aerial ones [37], [57] by applying classification to label the pixels or voxels as belonging to the structure of interest or to the background. However, this is a problem simpler than the one we consider. It is not accurate to find the actual centerlines from the segmentation even with post-processing operations. In particular, there is no guarantee that the classifier responses will be maximal at the centers of the structures. By contrast, we recover the centerlines and the corresponding thickness of the linear structures to which they belong, and it is straightforward to generate a segmentation from this data.

Other techniques, such as [7], [20], [56], [60], aim at extracting the centerlines as we do, but still rely on binary classification to distinguish the image locations on centerlines from the rest. [7], [60] use Haar wavelets in conjunction with boosted trees to detect the centerlines of tubular structures at different scales. [20] uses spectral-structural features instead and SVMs to find road centerlines. In [56] co-occurrence features and the AdaBoost algorithm are used to detect the spinal column centerline.

These methods exhibit limited localization accuracy because points near the centerlines can easily be also classified as centerline points due to their similar appearance. As our experiments show, our approach based on regression rather than classification is more adapted to the problem at hand.

2.3 Contour Detection

Edge detection has been one of the most widely studied problem in Computer Vision. Early attempts at solving it were based on filters designed to respond to specific image intensity profiles [33] and the resulting algorithms [8], [40] are still in wide usage. However, attention has recently shifted to classification based methods [2], [11], [12], [29], [31], [41], [47].

For example, in [2] gradients on different image channels are fed to a logistic regression classifier to predict contours in natural images. In [41], SVMs are trained to predict contours from features computed using sparse coding. In [11], a boosting algorithm is used to predict the probability of a boundary while Structured Random Forests are used in [12], which result in an extremely efficient contour detector. Recently, [3] achieved state-of-the-art performance in contour detection by proposing a fast way to group multiscale segmentations of an image into object candidates. Finally, the approach of [47] relies on a cascade of classifiers at different resolutions to predict a boundary map. It is related to ours in the sense that we also use cascades but we will show that we outperform it because regression is more appropriate than classification in this context.

2.4 Deep Learning Architectures

As mentioned in the introduction, we use an iterative process for regression. This is related to many different works in Computer Vision. Typically, in such frameworks, non-linear transformations are applied sequentially to the input signal to obtain a high order representation of the input, capture more contextual information and improve classification rate at each iteration.

Among these methods, Deep Convolutional Networks [28] have recently become very popular thanks to their impressive results on many benchmark datasets [9], [10], [25]. Typically, a large Convolutional Network is trained using back-propagation to minimize the classification error. However, because of the huge number of parameters to be optimized, Deep Networks require extremely large amounts of labeled training data and computational power to reach state-of-the-art performance. When the input image is too large or only limited amounts of training data are available, which is the typical situation in the biomedical domain, their applicability and performance are reduced. By optimizing the features in an unsupervised way and by optimizing the successive regressors one after the other, our method is easier to train. Moreover it can efficiently process large input volumes.

As in our approach, cascades of classifiers [13], [19], [23], [46], [47], [51], [52] have been used by several authors to improve classification performance. In these architectures, each layer is composed by one or more classifiers that are trained with a set of features extracted from the output of the previous layer, the first layer being the raw image. The training is easier than CNNs since each classifier is treated separately.

In [52] for example, the auto-context algorithm is introduced to improve segmentation results by integrating image and contextual features. In [46] instead multiscale information is exploited to train a sequence of classifiers to detect membranes in Electron Microscopy images. Their model is improved in [47], where, thanks to a hierarchical architecture, Deep Convolutional Networks are outperformed on the ISBI 2012 Challenge [21].

In contrast with all these methods, which consider a classification problem, we consider a regression formulation and show that this is the right formalism to solve centerline and boundary detection problems.

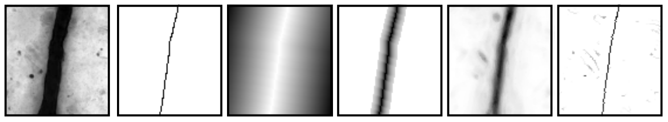


Fig. 2: Learning a regressor for centerline detection. (a) Raw image; (b) Ground truth centerline; (c) The distance transform to the centerline is used to discriminate points close to it; (d) The function we want to learn is maximal at the centerlines and it is thresholded to a constant value when the local window used to compute features does not contain any centerline points; (e) The function learned with our method; (f) Centerline detected after Non-Maxima Suppression (NMS) on function φ. In images from (b) to (f), white indicates lower values.

3 Method

Let $I(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^N$, be an N-dimensional image containing curvilinear structures of various radii. A classification-based approach to finding their centerlines involves learning a function $y(\cdot)$, such that

$$y(f(\mathbf{x}, I)) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is on a centerline,} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $f(\mathbf{x}, I)$ is a feature vector computed from a neighborhood surrounding $\mathbf{x}$ in image I. As discussed above, this is hard to do because points on the centerline itself for which $y(\cdot)$ should return 1 and their immediate neighbors for which it should return 0 look very similar. One way to solve this is to train $y(\cdot)$ to return 1 for all points within a given distance from the centerline. However, in practice, even if $y(\cdot)$ is allowed to return floating point values between zero and one, using for instance an SVM-style classifier, there is no guarantee that its value will be maximal at the centerline itself. This makes finding its accurate location, for example by using non-maximum suppression, problematic.

Our solution is to learn instead $y(\cdot)$ as a regressor whose values decrease monotonically as the distance of point $\mathbf{x}$ to the centerline increases. Then, as shown in Fig. 2, we can rely on simple non-maximum suppression to localize the centerlines. We will show in the next section that this solution is significantly more robust than both classification-based and filter-based methods.

Moreover, many automated and semi-automated tracing algorithms [6], [38], [55] rely on the extraction of local maxima from a tubularity measure as an initial step. Our method is designed to return a score with a well defined maximum along the centerlines and therefore can be used as input to improve the accuracy of these methods, as shown in Section 4.4.

In the remainder of this section, we first describe this process for structures whose scale is assumed to be known *a priori*. We then relax this constraint to handle structures of arbitrary scale and discuss the feature vectors we use as input to our regressors. Moreover, we will use the terms radius and scale interchangeably. The main notations used in the paper are summarized in Table 1.

TABLE 1: Main mathematical notations used in the paper.

Notation	Meaning
$I(\mathbf{x})$	Input image (resp. volume) at pixel (resp. voxel) $\mathbf{x}$
$f(\mathbf{x}, I)$	Feature vector computed on image I, at pixel $\mathbf{x}$
C	Set of centerline points for a given image
$y(f(\mathbf{x}, I))$	Ideal classifier output: $y(f(\mathbf{x}, I)) = 1$ iff $x \in C$
$\mathcal{D}_C(\mathbf{x})$	Euclidean distance transform of the set C at pixel $\mathbf{x}$
$d(\mathbf{x})$	Ideal regressor response. Exponential scaling of $\mathcal{D}_C$
$\varphi^{(m)}(f(\mathbf{x}, I))$	Actual regressor response for iterative regression, at iteration m
$g(\mathbf{x}, \varphi^{(m)})$	Feature vector for iterative regression, computed on score image $\varphi^{(m)}$ at pixel $\mathbf{x}$
$y(\cdot; r), \mathcal{D}_C(\cdot; r), d(\cdot; r), \varphi_r^{(m)}$	As above, but for centerlines corresponding to tubular structures of radius r
$\Phi(\mathbf{x})$	Multiscale regressor, used as final approximation

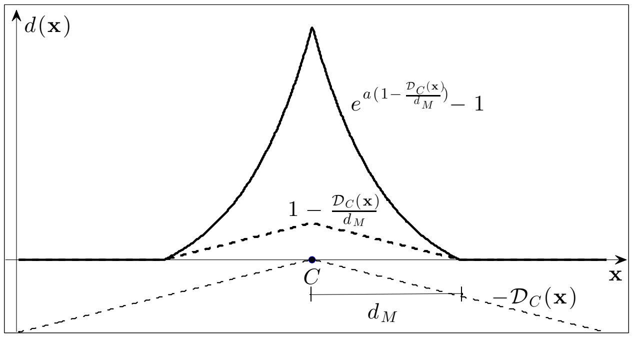


Fig. 3: The function d in the case of $\mathbf{x} \in \mathbb{R}$. If a centerline point is located in C, the function we want to learn is obtained from the distance transform $\mathcal{D}_C$, after thresholding and scaling. The vertical axis has been scaled for visualization purposes.

3.1 Learning a Regressor for Fixed Radius Structures

Let us momentarily assume that the linear structures have a known radius r. Let C be the set of centerline points and $\mathcal{D}_C$ the corresponding Euclidean distance transform, that is, $\mathcal{D}_C(\mathbf{x})$ is the metric distance from location $\mathbf{x}$ to the closest location in C.

Our goal is to learn a function $y(\cdot)$ such that $y(f(\mathbf{x}, I))$ is maximal for $\mathbf{x}$ on the centerline and whose value decreases monotonically as $\mathbf{x}$ moves away of it. The function $d(\mathbf{x}) = -\mathcal{D}_C(\mathbf{x})$ has this property, see Fig. 3. In theory, given training data, we could learn $y(\cdot)$ as a regressor that takes $f(\mathbf{x}, I)$ as input and returns $-\mathcal{D}_C(\mathbf{x})$ as output. However, in practice, we learn a different function for the two following reasons.

First, because our feature vectors $f(\mathbf{x}, I)$ are computed using local neighborhoods of size s, a regressor could only learn it for points that are close enough to the centerlines for their neighborhood to be affected by it. For this reason, it makes sense to threshold d when $\mathcal{D}_C$ is greater than a given value d_M, which is a function of the neighborhood size s. This yields the modified function

$$d(\mathbf{x}) = \begin{cases} 1 - \frac{\mathcal{D}_C(\mathbf{x})}{d_M} & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

which takes values between 0 and 1, see Fig. 3. In our implementation, we set $d_M = s/2$, which means that d is uniformly 0 for points whose corresponding neighborhood does not overlap the centerline.

Second, a regressor trained to associate to a feature vector $f(\mathbf{x}, I)$ the value of $d(\mathbf{x})$ can only do so approximately. As a result, there is therefore no guarantee that its maximum is exactly on the centerline. To increase robustness to noise, we have therefore found it effective to train our regressor to reproduce a distance function whose extremum is better defined. In our actual implementation, we take it to be

$$d(\mathbf{x}) = \begin{cases} e^{a(1-\frac{\mathcal{D}_C(\mathbf{x})}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where $a > 0$ is a constant that control the exponential decrease rate of d close to the centerline, see Fig. 3. In all our experiments, we set $a = 6$.

The regression method we use to learn function d is the GradientBoost algorithm [18]. It can be viewed as a generalization of the AdaBoost algorithm and it can efficiently approximate very complex functions.

Given training samples $\{(f_i, y_i)\}_i$, where $f_i = f(\mathbf{x}_i, I_i) \in \mathbb{R}^J$ is the feature vector corresponding to a point $\mathbf{x}_i$ in image I_i and $y_i = d(\mathbf{x}_i)$, GradientBoost approximates $y(\cdot)$ by a function of the form

$$\varphi(f(\mathbf{x}, I)) = \sum_{k=1}^{K} \alpha_k h_k(f(\mathbf{x}, I)) \,, \quad (4)$$

where $h_k : \mathbb{R}^J \to \mathbb{R}$ are weak learners and $\alpha_k \in \mathbb{R}$ are weights. Function φ is built iteratively, selecting one weak learner and its weight at each iteration, to minimize a loss function $\mathcal{L}$ of the form $\mathcal{L} = \sum_i L(d_i, \varphi(f_i))$, where $L(.)$ is the squared loss function $L(d_i, \varphi(f_i)) = (d_i - \varphi(f_i))^2$. We also experimented with the L1 and Huber loss functions and the results proved to be very similar.

As is usually done with GradientBoost, we use regression trees as weak learners since they achieve state-of-the-art performance in many applications [18]. Unless otherwise stated, in all our experiments we used $K = 250$ trees of depth 2. Fig. 2 shows the output of the learned function for a sample image. For simplicity, unless there are ambiguities, we will write $\varphi(\mathbf{x})$ instead of $\varphi(f(\mathbf{x}, I))$ and $h_k(\mathbf{x})$ instead of $h_k(f(\mathbf{x}, I))$.

3.2 Iterative Regression

Using $\varphi(\cdot)$ to predict function $y(\cdot)$, as explained above and as was done in our earlier work [48], may result in incorrect large values on the background or missed parts of the linear

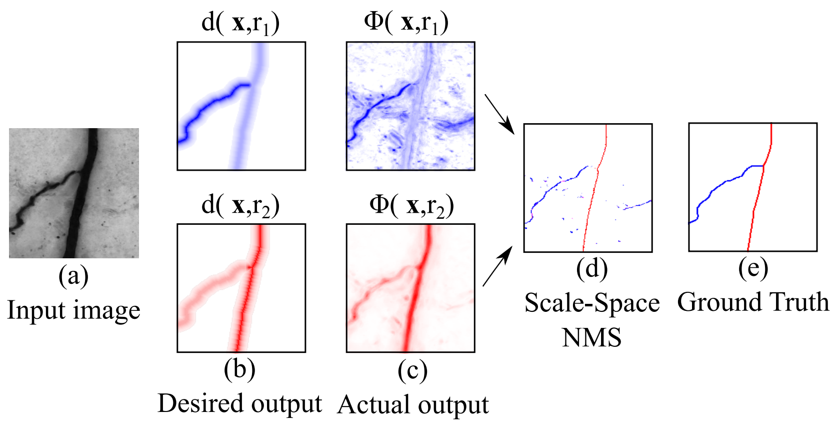


Fig. 4: Multiscale centerline detection. (a) Input image containing linear structures at different scales. We want to learn a function with local maxima at centerline points along the spatial and radial axes. (b, top): Values of d for the smaller radius r_1, (b, bottom): values for the larger radius r_2. (c) The learned multiscale approximation Φ for r_1 and r_2. (d) The centerlines and the radii are detected with non-maxima suppression in the scale-space. (e) Ground truth centerlines. Best viewed in color.

structures, since only local information is used for prediction purposes.

These mistakes can be avoided by including more contextual information in the algorithm, as is done in the so-called Auto-Context algorithms [47], [52] for classification purposes. To this end, we use the score map $\varphi(\mathbf{x})$ to extract a new set of features able to discriminate isolated responses on the background and to fill gaps in the detected structures. These new features are added to the original ones to train a new regressor. By iterating this process we obtain a sequence of regressors able to include more and more contextual information in the learning algorithm and to correct the mistakes done at the previous iterations.

More precisely, let $g(\mathbf{x},\varphi^{(0)})$ be the feature vector extracted from the score map $\varphi^{(0)}(\mathbf{x}) = \varphi(\mathbf{x})$ and let $\{(f_i, g_i, y_i)\}_i$ be the new training set, with $g_i = g(\mathbf{x}_i, \varphi_i^{(0)}) \in \mathbb{R}^{J'}$ and $\varphi_i^{(0)} = \varphi^{(0)}(f(\mathbf{x}_i, I_i))$. We apply again the Gradient Boost algorithm to learn a better approximation of the function $y(\cdot)$:

$$\varphi^{(1)}(f(\mathbf{x},I), g(\mathbf{x},\varphi^{(0)})) = \sum_{k=1}^{K} \alpha_k^{(1)} h_k^{(1)}(f(\mathbf{x},I), g(\mathbf{x}_i,\varphi_i^{(0)})) . \tag{5}$$

We iterate this process M times learning a series of regressors $\{\varphi^{(m)}(f(\mathbf{x},I), g(\mathbf{x},\varphi^{(m-1)}))\}_{m=0,\ldots,M}$. The final output $\varphi^{(M)}(\cdot)$ will be used as approximation of $y(\cdot)$. Again, for the sake of brevity, we will write $\varphi^{(m)}(\mathbf{x})$ instead of $\varphi^{(m)}(f(\mathbf{x},I), g(\mathbf{x},\varphi^{(m-1)}))$.

To prevent overfitting, which is a known weakness of Auto-context frameworks, we adopt several strategies. First, at the beginning of each auto-context iteration new pixel locations are sampled from the train images to build a new training set. Second, at each boosting iteration k we learn the weak learner h_k using only a random subset of the whole training set, as in Stochastic GradientBoost [16]. Finally, as discussed in Section 3.4, the features used to learn a weak learner are also subsampled at each boosting iteration. Fig. 6(c) shows the advantage of using iterating regression.

In practice, the method converges fast: The performance does not improve beyond the second iteration and we therefore set $M = 2$ in all our experiments.

3.3 Handling Structures of Arbitrary Radius

In the previous section, we focused on structures of known radius. In general, however, structures of many different radii are present. To generalize our approach to this multi-scale situation, we redefine the function d of Eq. (3) once again as

$$d(\mathbf{x};r) = \begin{cases} e^{a\cdot(1-\frac{\mathcal{D}_C(\mathbf{x};r)}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x};r) < d_M, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where C is now the set of centerline points and the corresponding radii. In other words, C now is a set of $(\mathbf{x};r)$ $(N+1)$-dimensional vectors and $\mathcal{D}_C(\mathbf{x};r)$ is the *scale-space* distance transform of C:

$$\mathcal{D}_C^2(\mathbf{x};r) = \min_{(\mathbf{x}',r')\in C} \|\mathbf{x}-\mathbf{x}'\|_2^2 + k(r-r')^2 \quad , \tag{7}$$

where k is used to weight the scale component differently from the space component. In practice k depends on the image resolution and the range of scales. In Section 4 we discuss the choice of k.

If we consider the maximum projection of $d(\mathbf{x};r)$ along the radial component, we obtain a function of $\mathbf{x}$, whose local maxima are the centerline points for all the values of r. Therefore, if we train a regressor to output the values of d, the problem of multiscale centerline detection is reduced to the problem of finding local maxima in the projected image, see Fig. 4. Moreover, function $d(\mathbf{x};r)$ is defined so that points in C are local maxima of d not only along the spatial dimensions, but also along the radial component, as shown in Fig. 4(b). Therefore, we can easily find the scale corresponding to a centerline point as the one that gives the maximal value for that point, Fig. 4(d).

We now want to learn a regressor $y(\cdot;r)$ that returns the values of this new d function. The simplest way would be to discretize the range of possible scales r into a finite set of scales and to use the fixed-radius method of Section 3.1 to learn one regressor φ_r for each scale in this set. This approach, however, decreases the number of training samples available to train each regressor, which in our experience severely impairs performance.

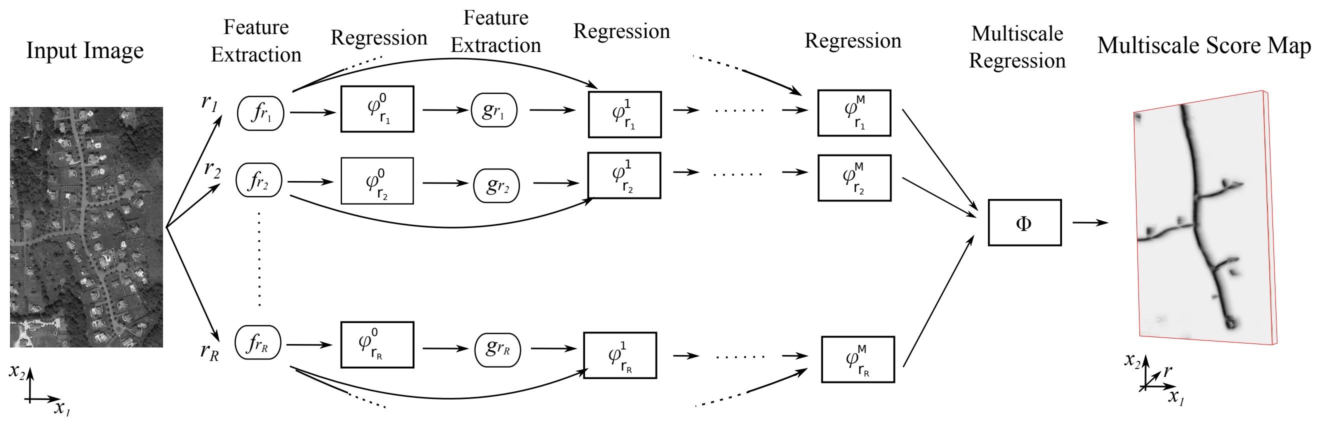
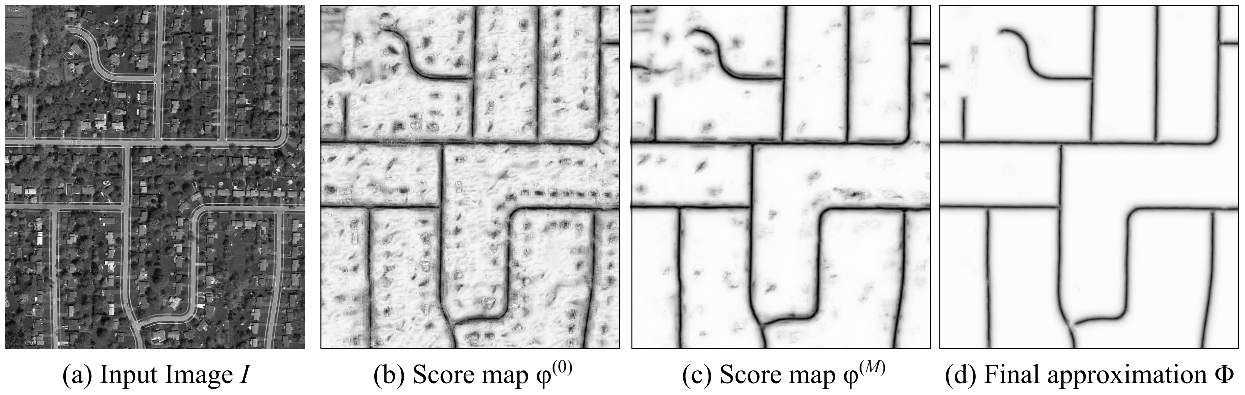


Fig. 5: For each value of radius r, the input image is convolved with a bank of filters to extract a set of image features. The features are used as input to regressors $\varphi_r^{(0)}$. The outputs of the regressors are then convolved with other filter banks to extract new features. These features are fed together with the image features to a second layer of regressors $\varphi_r^{(1)}$. This process is iterated M times. In the final step, the output of the regressors is fed to Φ, a multiscale regressor, that computes the final score map.



(a) Input Image I
(b) Score map $\varphi^{(0)}$
(c) Score map $\varphi^{(M)}$
(d) Final approximation Φ

Fig. 6: Improvement obtained by iterating regression. (a) Input image $I(\mathbf{x})$; (b) Score map $\varphi^{(0)}(\cdot)$ obtained for $M = 0$, which corresponds to our earlier work [48]; (c) Score map $\varphi^{(M)}(\cdot)$ with $M = 2$ obtained using our new approach as described in Section 3.2; (d) Score map $\Phi(\cdot)$ obtained by adding the multiscale learning step described in Section 3.3. Both the iterations and the last multiscale regressor help to remove false detections on the background and to obtain a better localization accuracy of the centerlines. For (b), (c), and (d) we show the maximum projection along the radial dimension for visualization purposes.

An alternative approach is to rely on scale-space theory [30] to train a single regressor φ_{r^0} for radius r^0. By properly scaling and normalizing the convolutional filters used to compute the feature vectors $f^{r_0}(\mathbf{x}, I)$, we can use φ_{r^0} to find the centerlines for all the other radii. The advantage of this approach is that we can exploit all training samples to train φ_{r^0} by rescaling them to have a radius equal to r^0. However, this assumes that the aspect of tubular structures is scale invariant. When this is not the case, the results are less accurate, especially for large differences between the actual radius of the structure and r^0.

We therefore adopt a hybrid approach. We learn a set of regressors $\{\varphi_{r_i}\}_{i=1}^R$ for a small set of regularly sampled radii. We then apply the scale-space approach for intermediate radii and use the closest r_i to the scale we want to predict. In Section 4 we discuss how these radii are selected. As in the single-scale case, and as summarized in Fig. 5, we use the Auto-Context strategy to improve the accuracy of our method and create a sequence $\{\varphi_{r_i}^{(m)}(\cdot)\}_{i,m}$ of scale-space regressors. Potentially all the outputs at every scale, obtained at iteration $m - 1$ could be used to train the regressors for the next iteration. However, this would increase the learning and test times considerably. We therefore adopt a different strategy and divide the learning process into simpler subproblems. At the first M iterations the regressors at different scales are learned independently: $\varphi_{r_{i_0}}^{(m)} = \varphi_{r_{i_0}}^{(m)}(f^{r_{i_0}}(\mathbf{x}, I), g(\mathbf{x}, \varphi_{r_{i_0}}^{(m-1)}))$. Then, as a final step, we take the score maps obtained at all scales $\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^R$ and use them to train a last multivariate regressor $\Phi(\cdot)$, where now $\Phi(\mathbf{x}) \in \mathbb{R}^R$.

We build the function $\Phi(\cdot)$ again with GradientBoost:

$$\Phi(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k h_k(\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^R) \, , \tag{8}$$

where now the weak learners $h_k(\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^R) \in \mathbb{R}^R$ return a vector of values, each component corresponding to a different scale. We use $\Phi(\cdot)$ as the final approximation of the scale-space function $d(\cdot, \cdot)$.

This last step imposes consistency and smoothness on the values returned by the previous regressors, which were trained independently. Fig. 6(d) shows the advantage of this last step on a sample road image.

3.4 Computing the Feature Vectors

We first discuss the image features we feed as input to the initial regressors of Section 3.1 then those we use to implement the Iterative regression of Section 3.2.

3.4.1 Image Features

There are many possible ways to compute the feature vectors $f(\mathbf{x}, I)$ of Eq. (1), such as Haar wavelets, steerable filters, or Gabor filters. Recent work [42] has shown that learning a set of convolutional filters via sparse coding techniques can produce expressive features that perform well on linear structures. For this reason, in our earlier work [48] we used

$$f(\mathbf{x}, I) = [(\mathbf{f}_1 * I)(\mathbf{x}), \ldots, (\mathbf{f}_J * I)(\mathbf{x})]^\top , \tag{9}$$

where the $\mathbf{f}_i$'s are convolutional filters learned in a unsupervised way by enforcing sparsity as in [42] from a set of training images, and applied to image I.

Here we exploit additional image information by also considering locations within a certain distance of location $\mathbf{x}$. This produces a much larger pool of possible features

$$f(\mathbf{x}, I) = \{(f_j * I)(\mathbf{x} + \rho)\}_{j,\rho} , \tag{10}$$

with $\rho \in \mathbb{R}^N$ and $\|\rho\| \leq L$ for some fixed $L > 0$. For example, for a 121-filter bank, setting $L = 13$ results in approximately $64,200$ possible features. We handle this potentially large number by considering at each boosting iteration only a random subset of all possible locations and convolutional features. This also has the added benefit to reduce overfitting.

In the case of 2D images, we used $J = 121$ filters. In the case of 3D volumes, the number of possible orientations of the tubular structures is significantly larger and therefore more filters should be used. We found it most effective to learn first a filter bank of $J = 121$ filters and then extend it by rotating the learned filters at different orientations, 14 in practice. To speed up the convolutions required to compute the descriptor in Eq. (9), we rely on the technique introduced in [49], which approximates the filters $\{\mathbf{f}_i\}$ with a set of separable ones.

3.4.2 Auto-Context Features

After iteration m, we extract a new set of features from the score map $\varphi^{(m)}(\cdot)$ that will be used in the next iteration. As for the image information, we use the method of [42] to learn a filter bank specifically trained to extract features from the score map.

Ideally we should learn a set of filters on the score maps at each auto-context iteration. In practice however, this would be prohibitively expensive. In biomedical imagery, the background is relatively uniform and the score maps produced by the regressors exhibit characteristics similar to those of the original images. We therefore use the same filters as for the image to extract features from the score maps. In other kinds of images, such as aerial or natural images, which contain objects other than the linear structures, we learn instead a bank of filters from the ground truth training images $d(\mathbf{x})$. This produces filters able to detect linear structures and junctions similar to those we would have learned on the score maps.

Again neighbor locations are considered to capture more context. Formally, the feature vector on the score image at the auto-context iteration m is given by $\{f_j * \varphi^{(m)}(\mathbf{x} + \rho)\}_{j,\rho}$. To keep the computational complexity under control, we again subsample the set of possible features at each boosting iteration.

In the case of multiscale detection, at the last iteration, the function Φ is learned from all the previously computed maps $\{\varphi_{r_i}^{(M)}\}_i$ and we do not use features extracted from the original image anymore. This compensates for the increased number of score map features. Moreover image features are not really needed here because the purpose of this last step is to enforce score consistency over the different scales.

3.5 Non-Maximum Suppression

Applying our method to an N-dimensional image, yields an $(N + 1)$-dimensional one, with N spatial dimensions and one scale dimension. Our method is designed to respond maximally at the centerlines in scale-space. To find these local maxima, we first compute a N-dimensional image by keeping for each location the maximum along the radii, and saving the radius corresponding to the maximum. We then perform a Canny-like non-maximum suppression by keeping only the locations that correspond to a local maximum along a line perpendicular to the local orientation, and within a neighborhood of width defined by the radius. We estimate the orientation using the eigenvectors of the oriented flux matrix [26], which we found to be more robust than using the Hessian matrix. Results from this non-maximum suppression step are shown in Fig. 7.

4 Results

In this section, we first introduce the datasets and the parameters used to test our centerline detection method. Then, we describe our evaluation methodology and we discuss our results. We then evaluate the improvement brought by our method when used in conjunction with automated tracing algorithms. Finally we apply our algorithm to the problem of boundary detection in natural images.

4.1 Datasets and Parameters

Our method depends on few parameters, namely: the radial weight k in Eq (7); the size s of the filters used to extract the features; the range of sampled scales and the number of trained regressors.

The range of scales sampled for the different datasets is automatically determined from the ground truth data and was always sampled uniformly. We optimized the other parameters by a cross validation procedure on small volumes. We tested our method on the 2D road images and 3D biological image stacks depicted by Fig. 7. More specifically we used the following datasets:

- **Aerial**: Aerial images of road networks. We used an extended version of the training set we used in [48]. It is composed of 13 images for training and 17 additional images for testing. We sampled 10 scales ranging from

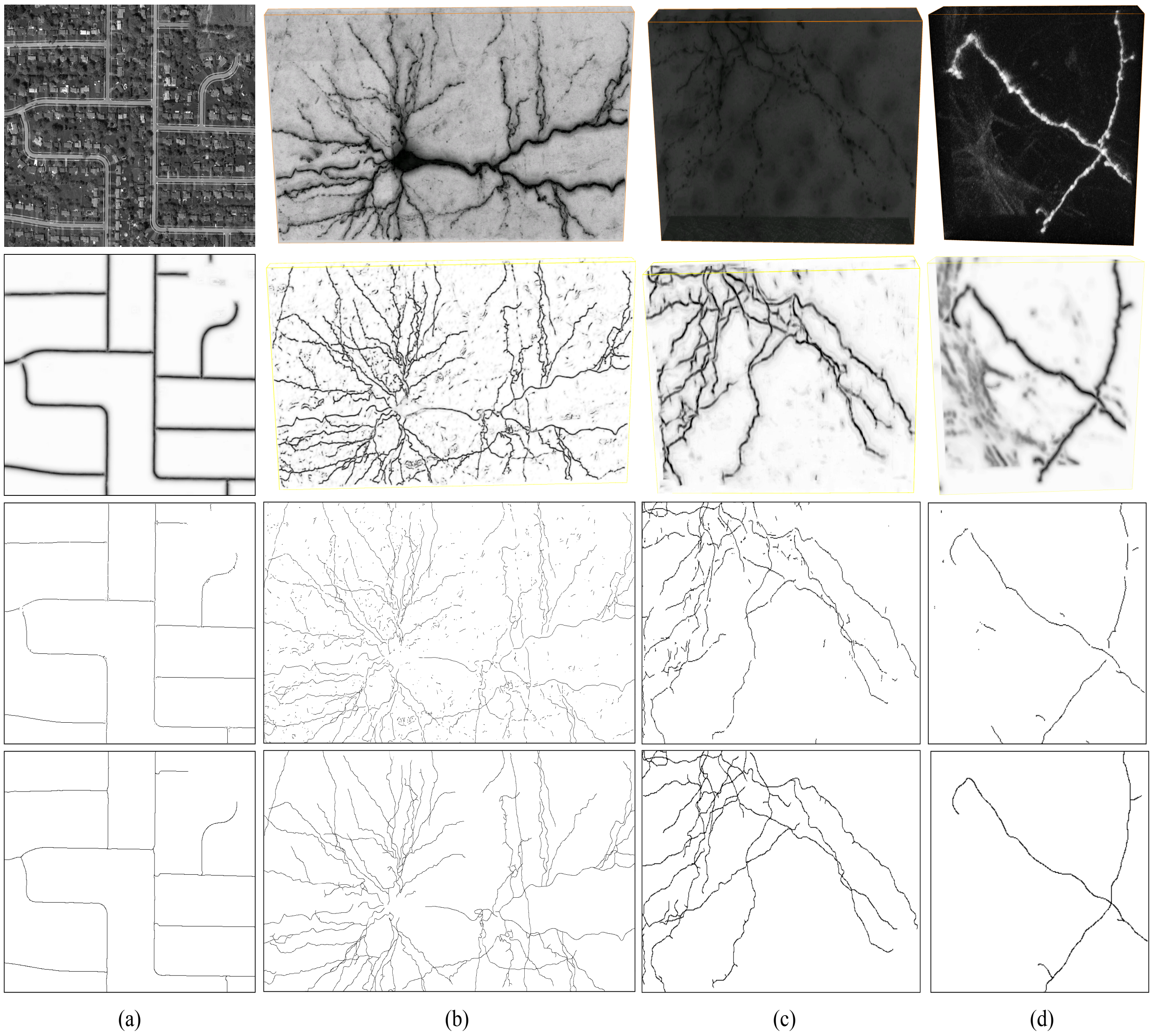


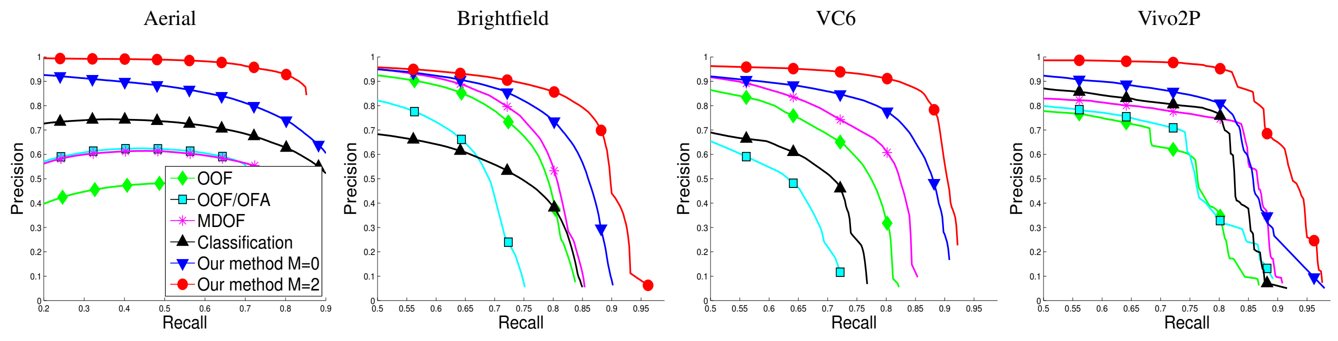
Fig. 7: Centerline Detection Results. (a) Aerial image. (b) Brightfield image stack. (c) VC6 image stack (d) *In vivo* two-photon image stack. In each case, we show from top to bottom the original image, the maximum projection along the radial component of our regressor's output, centerlines detected by thresholding after non-maximum suppression, and ground truth centerlines.

5 to 14. We trained 2 regressors at scales 6 and 9 and learned filters of size $s = 21$. We set k to 1.

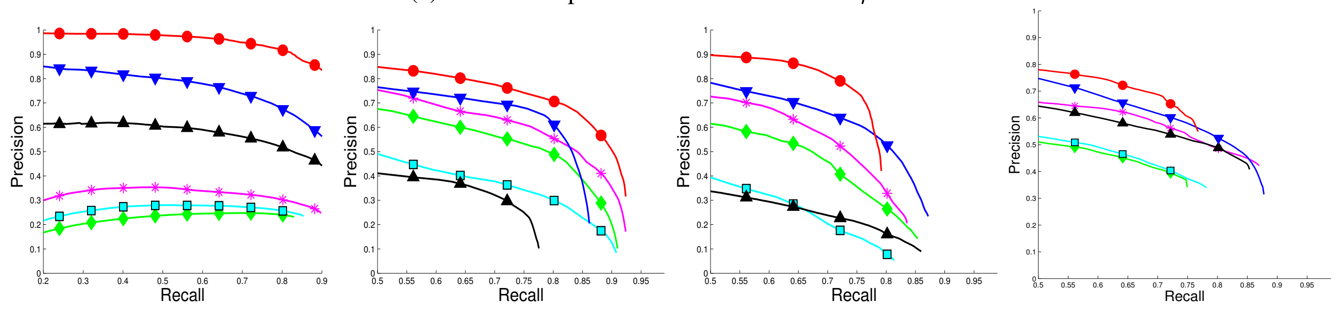
- **Brightfield**: A dataset of 3D image stacks acquired by brightfield microscopy from biocityne-dyed rat brains. We used 3 images for training and 2 for testing. We sample 12 scales corresponding to radii from 1 to 12 microns. We trained 2 regressors at scales 2 and 8. We learned filters of size $s = 21$ and used $k = 1$.
- **VC6**: Three dimensional brightfield micrographs of biocytin-labeled cat primary visual cortex layer 6 taken from the DIADEM challenge data [4]. We used 3 images for training and 2 for testing. We sampled 6 scales from 1 to 6, trained 3 regressors at scales 1, 3, and 5. We used $k = 7$ and $s = 11$.
- **Vivo2P**: Three dimensional *in vivo* two-photon images of a rat brain, capturing the evolution of neurons in the neocortex. We used 2 images for training and 3 sequences of 3 images for testing. We sampled 3 scales, 0.6, 0.7, and 0.8 microns. We trained one regressor at scale 0.7, using $k = 1$ and $s = 21$.

For training, at each auto-context iteration we randomly sampled $100\,000$ image locations within the distance d_M to the centerline and other $100\,000$ from points further than d_M to the centerline. During the boosting iterations half of the samples were randomly used to learn the weak learner. More samples are needed to train the the final regressor Φ, and we therefore used 10^6 samples for this purpose.

The size of the images ranges from $\sim 10^5$ pixels for the Aerial dataset to $\sim 10^8$ voxels for Brightfield. The running time in our Matlab implementation is of several hours for



(a) Centerline precision-recall curves for $\rho = 2$.



(b) Segmentation precision-recall curves for $\delta = 0.4$.

Fig. 8: Precision Recall curves. Our method outperforms the others on all the datasets we considered, both for centerline detection and joint centerline and radius estimation. Using iterative regression allows us to further improve the results.

training and from few minutes to few hours for testing.

4.2 Pixel-Wise Evaluation

We compare our approach against three of the most powerful model-based methods for centerline detection. Optimally Oriented Flux (OOF) [26], Oriented Flux with Oriented Flux Antisymmetry [27] (OOF+OFA), and Multidirectional Oriented Flux [53] (MDOF). Moreover, to prove the importance of our regression approach compared to classification, we also train a GradientBoost classifier to segment the centerlines from the rest of the images, thus emulating the approach of [5].

As usually done to evaluate methods extracting one-pixel-wide curves [34], [37], [53], we introduce a tolerance factor ρ to perform plot precision-recall (PR) curves analysis. A predicted centerline point is considered a true positive if it is at most ρ distant from a ground truth centerline point. We generate PR curves for all the methods for different value of ρ. The results for $\rho = 2$ are shown in Fig. 8(a) and show that our approach clearly outperforms all other datasets. Moreover, performing iterative regression further improve the performance of our method in all the studied cases, confirming the importance of using more contextual information to solve the problem.

We also evaluate the accuracy of the radii we estimate. Again we follow the same evaluation methodology of [53]. We start by thresholding the image after non-maximum suppression at different values. Then, for each point in the thresholded image, we construct a sphere using the corresponding estimated radius. In this way we obtain for every threshold value a full segmentation of the tubular structures, which we can compare to the ground truth.

Since the ground truth data itself can be inaccurate, we introduce also in this case a tolerance factor δ, and eliminate from comparison points that are closer than δr from the surface of a ground truth tube of radius r.

Fig. 8(b) shows the precision-recall results for $\delta = 0.4$. In this case also, our method with iterative regression outperforms all the others for all the relevant ranges of precision and recall.

We observe the biggest improvement for the Aerial dataset. There, model-based methods do worst because they respond strongly to bright polygonal objects such as houses. Learning-based methods can be taught to discount them, and in this case, classification does better than hand-crafted methods, but still not as well as our approach. Classification is also competitive on the Vivo2P dataset. The reason is that there are mainly thin branches in this dataset. However, our method gives the higher accuracy.

On the Brightfield and VC6 datasets, our approach still does best but classification does worst, especially in Brightfield case, due to the presence of very wide branches. As shown in Fig. 1, in such cases, the maximum response is not necessarily on the centerline and non-maxima suppression behaves badly. Our regression-based approach avoids this problem. As observed in [53], the antisymmetric term introduced by OFA degrades the results with respect to OOF for very irregular structures. However, with and without it, OOF is more sensitive than our algorithm to strong artifacts and image noise, which are hard to ignore for hand-crafted methods.

4.3 Tracing Evaluation

The evaluation measures used to compare the results in the previous section are only local measures. For the problem of linear structure reconstruction it is also interesting to have a global measure able to evaluate how well a tubularity score can be used to trace linear structures.

To this end, we use in this section the tubularity scores obtained with our method and the baselines with the Fast Marching to generate paths between two points on a connected linear structure. We then evaluate how well these paths match the ground truth path using the approach proposed in [45], which we adapted to take into account also the estimation of the radius. In particular we use the overlapping measure OV and the accuracy measure AD. OV represents the ability to track the complete ground truth path, and is defined as

$$OV = \frac{TPM + TPR}{TPM + TPR + FN + FP} , \tag{11}$$

where TPR and TPM are the numbers of true positives in the reference path and the reconstructed path respectively, and FN and FP are the numbers of false negatives and false positives. In addition of the condition defined in [45] for true positives, we also take into account the radial estimation and impose that $\min(r_{\text{est}}, r_{\text{gt}})/\max(r_{\text{est}}, r_{\text{gt}}) > th$, where r_{est} is the radial estimation at one point, r_{gt} the ground truth radius and $th \in [0, 1]$ is a threshold value. Setting $th = 0$ means not taking into account the estimation of the radius and thus having the same definition as [45]. For $th = 1$ instead perfect match of the radius is required. In our experiments we set $th = 0.75$.

AD is the average distance between ground truth path and the path extracted automatically. It was computed in the scale-space to again take into account the radial estimation.

For each dataset, we randomly sampled 1000 paths of fixed length L from the ground truth, generated the paths joining the starting and ending points of these patches using the different tubularity measures and the Fast Marching algorithm and finally computed the corresponding OV and AD values.

Fig. 10 shows the OV and AD values as a function of the path length L. Note that as the length of the path increases our method remains more robust. Unlike the others, it also retains its accuracy. As shown in Fig. 9, the resulting paths follow the true linear structures over longer distances, without being disturbed by adjacent structures or background objects.

For smaller values of L, all methods perform well. The OOF-based measures even appear slightly better than the learning-based ones. This is an artifact due to the fact that the ground-truth paths have been generated using a semi-automated tracing tool that itself relies on OOF [6]. This is particularly true for the Vivo2P dataset, that only features short dentritic trees with simple topology.

4.4 Automated Reconstruction

We evaluated our approach in combination with a state-of-the-art tracing algorithm [54]. A tubularity measure is used in the first step of the algorithm to build a graph. This graph is then processed to extract the subgraph describing the linear structures. In the original implementation of [54] the tubularity measure was OOF [27]. Here we used instead the more accurate MDOF measure [53]. We used the DIADEM score [4] as an evaluation metric. The results are reported in Table 2. Using our method yields a substantial improvement on the challenging Aerial and Brighfield datasets. On the Vivo2P dataset the difference is smaller but our method still performs slightly better. This is due to the comparative simplicity of that dataset, which contains only structures with a simple topology.

The reconstruction obtained with our method are shown in Fig. 11 for two Aerial images and in Fig. 12 for a Brightfield and a Vivo2P image stacks. In particular from Fig. 11(d) we see that we are able to correct errors present in the ground truth.

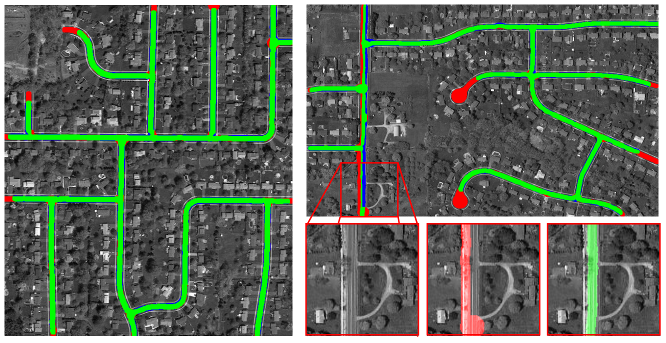


Fig. 11: Road tracing. We used our method as a preprocessing step for a state-of-the-art tracing algorithm to reconstruct road networks. The green color corresponds to true positives, red to false negatives, and blue to false positives. Most of the mistakes are made at the ends of the roads. (a) An image where the roads were almost perfectly reconstructed. (b)-(d) Another image for which our method correctly recovers the centerlines and the radii (d) while the ground truth was incorrect (c) for the vertical road on the bottom-left part of the image. Best viewed in color.

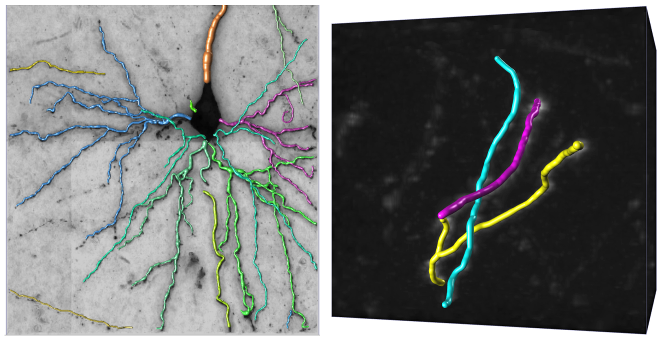


Fig. 12: Automated neuron delineations obtained by feeding the output of our approach to the algorithm of [54]. Different colors indicate different dentritic trees. (e) Reconstruction of neurons in a Brightfield image stack; (f) Reconstruction in a Vivo2P image stack. Best viewed in color.

TABLE 2: Automated reconstruction results. DIADEM scores computed on the reconstruction returned by [54] using either our method or MDOF [53] as initial tubularity measure. For the aerial dataset, we considered only the images containing road networks having a tree topology, since the DIADEM score is not defined for graphs.

Method	Aerial	Brightfield	Vivo2P
Our Method + [54]	**0.84**	**0.60**	**0.71**
MDOF + [54]	0.75	0.56	0.70

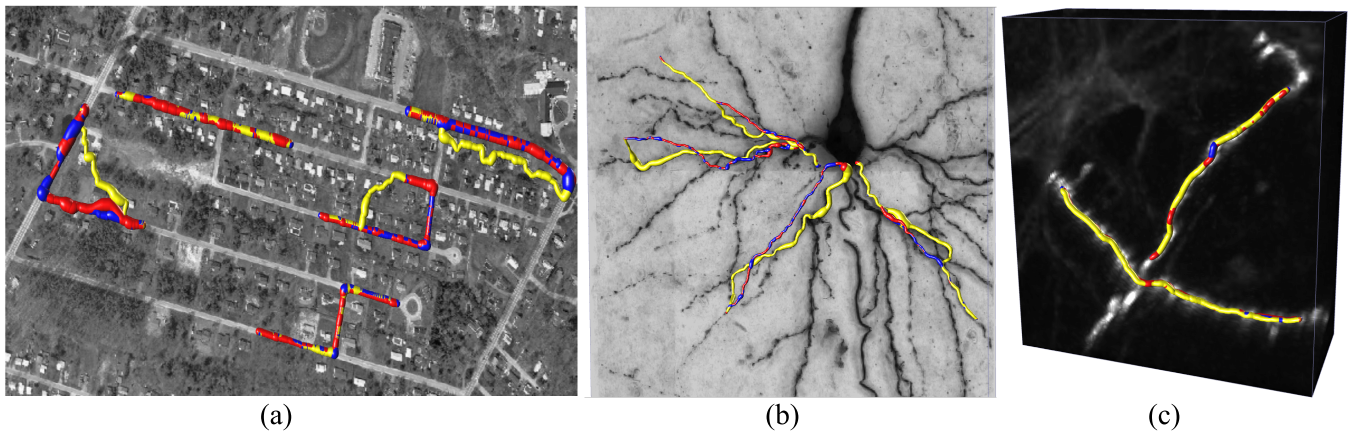
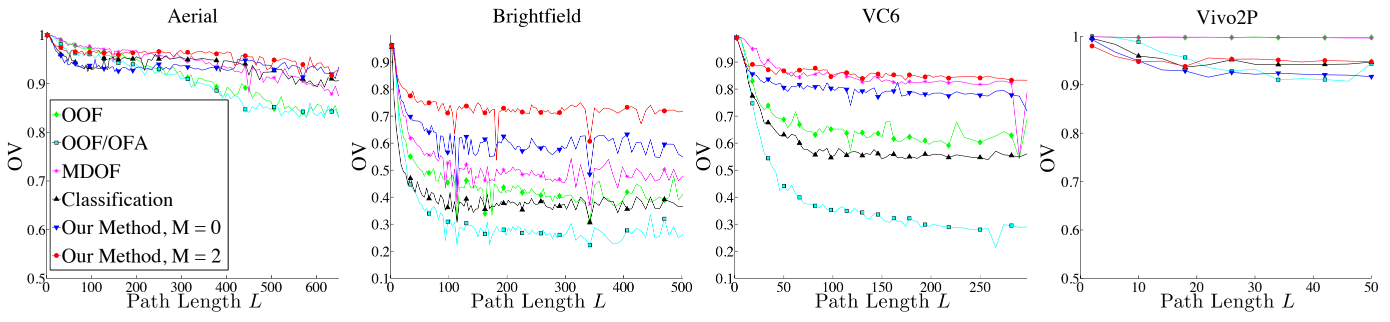
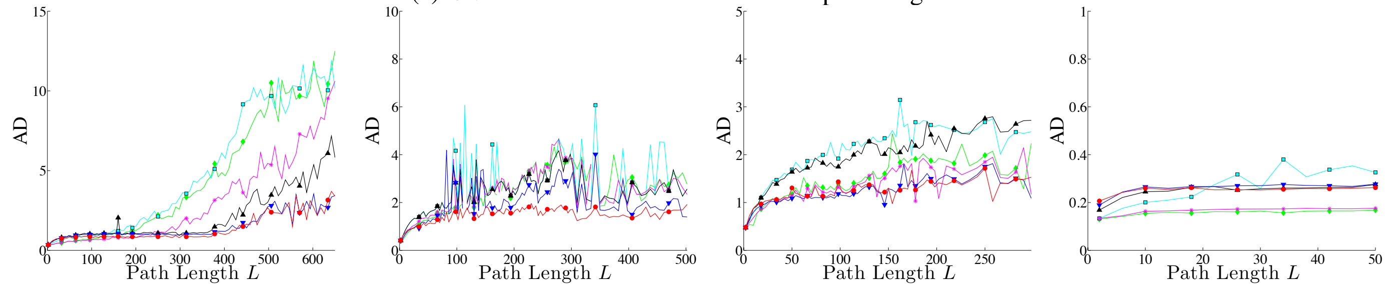


Fig. 9: Example of random paths used in the Tracing Evaluation of Section 4.3. The ground truth paths are represented in blue. The paths obtained from the MDOF tubularity score are represented in yellow, while those obtained using the tubularity score returned by our method are in red. The paths obtained with our method are most of the time much closer to the ground truth paths. In particular our method is able to follow the linear structure on a longer distance even in case of complex tree topology, such as the Brightfield stack (b) or in images with many background objects, such as the Aerial image (a). In such cases, the paths returned using MDOF are partially on the background or on adjacent structures. In simpler situations, such as for the Vivo2P dataset (c), the two methods are both able to provide the correct path. Best viewed in color.



(a) OV measure as a function of the path length L.



(b) AD measure as a function of the path length L.

Fig. 10: Tracing Evaluation. Our method is more robust when used to trace linear structures. The accuracy of our method remains constant for large values of the path length L, while the performance of the other methods decrease. OV is the fraction of points on the ground truth path marked as true positives, the larger the better. AD is the average distance between ground truth path and centerlines extracted automatically, the smaller the better. On the simpler Vivo2P dataset all methods perform well; OOF-based measures appear slightly better than the learning-based ones because the ground-truth paths have been generated using a semi-automated tracing tool that relies on OOF [6].

4.5 Boundary Detection

To demonstrate how generic our approach is, we consider here the problem of boundary instead of centerline detection. As centerlines, natural image boundaries are one dimensional structures whose exact location can be uncertain, as shown in Fig. 13(a). In this example and as often the case, the local appearance of boundary pixels and of their neighbors are extremely similar. In fact, as shown in Fig. 13(b), different people may mark different pixels as boundary points.

Our regression-based approach gives us a robust way to deal with this uncertainty as shown in Fig. 13(d) and we can ensure a single maximal response for each boundary.

In the remainder of this section, we describe how we adapt our method for boundary detection purposes and the dataset used to test the performance. To test our algorithm we used the Berkeley BSDS500 dataset [2]:

- **BDSD500** is a standard dataset used to test boundary detection algorithms. It is composed of 500 color images. 200 are used for training, 100 for validation and the remaining 200 for testing. Ground truth is made of boundaries as drawn by several human annotators.

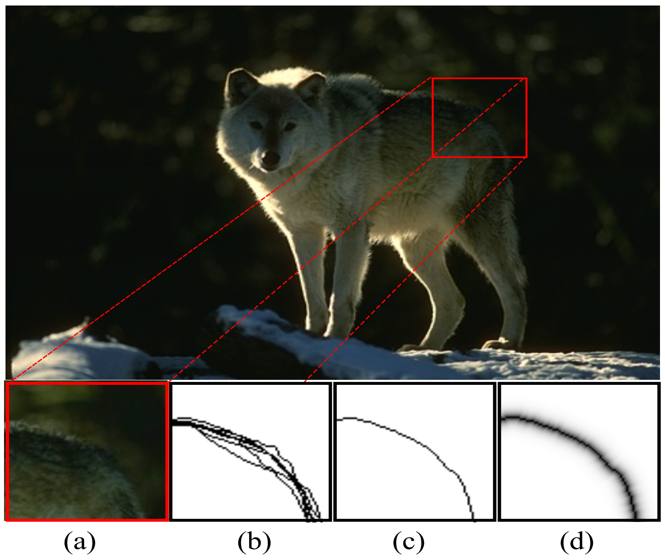


Fig. 13: Learning to predict boundaries in natural images. (a) Detail of the original image on the top. (b) The ground truth annotated by several human subjects. All the annotations are typically used in classification based approaches. This can produce multiple responses on a boundary. (c) Aligned ground truth obtained using [3]. (d) The function d we would like to learn with our method ensures a single detection and can model uncertainty.

Since the appearance of a boundary is often more complex than linear structures, we used 10^6 training samples and trees of depth 5 as weak learners. Moreover, in natural images color and texture are important sources of information to detect boundaries [34]. To compute color features we converted the input RGB images to Luv color space and learned a different filter bank on each channel. To detect texture boundaries instead, we added to our feature vector a corresponding pooled version. We tried different pooling strategies and the one that worked better for us was to take the absolute value of the average over a 5×5 region. As for centerline detection, we use $M = 2$ regression iterations.

In the BSDS500 dataset the ground truth is made of 5 different annotations. In order to compute the function d of Eq. (3), we first align the different annotations using the approach described in [3], as shown in Fig. 13(c). In this way we obtain a single response for each boundary and we compute the function d from this binary map.

Finally, as done in [12], [41], [47], we also run our boundary detector on the test images at 3 different resolutions: half, original and double size, and then average the results. [1]

The Precision-Recall curves obtained using the standard Berkeley benchmark [2] are shown in Fig. 14. Our method is more accurate than state-of-the-art techniques and, unlike gPb [2], SCG [41] and MCG [3], does not include any globalization step.

The advantage of our method is mainly given by the regression approach: The Classification results also shown in Fig. 14 are significantly worse. These results were obtained

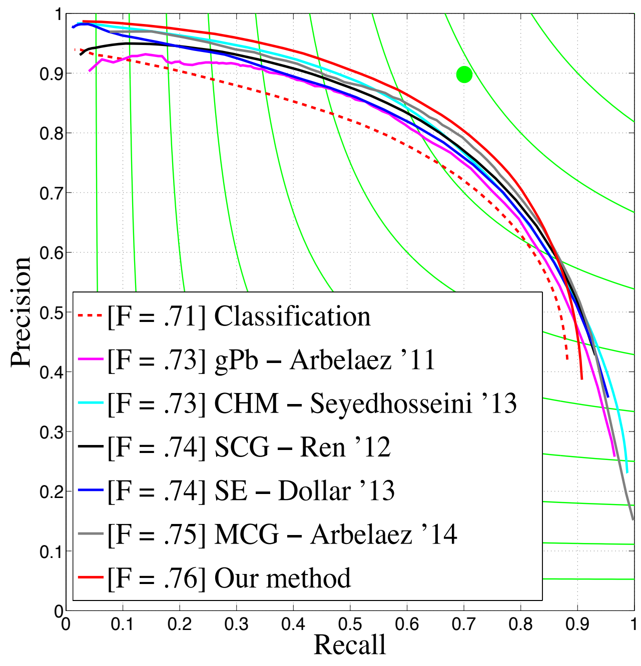


Fig. 14: Precision-Recall curves BSDS500 dataset for different boundary detection methods. In bracked is shown the F measure computed with a fixed threshold for every image (ODS). Our method achieve state-of-the-art performance also on this task. The advantage of out method is mainly given by the regression formulation. Retraining our model using the binary ground truth to classify the boundaries deteriorate the performance by 5% (red dashed line in the graph).

with the same implementation as for our regression method, using the same image features and the same number of auto-context iterations—the only difference was the objective function minimized during training: The classifier was trained to classify the pixels lying on boundaries versus the other pixels by minimizing the exponential loss, rather than regressing the distance transform.

The boundaries detected by the different methods on some test images are shown in Fig. 16. When we compare the qualitative results obtained with our method against two state-of-the-art classification based detectors, namely SCG [41] and CHM [47], we see that our approach can avoid double responses and multiple detections (Fig. 15). The gPb-ucm [2] and SE [12] methods do not have this problem since they obtain boundaries after segmenting an image or an image patch in different regions. However, our method is still more accurate and also gives a general framework that is not limited to the contour detection problem. Moreover, as a possible extension of our method, we can naturally incorporate information about the strength of a boundary [3], [32] by modifying the shape of the distance function we want to learn.

5 Conclusion

We have introduced an efficient regression-based approach to centerline detection, which we showed to outperform both methods based on hand-designed filters and classification-based approaches.

The output of our method can be used in combination with tracing algorithms requiring a scale-space tubularity measure as input, increasing accuracy also on this task.

1. The code and the parameters used in our experiments are publicly available at: `http://cvlab.epfl.ch/software/centerline-detection`.

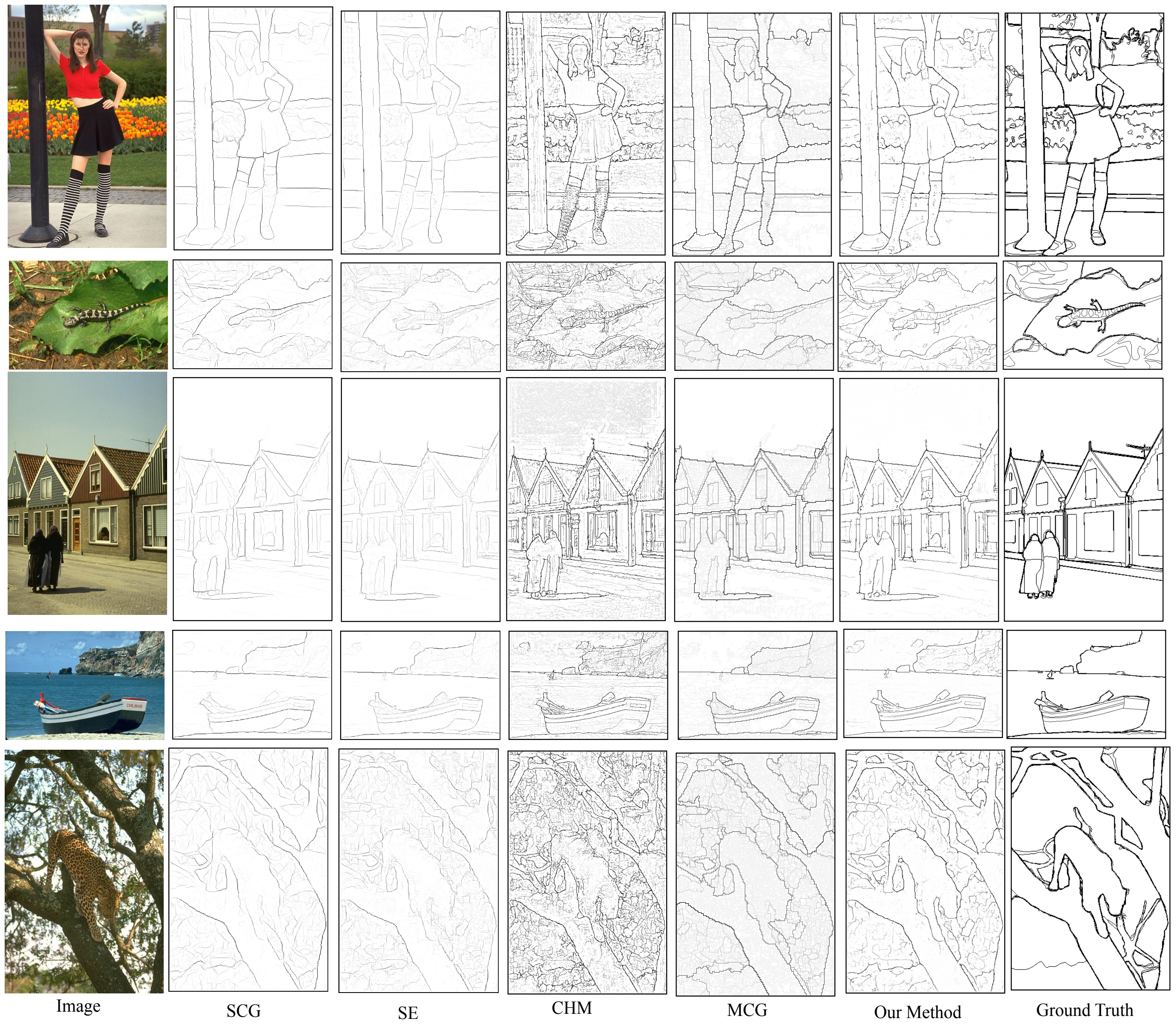


Fig. 16: Boundary detection results. Our method is able to capture finer details compared to MCG, and is also more robust to false edges than CHM. See for example the lezard in the second top image or the front paw of the leopard on the bottom.

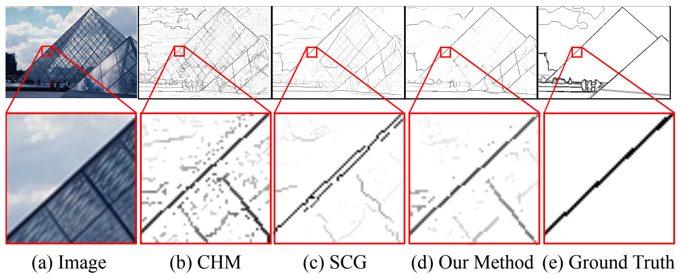


Fig. 15: Compared to classification based approaches, the boundaries detected by our method are better localized and less noise is detected on the background. In particular, from the details of the image at the bottom, we see that thanks to our regression formulation we avoid multiple detections of a boundary.

Our approach is very general and applicable to other linear structure detection tasks when training data is available. For example, we obtained an improvement over the state-of-the-art when training it to detect boundaries on a set of natural images.

References

[1] G. Agam and C. Wu. Probabilistic Modeling-Based Vessel Enhancement in Thoracic CT Scans. In *CVPR*, pages 684–689, 2005.
[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *PAMI*, 33(5):898–916, 2011.
[3] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marqués, and J. Malik. Multi-scale combinatorial grouping. In *CVPR*, 2014.
[4] G. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge, 2010.
[5] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised Feature Learning for Curvilinear Structure Segmentation. In *MICCAI*, September 2013.
[6] F. Benmansour and L. Cohen. Tubular Structure Segmentation Based on Minimal Path Method and Anisotropic Enhancement. *IJCV*, 92(2):192–210, 2011.

[7] D. Breitenreicher, M. Sofka, S. Britzen, and S. Zhou. Hierarchical Discriminative Framework for Detecting Tubular Structures in 3D Images. In *International Conference on Information Processing in Medical Imaging*, 2013.
[8] J. Canny. A Computational Approach to Edge Detection. *PAMI*, 8(6), 1986.
[9] D. Cireşan, A. Giusti, L. Gambardella, and J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *NIPS*, 2012.
[10] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-Column Deep Neural Networks for Image Classification. In *CVPR*, 2012.
[11] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries. In *CVPR*, 2006.
[12] P. Dollár and C. L. Zitnick. Structured Forests for Fast Edge Detection. In *ICCV*, 2013.
[13] M. Fink and P. Perona. Mutual boosting for contextual inference. In *In NIPS*. MIT Press, 2004.
[14] A. H. Foruzan, R. A. Zoroofi, Y. Sato, and M. Hori. A Hessian-Based Filter for Vascular Segmentation of Noisy Hepatic CT Scans. *International Journal of Computer Assisted Radiology and Surgery*, 7(2):199–205, 2012.
[15] A. Frangi, W. Niessen, K. Vincken, and M. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998.
[16] J. Friedman. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, 2002.
[17] G. Gonzalez, F. Aguet, F. Fleuret, M. Unser, and P. Fua. Steerable Features for Statistical 3D Dendrite Detection. In *MICCAI*, pages 625–32, September 2009.
[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
[19] G. Heitz and D. Koller. Learning Spatial Context: Using Stuff to Find Things. In *ECCV*, 2008.
[20] X. Huang and L. Zhang. Road Centreline Extraction from High-Resolution Imagery Based on Multiscale Structural Features and Support Vector Machines. *International Journal of Remote Sensing*, 30:1977–1987, 2009.
[21] ISBI Challenge: Segmentation of neuronal structures in EM stacks. http://brainiac2.mit.edu/isbi_challenge, 2012.
[22] M. Jacob and M. Unser. Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. *PAMI*, 26(8):1007–1019, August 2004.
[23] E. Jurrus, A. Paiva, S. Watanabe, J. Anderson, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using a Serial Neural Network Architecture. *MIA*, 14(6):770–783, 2010.
[24] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset. Model-Based Detection of Tubular Structures in 3D Images. *CVIU*, 80(1):130–171, November 2000.
[25] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, pages 1097–1105, 2012.
[26] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *ECCV*, 2008.
[27] M. Law and A. Chung. An Oriented Flux Symmetry Based Active Contour Model for Three Dimensional Vessel Segmentation. In *ECCV*, pages 720–734, 2010.
[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *PIEEE*, 1998.
[29] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch Tokens: A Learned Mid-Level Representation for Contour and Object Detection. In *CVPR*, 2013.
[30] T. Lindeberg. Scale-Space for Discrete Signals. *PAMI*, 12(3):234–254, 1990.
[31] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR*, 2008.
[32] M. Maire, S. X. Yu, and P. Perona. Hierarchical Scene Annotation. In *BMVC*, 2013.
[33] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London, Biological Sciences*, 207(1167):187–217, 1980.
[34] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *PAMI*, 26(5), 2004.
[35] E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. *Cytometry Part A*, 58A(2):167–176, April 2004.
[36] H. Mirzaalian, T. Lee, and G. Hamarneh. Hair enhancement in dermoscopic images using dual-channel quaternion tubularness filters and mrf-based multi-label optimization. *TIP*, 2014.
[37] V. Mnih and G. Hinton. Learning to Label Aerial Images from Noisy Data. In *ICML*, 2012.
[38] A. Narayanaswamy, Y. Wang, and B. Roysam. 3D Image Pre-Processing Algorithms for Improved Automated Tracing of Neuronal Arbors. *Neur. Inf.*, 9(2-3):219–231, 2011.
[39] M. Pechaud, G. Peyré, and R. Keriven. Extraction of Tubular Structures over an Orientation Domain. In *CVPR*, pages 336–342, 2009.
[40] R. R. Deriche. Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. *IJCV*, 1(2):167–187, 1987.
[41] X. Ren and L. Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *NIPS*, December 2012.
[42] R. Rigamonti and V. Lepetit. Accurate and Efficient Linear Structure Segmentation by Leveraging Ad Hoc Features with Learned Filters. In *MICCAI*, 2012.
[43] A. Santamaría-Pang, T. Bildea, C. M. Colbert, P. Saggau, and I. Kakadiaris. Towards Segmentation of Irregular Tubular Structures in 3D Confocal Microscope Images. In *MICCAI Workshop in Microscopic Image Analysis and Applications in Biology*, 2006.
[44] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. *MIA*, 2:143–168, June 1998.
[45] M. Schaap, C. Metz, T. van Walsum, A. V. der Giessen, A. Weustink, N. Mollet, C. Bauer, H. Bogunovi, C. Castro, X. Deng, E. Dikici, T. O'Donnell, M. Frenay, O. Friman, M. H. Hoyos, P. Kitslaar, K. Krissian, C. Kuhnel, M. A. Luengo-Oroz, M. Orkisz, O. Smedby, M. Styner, A. Szymczak, H. Tek, C. Wang, S. K. Warfield, S. Zambal, Y. Zhang, G. Krestin, and W. Niessen. Standardized Evaluation Methodology and Reference Database for Evaluating Coronary Artery Centerline Extraction Algorithms. *MIA*, 13/5:701–714, 2009.
[46] M. Seyedhosseini, R. Kumar, E. Jurrus, R. Guily, M. Ellisman, H. Pfister, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using Multi-Scale Context and Radon-Like Features. In *MICCAI*, pages 670–677, 2011.
[47] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *ICCV*, 2013.
[48] A. Sironi, V. Lepetit, and P. Fua. Multiscale Centerline Detection by Learning a Scale-Space Distance Transform. In *CVPR*, 2014.
[49] A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit, and P. Fua. Learning Separable Filters. *PAMI*, 99, 2014.
[50] M. Sofka and C. Stewart. Retinal Vessel Centerline Extraction Using Multiscale Matched Filters, Confidence and Edge Measures. *TMI*, 2006.
[51] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection. In *CVPR*, 2004.
[52] Z. Tu and X. Bai. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *PAMI*, 2009.
[53] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua. Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *ICCV*, December 2013.
[54] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing Loopy Curvilinear Structures Using Integer Programming. In *CVPR*, June 2013.
[55] E. Turetken, F. Benmansour, and P. Fua. Semi-Automated Reconstruction of Curvilinear Structures in Noisy 2D Images and 3D Image Stacks. Technical report, EPFL, 2013.
[56] C. Wang, Y. Li, W. Ito, K. Shimura, and K. Abe. A Machine Learning Approach to Extract Spinal Column Centerline from Three-Dimensional CT Data. In *SPIE*, 2009.
[57] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler. A Higher-Order CRF Model for Road Network Extraction. In *CVPR*, 2013.
[58] C. Xiao, M. Staring, Y. Wang, D. Shamonin, and B. Stoel. Multiscale Bi-Gaussian Filter for Adjacent Curvilinear Structures Detection with Application to Vasculature Images. *TIP*, 22(1):174–188, 2013.
[59] Y. Zheng, M. Loziczonek, B. Georgescu, S. Zhou, F. Vega-Higuera, and D. Comaniciu. Machine Learning Based Vesselness Measurement for Coronary Artery Segmentation in Cardiac CT Volumes. *SPIE*, 7962(1):79621–7962112, 2011.
[60] S. K. Zhou, C. Tietjen, G. Soza, A. Wimmer, C. Lu, Z. Puskas, D. Liu, and D. Wu. A Learning Based Deformable Template Matching Method for Automatic Rib Centerline Extraction and Labeling in CT Images. In *CVPR*, 2012.
[61] Q. Zhu, D. Zheng, and H. Xiong. 3D Tubular Structure Extraction Using Kernel-Based Superellipsoid Model with Gaussian Process Regression. In *VCIP*, pages 1–6, 2012.