ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Filter Learning for Linear Structure Segmentation[*]

Roberto Rigamonti (roberto.rigamonti@epfl.ch)
http://cvlab.epfl.ch/~rigamont

Engin Türetken (engin.turetken@epfl.ch)
http://cvlab.epfl.ch/~turetken

Germán González (german.gonzalez@epfl.ch)
http://cvlab.epfl.ch/~ggonzale

Pascal Fua (pascal.fua@epfl.ch)
http://cvlab.epfl.ch/~fua

Vincent Lepetit (vincent.lepetit@epfl.ch)
http://cvlab.epfl.ch/~lepetit

School of Computer and Communication Sciences
Swiss Federal Institute of Technology, Lausanne (EPFL)

Technical Report

June 14, 2011

[*]This work has been supported in part by the Swiss National Science Foundation.

Abstract

We introduce an approach to learning convolution filters whose joint output can be fed to a classifier that labels them as belonging to linear structures or not. The filters are learned using sparse synthesis techniques but we show that enforcing constraints is not required at run-time to achieve good classification performance. In practice, this is important as it drastically reduces the computational cost.

We show that our approach outperforms the state-of-the-art on difficult, and very different, images of roads, retinal scans, and dendritic networks.

Contents

Chapter 1

Introduction

Linear structures appear at many different scales and in many different contexts. They can be micrometer scale dendrites in light microscopy image-stacks, centimeter-scale blood vessels in retinal scans, or meter-scale road networks in aerial images. Extracting them automatically and robustly is therefore of fundamental relevance to many scientific disciplines. However, even though the topic has received sustained attention ever since the inception of the field of Computer Vision, both robustness and automation remain elusive.

An important first step is to detect pixels that appear to be the most filament-like so that they can then be linked into complete line-like structures. Many techniques have been proposed over the years and some of the most successful ones involve computing the Hessian matrix at individual pixels by convolving with Gaussian derivatives and relying on the matrix eigenvalues to classify pixels as filament-like or not [28, 10, 32]. This kind of approach, however, works best when the linear structures appear as regularly-shaped ribbons, which is not the case in images such as those of Fig. 3.1 (left). Furthermore, they tend to fail around junctions. Recently, it was shown that it was possible to improve upon this situation by convolving the image with steerable filters of various scales and orientations and training an SVM to classify the pixels as belonging to filaments or not on the basis of their output [11]. The improvement comes from the fact that the SVM can learn to recognize more complex appearances than that of a simple ribbon. However, in that approach, the filters themselves are *ad hoc*.

In this paper, we go one step further and show that the filters themselves can be learned using techniques from the dictionary learning literature. This yields better results than the methods that rely either on the Hessian [10] or the steerable filters [11] for many different kinds of linear structures.

More specifically, we exploit the fact that natural images can be represented as linear combinations of a relatively small number of dictionary elements, which can be learned in an unsupervised way from training images. This has been extensively used for image denoising and object recognition purposes [25, 30, 19, 13, 4], but only rarely for segmentation purposes. Furthermore most existing approaches rely on a matrix formulation that is computationally intensive both for training and at run-time. Here we learn convolutional filters instead [17, 35, 26]. They are still expensive to train. However, they are more versatile because they can be used for images of any size and yield much faster run-times than those of algorithms that use dictionary items of the same size as the input images. They also do not produce the stitching artifacts inherent to approaches that represent images as arrays of patches stitched together. As in [26], we do not need to explicitly compute the weights required to reconstruct the images. Instead, we can directly convolve the input image with the learned filters and feed their responses to a classifier. This is important in practice because it removes the requirement for time-consuming coefficients estimation.

In the remainder of the paper, we first discuss related work on linear structure detection and dictionary learning. We then present our method and our results.

Chapter 2

Relevant work

One simple approach to filament detection is to assume that the filament profile has a ridge shape [5]. This was used in [21] to derive efficient steerable filters for neurite tracing. The Hessian matrix of the image signal and its eigenvalues have also been intensively used to detect tubular structures [28, 10, 14, 32]. To detect filaments of various widths, a range of variances for the Gaussian derivative filters must be used and compared. Other models have used differential kernels [2], looked for parallel edges [8], or fitting superellipsoids to the image [29, 34]. However real linear structures do not necessarily conform to these *ad hoc* models, which can drastically impact performance.

As a result, machine-learning based approaches that can learn more convoluted appearances have become increasingly attractive. In [1], the distribution of the eigenvalues of the structure tensor are estimated via Expectation Maximization. Probabilistic Boosting Trees with sparse rotational features have also been demonstrated for vessel segmentation purposes [31]. Support Vector Machines operating on the Hessian's eigenvalues have been used to discriminate between filament and non-filament pixels [27]. In [11] rotational features were computed at each pixel using steerable filters and fed to an SVM to classify pixels as filament-like or not. It is closely related to our approach but still relies on *ad hoc* filters whereas we learn them. As it outperforms the previous methods, we will compare our approach to it as well as to a Hessian based method [10], which probably is one of the most popular methods.

Our approach to filter learning is inspired by the recent literature on image modeling [25, 30, 19, 13, 4]. The assumption is that a natural image can be modeled as a sum of a few elements among a large dictionary that can be learned. The optimization problem is large, but solving it has become a very active area of research, and it usually converges nicely. This is in contrast with neural networks, which are also related to our approach but are more prone to local minimums [3, 12, 22]. The model is usually formulated in matrix form, which would be inefficient for our purposes and we adopt a convolutional approach [17, 35, 26]. Dictionary learning have been applied mostly to image restoration [20] and object recognition [24], but not, to the best of our knowledge, to image segmentation in the literature. It was suggested very recently in [18] but no experiment was presented. By contrast, here we demonstrate the validity of our approach on challenging data.

Chapter 3

Approach

During a training phase, we first learn a filter bank from a set of training images that are representative of the data we want to segment. We tried both an unsupervised approach and a supervised one that relies on segmentation ground truth for training purposes.

Given the filter banks, a feature vector can be computed for each pixel of a given image, and a binary classifier trained to decide if the pixel lies or not on a filament based on its feature vector.

3.1 Learning the Filters

The unsupervised approach we use to learn a bank of linear filters $\{\mathbf{f}^j\}$ does so by solving

$$\min_{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \lambda_{learn} \sum_j \left\| \mathbf{t}_i^j \right\|_1 \right) , \tag{3.1}$$

where the $\mathbf{x}_i$ are the training images, $*$ denotes the convolution operator, and the $\mathbf{t}_i^j$s are images of the same size as the $\mathbf{x}_i$ images, whose cardinality is that of the filter bank. Similar intermediate representations have been called "feature maps" in the Convolutional Neural Networks literature [15]. In practice, we take the $\mathbf{f}^j$s to be a bank of 121 21×21 filters.

Eq. (3.1) is the convolutional version of the equation considered in compressive sensing [9] or LASSO regularization [33]. It seeks a set of filters $\mathbf{f}^j$ so that the images $\mathbf{x}_i$ can be reconstructed by convolving the feature maps $\mathbf{t}_i^j$ that are forced to be sparse by the second term. λ_{learn} is a regularization parameter that establishes the relative importance of the two terms. We use stochastic gradient descent with clipping [7] to find the coefficients for the $\mathbf{f}^j$ filters, alternatively optimizing the $\mathbf{f}^j$s and the $\mathbf{t}_i^j$s in Eq. (3.1). An example of a filter bank together with one of the training images is depicted by Fig. 3.1.

One key weakness of this formulation is that nothing prevents two filters from independently converging to an identical solution. This usually happens when strong gradients, which dominate the reconstruction error term, are present in the images. It therefore tends to happen with images containing neat, curvilinear profiles. The regularization term pushes for an economy in the representation. However, the regularization parameter λ_{learn} cannot make the sparsity penalty prevail over the reconstruction error without trivial filters appearing. Furthermore, the ℓ_1 regularizer penalizes similarly all cases where a certain amount of energy is split equally among similar filters. In fact, this is the main difficulty in using the ℓ_1 norm in place of the ℓ_0 norm for sparsity promotion. Surprisingly enough, and to the best of our knowledge, there is no approach that aims specifically at solving this issue.

In order to encourage filters to assume different shapes, we have altered the optimization problem in Eq. (3.1) by adding a penalty term that accounts for the squared dot product between the filters:

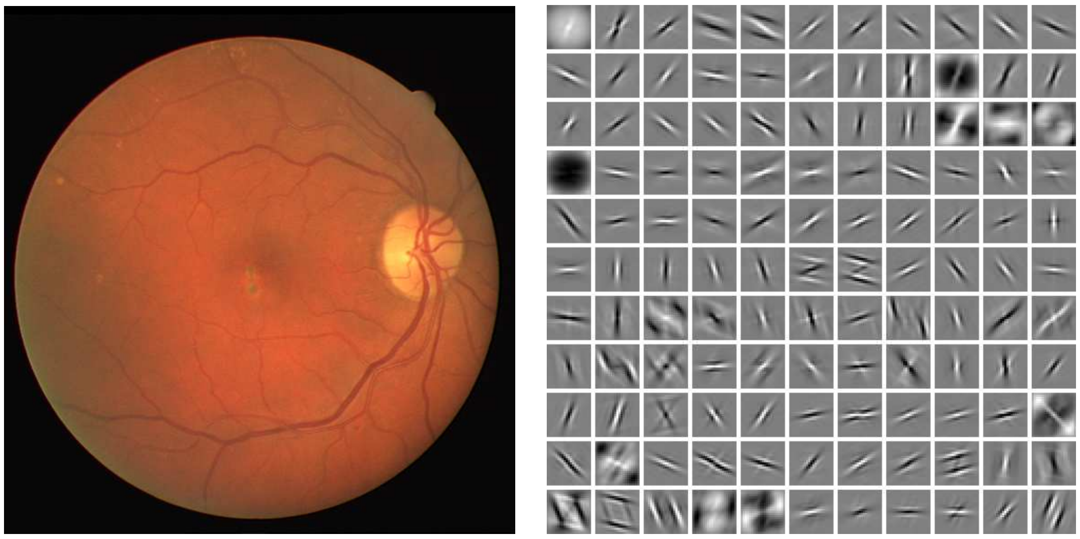

Figure 3.1: **Left:** One of the retinal scans we used in our experiments. **Right:** Learned filters for the retinal scans using the unsupervised approach of Eq.(3.2).

We solve

$$\min_{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \lambda_{learn} \sum_j \left\| \mathbf{t}_i^j \right\|_1 + \xi \sum_j \sum_{k \neq j} \left(\langle f^j, f^k \rangle \right)^2 \right) \quad , \tag{3.2}$$

where the third term penalizes filters that are too close from each other and whose dot product is therefore large. This is also solved by stochastic gradient descent with clipping, and tends to converge nicely.

3.2 Computing the Feature Vectors

The $\mathbf{f}^j$ filters can then be used to compute a feature vector for each pixel of an image x. One way to do this is to perform an optimization similar to Eq. (3.1) on the feature maps $\mathbf{t}^j$ only:

$$\min_{\{\mathbf{t}^j\}} \left(\left\| \mathbf{x} - \sum_j \mathbf{f}^j * \mathbf{t}^j \right\|_2^2 + \lambda_{segm} \sum_j \left\| \mathbf{t}^j \right\|_1 \right) . \tag{3.3}$$

As noted by previous works [24, 26], the regularization parameter λ_{segm} should be much smaller than λ_{learn} to reach good performances. The feature vector $\mathbf{v}(z)$ for a pixel z is then made of corresponding values in the $\mathbf{t}^j$:

$$\mathbf{v}(z) = [\mathbf{t}^1(z), \ldots, \mathbf{t}^n(z)]^\top , \tag{3.4}$$

where n is the number of filters.

The optimization of Eq. (3.3) is, however, computationally expensive. We therefore tried another approach, where we simply take the feature vectors as the results of the convolution between the learned filters and the image:

$$\mathbf{v}(z) = [(\mathbf{f}^1 * \mathbf{x})(z), \ldots, (\mathbf{f}^n * \mathbf{x})(z)]^\top . \tag{3.5}$$

This is much faster to compute, and as our experiments show, performs as good as the previous approach.

Finally, we train a classifier to label pixels as filament or background based on their feature vectors, using a training set made of the training images and their ground truth segmentations. We do not use any post-processing.

3.3 Potential Improvements

Because the segmentation ground truth is also available, we also tried to exploit it by introducing a supervised version of Eq. (3.1):

$$\min_{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \lambda_{learn} \sum_j \left\| \mathbf{t}_i^j \right\|_1 + \right.$$
$$\left. \xi \sum_j \sum_{k \neq j} \left(\langle f^j, f^k \rangle \right)^2 + \gamma \left\| \mathbf{y}_i - \tanh\left(\frac{3}{2} \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right) \right\|_2^2 \right) \quad , \tag{3.6}$$

where the $\mathbf{y}_i$s are images that contain the ground truth segmentations for the $\mathbf{x}_i$s, properly mapped in $[-1, 1]$, and γ weights the importance of the supervised term. The hyperbolic tangent is used to push the reconstruction obtained by convolution towards one of the two extrema represented by the ground truth. We are able to frame the problem in this manner because the areas we are interested in segmenting in retinal scans are typically darker than the rest of the image, therefore we can directly map the intensities to the ground truth data. In this way, a single dictionary can be used to filter the image and directly extract the segmentation.

This optimization problem is, however, more difficult than the one of Eq. (3.1), and convergence is difficult to achieve. In particular, when both the filter dot product and the supervised terms are included, the coefficients λ_{learn}, the stochastic gradient descent step η, ξ, and γ must be accurately orchestrated to thwart divergence. This turned out to be a very ticklish matter, increasing the attractiveness of unsupervised solutions.

To improve the convergence of the optimization scheme, one effective approach is to whiten the data [16]. This transforms a first order method, like our stochastic gradient descent, into a second order one, with a substantial acceleration in the convergence. Moreover, whitening has been recently observed to represent a key component in the classification pipeline for certain datasets [26, 6]. We have therefore extracted a whitening filter using the same technique presented in [26], and used it to whiten the dataset both at training and testing time for some of the experiments.

Chapter 4

Results

We evaluated our method on three very different datasets.

The first one is the publicly available DRIVE dataset of retinal images,with the aim of automatically segmenting blood vessels. It is composed of 40 RGB-formatted retinal scans, which are originally obtained for diagnosis of diabetic retinopathy. In our experiments, we used only the green channel since it has been shown to give the highest contrast between background and vessels [23]. Fig. 3.1 shows an example retinal scan from this dataset. The images typically have a uniform background with the vessels appearing as dark linear structures. We used segmentations of the underlying vasculatures provided by expert ophthalmologists as ground truth for performing Receiver Operating Characteristic (ROC) analysis.

The second dataset is made of brightfield micrographs such as that of Fig.4.1. The brightfield micrographs are obtained from biocityne-dyed rat brains. Due to irregularities of the staining process, they contain both structured and unstructured noise that is difficult to distinguish from the dendrites.

Our third dataset is made of satellite images such as that of Fig. 4.2. They contain road networks of a residential area in the United States. Segmenting streets from these images is a challenging task since they are often occluded by trees along roadsides and medians. Furthermore, the image intensities of the streets vary according to the quality of the concrete, and the background is cluttered with many complex structures that can be mistaken for roads, such as houses, swimming pools, and parking lots.

For ROC analysis of the two last datasets we manually annotated the dendrites and the streets and use these annotations as ground truth for training and testing.

In the following section, we first describe the experimental setup and provide comparative results for the three datasets. We then discuss the effect of enforcing sparsity instead of using plain filter convolutions during classification and that of incorporating supervision in learning the filter banks.

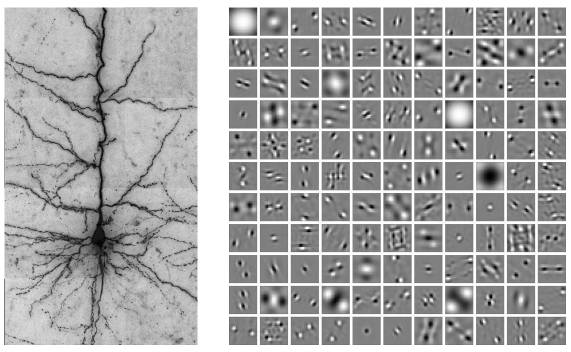

Figure 4.1: **Left:** One of the brightfield microscopy images used in our experiments. **Right:** Learned filters for the such dataset using the unsupervised approach.

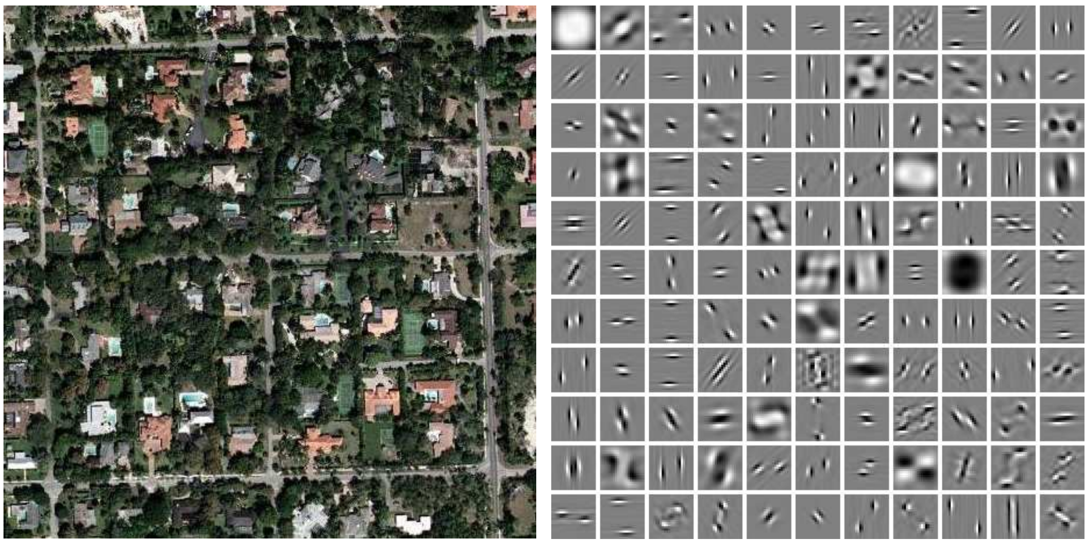

Figure 4.2: **Left:** One image of the roads dataset used in our experiments. **Right:** Learned filters for the such dataset using the unsupervised approach.

4.1 Experimental Setup

We divide the manually annotated ground truth for each dataset into disjoint training and test sets, leaving at least one whole image for testing. For supervised classification, positive examples are randomly selected from the centerlines of the linear structures and negative ones from the background. We collect hard negative instances by randomly sampling points within a certain distance from the linear structures. These examples constitute half of the negative samples. The other half are again randomly selected from the rest of the background. In our experiments, unless otherwise stated, we used 10000 positive and 10000 negative examples for training and validation for each dataset.

To account for contrast and brightness variations across different images, we rescale pixel intensity values using a zero-mean unit-variance normalization. For each sample $\mathbf{z}_n$ in the training set we then compute a feature vector $\mathbf{v}_n$ by convolving the learned filters with the normalized images. These feature vectors are used to train classifiers at training time and to obtain classification scores at test time. In this paper we used both AdaBoost and Support Vector Machines (SVMs) as baseline classifiers.

Recall that the function optimized by AdaBoost is

$$\psi(\mathbf{v}_n) = \sum_{i=1}^{M} \alpha_i h(\mathbf{v}_n), \tag{4.1}$$

where M stands for the number of the weak classifiers that form the strong classifier, and α_i is an scalar that multiplies the weak classifiers $h(\cdot)$. In this paper, we use two different functions of the feature vector as weak classifiers. The first one is a simple threshold on one of the vector coordinates. The second one is a threshold on the projection of v_n on a random unit vector u_n.

Similarly, the function that is optimized by the Support Vector Machine is

$$\phi(\mathbf{v}_n) = \sum_{i=1}^{M} \alpha_i k(\mathbf{v}_i, \mathbf{v}_n), \tag{4.2}$$

where $k(\mathbf{v}_i, \mathbf{v}_n)$ stands for the kernel evaluated between the vector under consideration and the support vector $\mathbf{v}_i$. In this work we use Gaussian kernels. To optimize the kernel variance ν and the regularization parameter C of the SVM, we perform 5-fold cross-validation while training. In Fig. 4.3 we plot the typical landscape of the validation error with respect to ν and C.

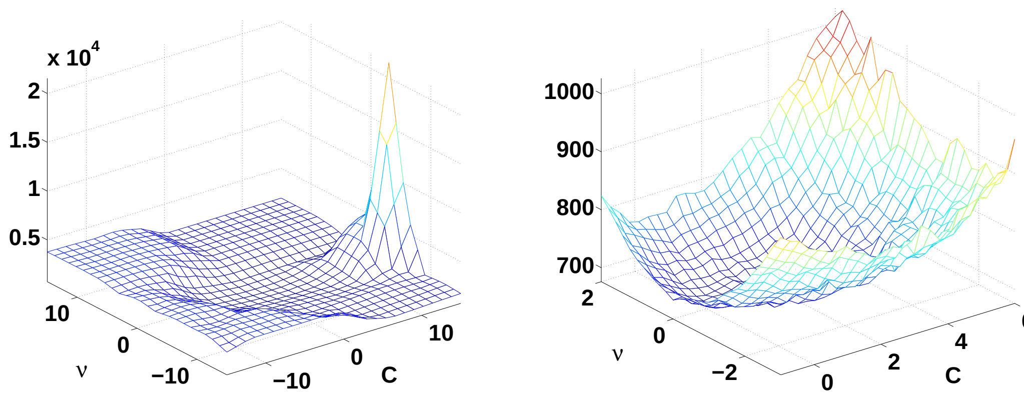

Figure 4.3: $C - \nu$ plot of the validation error for the DRIVE dataset. **Left:** Bird's eye view of the error function. **Right:** Close-up on the region that gives the best validation scores.

4.2 Comparative Results

We compared our results against the supervised learning approach of [11] that uses rotational filter banks and the well-known Hessian-based technique of Frangi *et al.* [10]. The former is similar to ours in spirit except for the fact that filters are chosen *a priori* instead of learned. The latter is one of the most widely used solutions for linear structure detection. For both methods we used multi-scale implementations and compared them to our filter banks learned at a single scale. We have chosen the steerable filter approach as one of the baselines since it also detects linear structures using a machine learning technique.

Figures 4.4, 4.5, and 4.6 summarize the results on our three datasets. The ROC curves for our method are consistently above the ones corresponding to our two baselines. Note that we also get consistently get better results by using SVM than AdaBoost. This is mainly due to the fact that the dimensionality of the feature spaces is relatively low—121 for our learned filters banks and 45 for the rotational filters—and hence the deterministic optimization of the SVM explores more exhaustively the search space than the random Adaboost selections.

The method in [11] does better than the one in [10] because it uses a richer vocabulary of filters that allows it to better account for irregularities in the data. However, these filters being weighted sums of Gaussians and Gaussian derivatives, they only have limited expressive power. Ours are learned on the data itself, and therefore are more expressive, at the cost of loosing the separability of the Gaussian filters.

4.3 Optimizing the feature maps does not help

As suggested in Sec.3.2, the same principles that were used for learning the filters could be used to optimize the feature maps using Eq. (3.3). This idea is at the underpinnings of many algorithms, in particular multilayer networks. We have compared the segmentation scores for the plain convolution case with those achieved by imposing several different levels of sparsity. The most significant results are reported in Fig. 4.8 and show that feature vectors computed by convolution performs better than the ones computed from optimized feature maps. This is an important result for practical applications as convolution is much faster than the feature map optimization.

4.4 Supervised filter learning does not help

Despite sparsity was initially conceived as a mechanism to drive learning in an unsupervised fashion to obtain biologically-plausible filters, recent works focused on learning these filters with

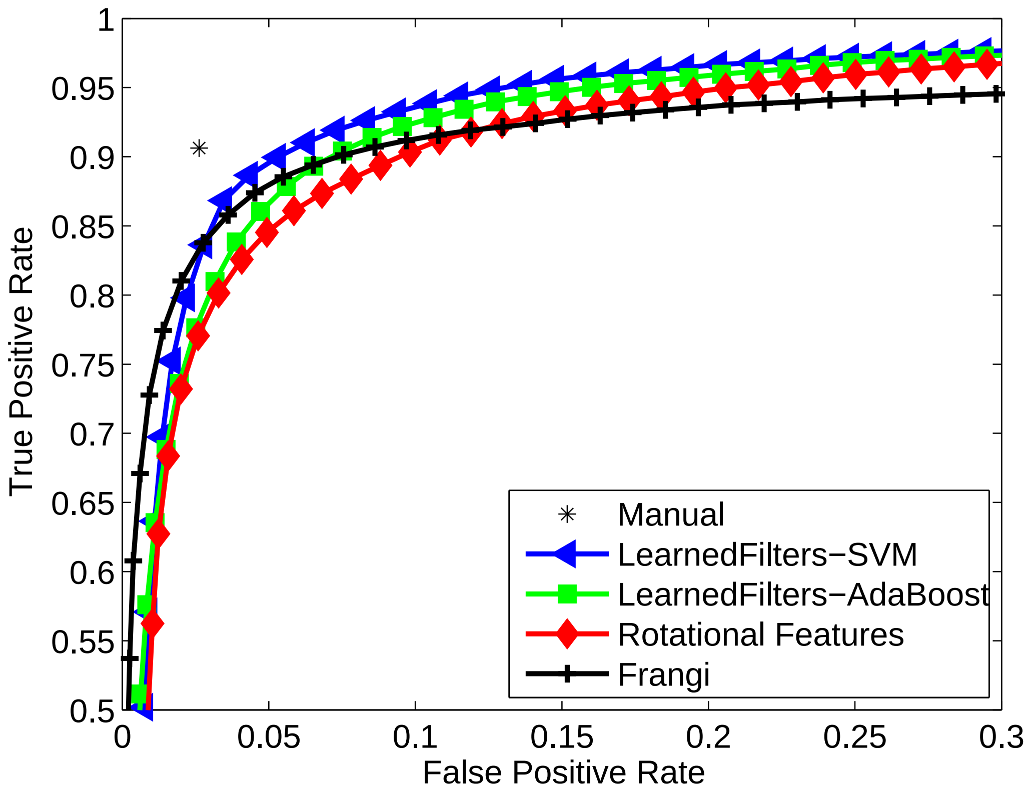

Figure 4.4: ROC curves for the DRIVE dataset. As shown in [11], for clean datasets and low false positive rates the Hessian-based detector of [10] outperforms learning-based ones. However, for true positive rates similar to the human performance, our method outperforms both [10] and [11].

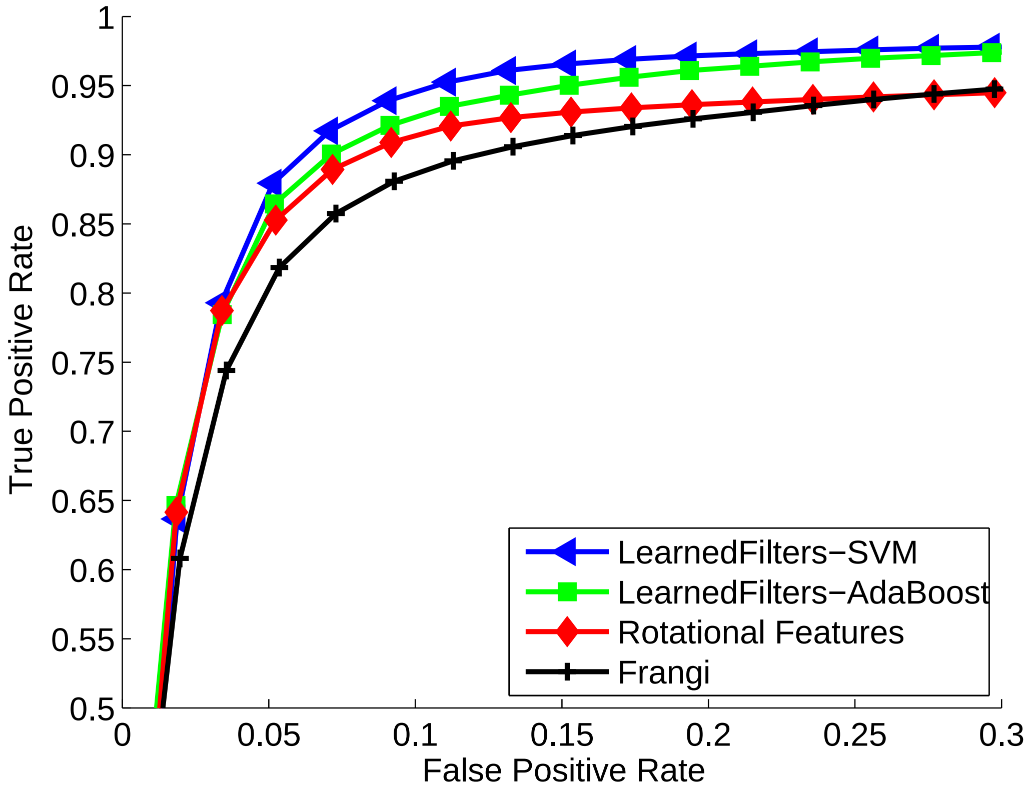

Figure 4.5: ROC curves for the neurons dataset. Our method clearly outperforms those of [10] and [11] over the whole ROC range. Learning a classifier improves the results, but learning both the filter bank and the classifier yields the best segmentation.

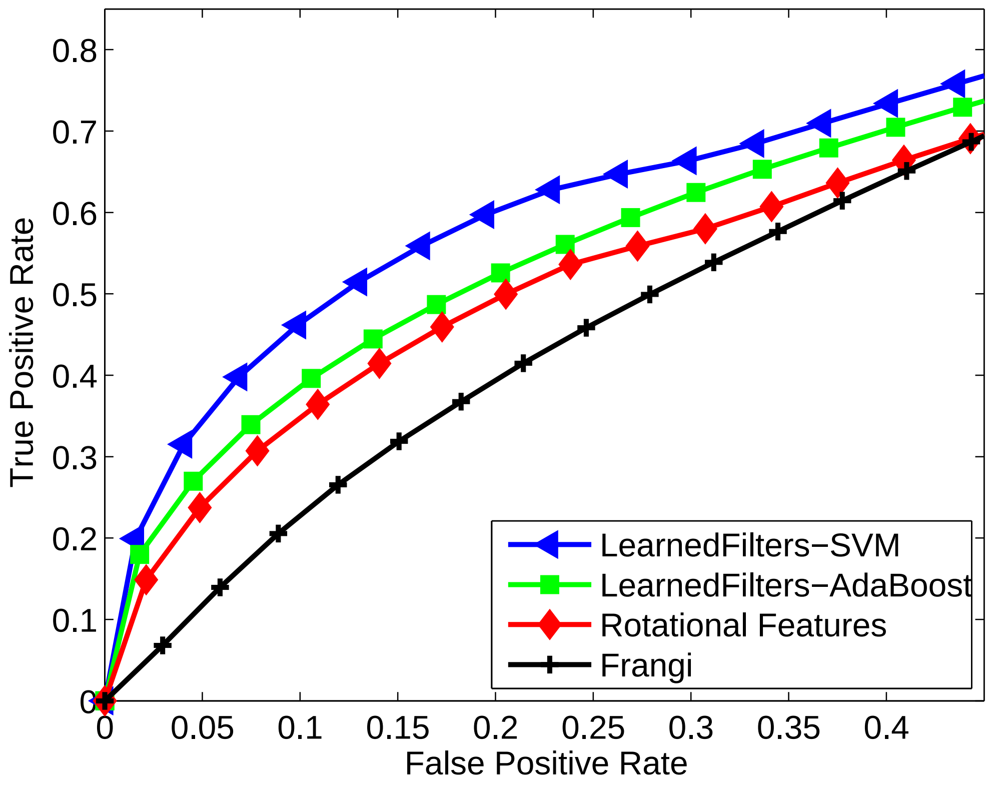

Figure 4.6: ROC curves obtained for the roads dataset. This dataset is the most challenging one and, as expected, yields the lowest True Positive Rates. Our method clearly outperforms [10] and [11] over the whole range.

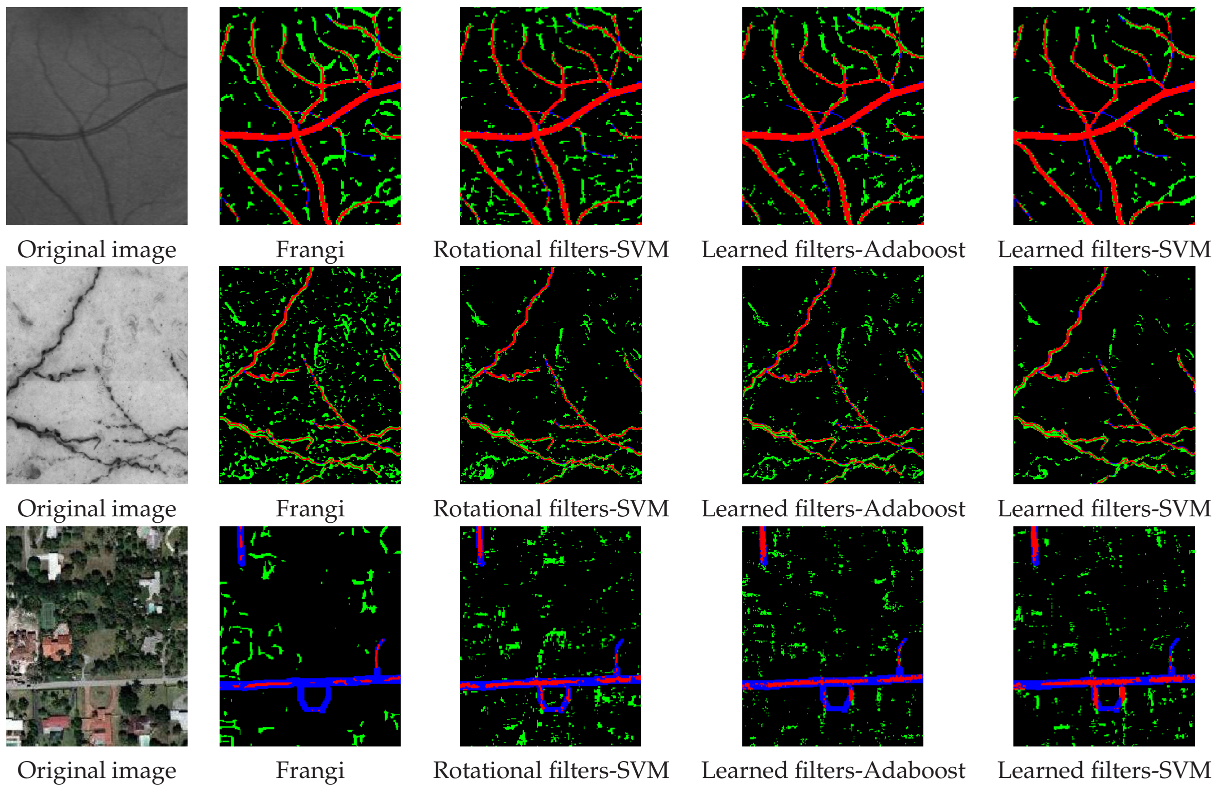

Figure 4.7: Segmentation results. True positive pixels are shown in red, false positives in green and false negatives in blue. **Top Row:** DRIVE dataset thresholded at a 90.6% TPR. **Middle Row:** Neurons on brightfield microscopy thresholded at 90% TPR. **Bottom Row:** Roads on satellite images thresholded at 5% FPR. In the first and second datasets the noise detected by our method is less than that of [10] and [11] when finding the same amount of true positives as a human observer. For the roads dataset, for an acceptable FPR, the proposed algorithm finds many more roads than [10] or [11].

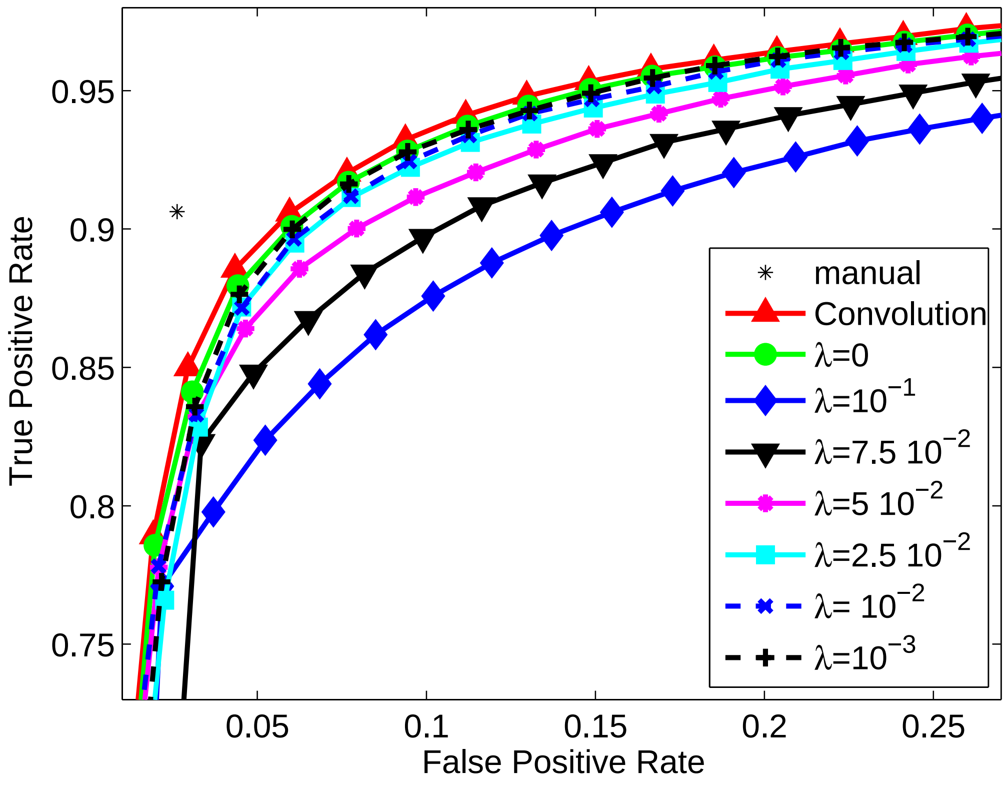

Figure 4.8: ROC curves corresponding to different refinement levels (classifier SVM, 2500 positive and 2500 negative samples).

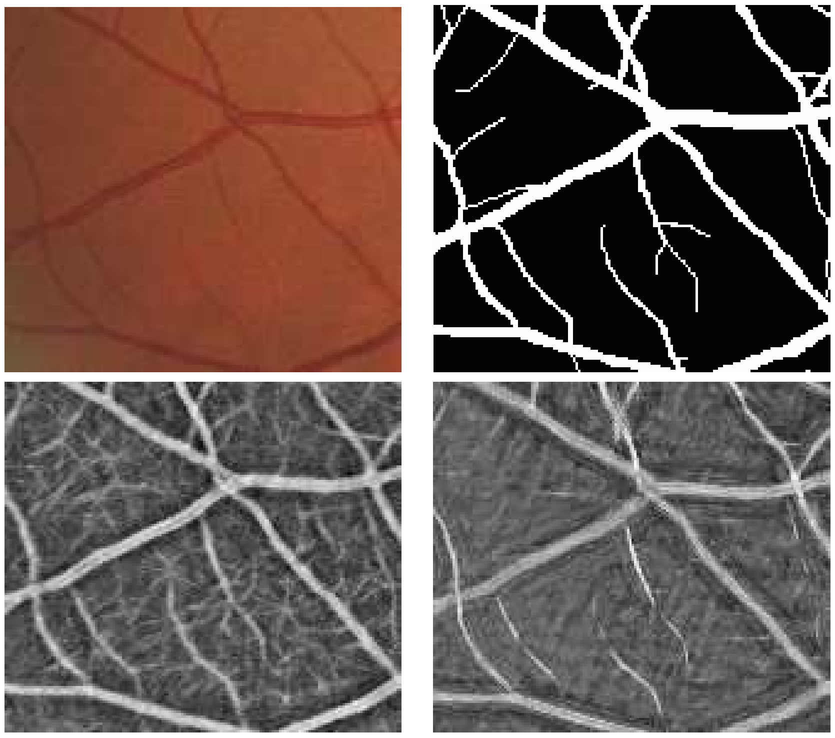

Figure 4.9: **Top:** Image patch from a test image from the DRIVE database along with its ground truth. **Bottom:** Output of the AdaBoost classifier when plain convolution (left) and sparse feature maps (with $\lambda_{segm} = 10^{-1}$, right) were used. In both cases, 2500 positive and 2500 negative samples were used, with 500 weak learners.

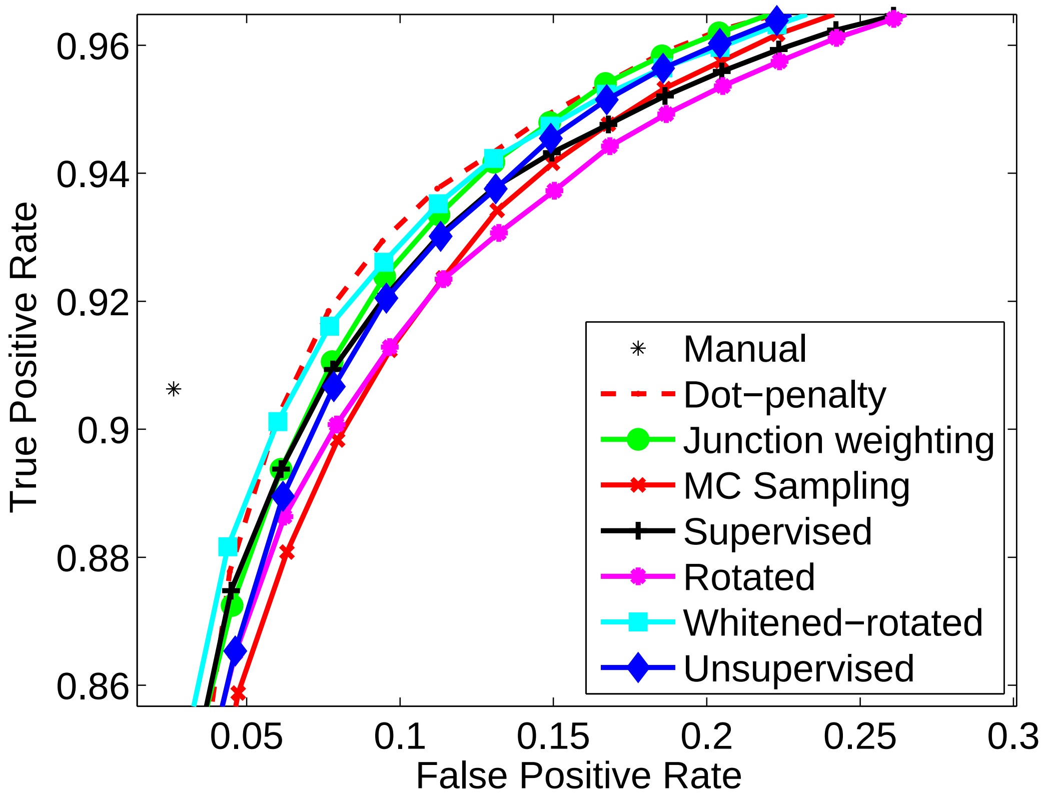

Figure 4.10: Comparison of the different supervised dictionary learning strategies with unsupervised techniques. The segmentation was performed using SVM with 1000 positive and 1000 negative samples. *Unsupervised* corresponds to the filter bank learned by optimizing Eq. (3.1), while *Dot-penalty* is learned by optimizing Eq. (3.2). *Junction weighting* weights more the reconstruction error in the regions close to junction points. *MC Sampling* adopts a weighted sampling scheme, preferring areas where junctions are present to those without junctions. *Supervised* is a filter bank learned according to Eq. (3.6). Junctions points are also taken into account. *Rotated* corresponds to a filter bank where supervision is paired up with a rotated version of the training set. Lastly, *Whitened-rotated* is the same as *Rotated* but learning is performed on whitened images.

supervision [18]. We have therefore decided to take advantage of the availability of ground truth data to evaluate how supervised strategies for dictionary learning perform in the segmentation task. We have considered three different supervision schemes:

- The reconstructions provided by the filters were penalized according to their difference from the ground truth image, as in Eq. (3.6).
- We have added a Gaussian weight at the position where junctions and bifurcations are located. These two particular configurations play indeed a key role in the construction of the tree that can be built from a given vessel image for subsequent processing. This weight had the role of focusing the algorithm's attention towards those areas, striving to obtain filters more suited to the reconstruction of these structures.
- MonteCarlo sampling was performed when patches were randomly sampled from the image during the training phase, in order to bias the type of structures shown to the learning algorithm. A patch was accepted even if it didn't contain a junction/intersection only in a fixed fraction of all the cases.

Figure 4.10 shows the ROC curves for the segmentation task corresponding to different combinations of the above strategies. The supervision terms seem not to be helpful in terms of the final segmentation score. This can be explained by the fact that, in both the supervised and unsupervised cases, the learned filter banks contain a rich description for the appearances of linear structures of interest. In the unsupervised case, this description is also coupled with a set of filters accounting for the structures in the background. However, the final supervised classification step suppresses these structures and selects the relevant features leaving the initial supervision for learning the filters redundant.

In one case we have fed the supervised dictionary learning algorithm with a rotated version of the dataset, where all the images and the ground truth were rotated at steps of $15°$ until the whole range of angles was covered. The rationale behind this experiment was to asses the degree of rotational invariance of the unsupervised filter bank by comparing it with a filter bank exposed to several different orientations during the learning stage. As can be seen in Fig. 4.10, this was unhelpful, and this could also have been deduced by observing how well the filter bank in Fig. 3.1 spans the whole range of orientations. We also wanted to measure the effectiveness of the whitening operation, and therefore we have learned a supervised filter bank on the rotated images pre-filtered with a whitening filter. Even though the performance were improved compared to the previous experiment, the simplest, unsupervised filter bank proved to be more effective. Further experiments with AdaBoost confirmed this trend.

Chapter 5

Conclusion

We showed we outperform the state-of-the-art in linear structure detection by learning a dictionary of convolution filters and then training an SVM to classify their output for individual pixels as filament-like or not. This is because the resulting filters are expressive enough to faithfully capture the various irregularities and deviations from strict tubularity one finds in noisy real-world data.

The major drawback of our filters is that they are not separable and therefore slower to compute than those that are. In future work, we will therefore focus on forcing them to either be sparse—that is, with many zero values— or, even, to be separable so as to speed-up the computation.

Bibliography

[1] G. Agam and C. Wu. Probabilistic Modeling-Based Vessel Enhancement in Thoracic CT Scans. In *CVPR'05*.

[2] K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, and B. Roysam. Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. *TITB*, 6(2):171–187, 2002.

[3] U. Bhattacharya and S. Parui. An Improved Backpropagation Neural Network for Detection of Road-Like Features in Satellite Imagery. *International Journal of Remote Sensing*, 1997.

[4] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-Level Features for Recognition. In *CVPR'10*.

[5] J. Canny. A Computational Approach to Edge Detection. *PAMI*, 8(6), 1986.

[6] A. Coates, H. Lee, and A. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *NIPS'10*.

[7] I. Daubechies, M. Defrise, and C. Mol. An Iterative Thresholding Algorithm for Linear Inverse Problems With a Sparsity Constraint. *Comm. Pure Appl. Math*, 2004.

[8] A. Dima, M. Scholz, and K. Obermayer. Automatic Segmentation and Skeletonization of Neurons from Confocal Microscopy Images Based on the 3D Wavelet Transform. *JMLR*, 11(7):790–801, 2002.

[9] D. Donoho. Compressed Sensing. *IEEE Transactions on Information Theory*, 2006.

[10] A. Frangi, W. Niessen, K. Vincken, and M. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1998.

[11] G. Gonzalez, F. Fleuret, and P. Fua. Learning Rotational Features for Filament Detection. In *CVPR'09*.

[12] 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*, 2009.

[13] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What Is the Best Multi-Stage Architecture for Object Recognition? In *CVPR'09*.

[14] 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, 2000.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[16] Y. LeCun, L. Bottou, G. Orr, and K. Müller. *Neural Networks: Tricks of the Trade*, chapter Efficient Back-Prop. 1998.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML'09*.

[18] J. Mairal, F. Bach, and J. Ponce. Task-Driven Dictionary Learning. Technical report, INRIA, 2010.

[19] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR'08*.

[20] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-Local Sparse Models for Image Restoration. In *ICCV'09*.

[21] E. Meijering, M. Jacob, J.-C. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. *Cytometry*, 2004.

[22] V. Mnih and G. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *ECCV'10*.

[23] M. Patasius, V. Marozas, D. Jegelevicius, and A. Lukoševičius. Ranking of Color Space Components for Detection of Blood Vessels in Eye Fundus Images. In *European Conference of the International Federation for Medical and Biological Engineering*, 2009.

[24] M. Ranzato, Y. Boureau, and Y. Lecun. Sparse Feature Learning for Deep Belief Networks. In *NIPS'07*.

[25] M. Ranzato, C. Poultney, S. Chopra, and Y. Lecun. Efficient Learning of Sparse Representations With an Energy-Based Model. In *NIPS'06*.

[26] R. Rigamonti, M. Brown, and V. Lepetit. Is Sparsity Really Relevant for Image Classification? Technical report, EPFL, 2010.

[27] A. Santamaría-Pang, C. M. Colbert, P. Saggau, and I. Kakadiaris. Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In *MICCAI'07*.

[28] 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. *Medical Image*

Analysis, 1998.

[29] S. Schmitt, J.-F. Evers, C. Duch, M. Scholz, and K. Obermayer. New Methods for the Computer-Assisted 3D Reconstruction of Neurons from Confocal Image Stacks. *NeuroImage*, 2004.

[30] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust Object Recognition With Cortex-Like Mechanisms. *PAMI*, 2007.

[31] R. Socher, A. Barbu, and D. Comaniciu. A Learning-Based Hierarchical Model for Vessel Segmentation. In *IEEE Symposium on Biomedical Imaging: From Nano to Macro*, 2008.

[32] G. Streekstra and J. van Pelt. Analysis of Tubular Structures in Three-Dimensional Confocal Images. *Network: Computation in Neural Systems*, 2002.

[33] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society*, 1996.

[34] J. Tyrrell, E. di Tomaso, D. Fuja, R. Tong, K. Kozak, R. Jain, and B. Roysam. Robust 3D Modeling of Vasculature Imagery Using Superellipsoids. *Medical Imaging*, 2007.

[35] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *CVPR'10*.